**Национальный исследовательский университет информационных технологий, механики и оптики**

**Факультет Программной Инженерии и Компьютерной Техники**

**Вариант № 27**

**Лабораторная работа №3**

**По дисциплине:**

**«Информатика»**

Работу выполнила:

Студентка группы P3112

Сенина Мария Михайловна

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург

2020

**Текст задания**

1. Определить номер варианта как остаток деления на 35 порядкового номера в списке группы в ISU. В случае, если в данный день недели нет занятий, то увеличить номер варианта на восемь.
2. Изучить форму Бэкуса-Наура.
3. Изучить особенности языков разметки/форматов JSON, YAML, XML, PROTOBUF.
4. Понять устройство страницы с расписанием для своей группы: http://www.ifmo.ru/ru/schedule/0/P3112/schedule.htm
5. 5. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного.
6. Написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый.
7. Нельзя использовать готовые библиотеки, кроме re (регулярные выражения в Python) и библиотеки для загрузки XML-файлов.
8. Необязательное задание для получения оценки «4» и «5» (позволяет набрать от 75 до 89 процентов от максимального числа баллов БаРС за данную лабораторную).
9. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
10. Сравнить полученные результаты и объяснить их сходство/различие.
11. Необязательное задание для получения оценки «5» (позволяет набрать от 90 до 100 процентов от максимального числа баллов БаРС за данную лабораторную).
12. Используя свою программу и найденные готовые библиотеки, сравнить десятикратное время выполнения парсинга + конвертации в цикле.
13. Проанализировать полученные результаты и объяснить их сходство/различие.
14. Проверить, что все пункты задания выполнены и выполнены верно.
15. Написать отчёт о проделанной работе.
16. Подготовиться к устным вопросам на защите.

**Исходный код программы**

**Табличка для четверга в формате XML:**

*<?*xml version="1.0" encoding="utf-8" *?>*<day\_schedule>  
 <day\_name>Четверг</day\_name>  
 <class>  
 <time>8:20 - 9:50</time>  
 <room>466 ауд.</room>  
 <adress>Кронверкский пр., д.49, лит.А</adress>  
 <lesson\_name>Математика (базовый уровень)(Лек)</lesson\_name>  
 <lesson\_lector>Возианова Анна</lesson\_lector>  
 <lesson\_format>!</lesson\_format>  
 </class>  
 <class>  
 <time>10:00 - 11:30</time>  
 <week\_parity>Нечётная неделя</week\_parity>  
 <adress>Кронверкский пр., д.49, лит.А</adress>  
 <lesson\_name>Информатика(Лек): Актовый зал</lesson\_name>  
 <lesson\_lector>Балакшин Павел Валерьевич</lesson\_lector>  
 <lesson\_format>Очно - дистанционный</lesson\_format>  
 </class>  
 <class>  
 <time>10:00 - 11:30</time>  
 <week\_parity>Чётная неделя</week\_parity>  
 <adress>Кронверкский пр., д.49, лит.А</adress>  
 <lesson\_name>Программирование(Лек): Актовый зал</lesson\_name>  
 <lesson\_lector>Письмак Алексей Евгеньевич</lesson\_lector>  
 <lesson\_format>Очно - дистанционный</lesson\_format>  
 </class>  
 <class>  
 <time>11:40 - 13:10</time>  
 <week\_parity>Чётная неделя</week\_parity>  
 <adress>Кронверкский пр., д.49, лит.А</adress>  
 <lesson\_name>Основы профессиональной деятельности(Лек): Актовый зал</lesson\_name>  
 <lesson\_lector>Клименков Сергей Викторович</lesson\_lector>  
 <lesson\_format>Очно - дистанционный</lesson\_format>  
 </class>

<day\_schedule>

Ссылка на файл в репозитории: <https://github.com/senina-m/ITMO/blob/master/Informatics_2020-10-2P6/my-schedule.xml>

**Парсер-конвертер для моего файла из XML в JSON:**

import time  
  
  
class Node:  
 def \_\_init\_\_(self):  
 self.node\_name = **""** self.body = **""** self.children = []  
  
  
def print\_node(node, file):  
 file.write(node.node\_name + **": "** + node.body)  
  
  
def print\_children(node, file):  
 length = len(node.children)  
 if length != 0:  
 file.write(node.node\_name + **": {"**)  
 for j in range(0, length - 1):  
 print\_children(node.children[j], file)  
 file.write(**",** \n**"**)  
 print\_children(node.children[length - 1], file)  
 file.write(**"}"**)  
 if node.body != **""**:  
 print\_node(node, file)  
  
  
def write\_2\_file(node, file): *# печатаем структуру в файл* if node.node\_name == **"root"**:  
 for j in range(0, len(node.children) - 1):  
 print\_children(node.children[j], file)  
 file.write(**",** \n**"**)  
 print\_children(node.children[len(node.children) - 1], file)  
 else:  
 print\_children(node, file)  
  
  
start\_time = time.time()  
  
filename\_for\_parsing = **"my-schedule.xml"**filename\_for\_result = **"my-new-schedule\_1.json"**file\_read = open(filename\_for\_parsing, **"r"**, encoding=**'utf-8'**)  
file\_write = open(filename\_for\_result, **"w"**, encoding=**'utf-8'**)  
  
name = False  
body = False  
this\_name = **""**this\_body = **""**line = **""**array = file\_read.read().split(**'**\n**'**)  
for i in range(1, len(array)):  
 line = line + array[i]  
  
root = Node()  
root.node\_name = **"root"**this\_node = root  
parents = []  
k = -1  
for i in range(0, len(line)):  
 symbol = line[i]  
 if i < len(line) - 1:  
 next\_symbol = line[i + 1]  
 else:  
 next\_symbol = **""** if symbol == **"<"** and next\_symbol != **"/"**: *# Новый тег node* node = Node()  
 this\_node.children.append(node)  
 parents.append(this\_node)  
 k = k + 1  
 this\_node = node  
 name = True  
 body = False  
 if name and symbol != **"<"** and symbol != **">"**: *# читаем имя node* this\_name = this\_name + symbol  
 elif name and symbol == **">"**: *# записываю имя node* name = False  
 body = True  
 this\_node.node\_name = this\_name  
 this\_name = **""** if body and symbol != **">"** and symbol != **"<"**: *# читаем body* this\_body = this\_body + symbol  
 elif symbol == **"<"** and next\_symbol == **"/"**: *# записываем body, если тег закрылся* body = False  
 this\_node.body = this\_body  
 this\_body = **""** this\_node = parents[k]  
 parents.pop()  
 k = k - 1  
  
file\_write.write(**"{ "**)  
write\_2\_file(root, file\_write)  
file\_write.write(**"**\n**}"**)  
print(**"--- %s seconds ---"** % (time.time() - start\_time))

Ссылка на файл в репозитории: <https://github.com/senina-m/ITMO/blob/master/Informatics_2020-10-26/parser_1.py>

Результат парсинга:

{ day\_schedule: {day\_name: Четверг,   
class: {time: 8:20 - 9:50,   
room: 466 ауд.,   
adress: Кронверкский пр., д.49, лит.А,   
lesson\_name: Математика (базовый уровень)(Лек),   
lesson\_lector: Возианова Анна,   
lesson\_format: !},   
class: {time: 10:00 - 11:30,   
week\_parity: Нечётная неделя,   
adress: Кронверкский пр., д.49, лит.А,   
lesson\_name: Информатика(Лек): Актовый зал,   
lesson\_lector: Балакшин Павел Валерьевич,   
lesson\_format: Очно - дистанционный},   
class: {time: 10:00 - 11:30,   
week\_parity: Чётная неделя,   
adress: Кронверкский пр., д.49, лит.А,   
lesson\_name: Программирование(Лек): Актовый зал,   
lesson\_lector: Письмак Алексей Евгеньевич,   
lesson\_format: Очно - дистанционный},   
class: {time: 11:40 - 13:10,   
week\_parity: Чётная неделя,   
adress: Кронверкский пр., д.49, лит.А,   
lesson\_name: Основы профессиональной деятельности(Лек): Актовый зал,   
lesson\_lector: Клименков Сергей Викторович,   
lesson\_format: Очно - дистанционный}}  
}

**Парсер-конвертер с использованием специализированной библиотеки xmljson**

(Ссылка на сайт библиотеки <https://pypi.org/project/xmljson/> )

import json  
import xmljson  
import time  
  
from lxml.etree import fromstring  
  
start\_time = time.time()  
filename\_for\_parsing = **"my-schedule.xml"**filename\_for\_result = **"my-new-schedule\_2.json"**file\_read = open(filename\_for\_parsing, **"r"**, encoding=**'utf-8'**)  
file\_write = open(filename\_for\_result, **"w"**, encoding=**'utf-8'**)  
  
line = **""**array = file\_read.read().split(**'**\n**'**)  
for i in range(1, len(array)):  
 line = line + array[i]  
  
xml = fromstring(line)  
json\_text = json.dumps(xmljson.badgerfish.data(xml), ensure\_ascii=False).encode(**'utf8'**)  
file\_write.write(json\_text.decode())  
print(**"--- %s seconds ---"** % (time.time() - start\_time))

Ссылка на файл в репозитории <https://github.com/senina-m/ITMO/blob/master/Informatics_2020-10-26/parser_2.py>

Результат парсинга:

{"day\_schedule": {"day\_name": {"$": "Четверг"}, "class": [{"time": {"$": "8:20 - 9:50"}, "room": {"$": "466 ауд."}, "adress": {"$": "Кронверкский пр., д.49, лит.А"}, "lesson\_name": {"$": "Математика (базовый уровень)(Лек)"}, "lesson\_lector": {"$": "Возианова Анна"}, "lesson\_format": {"$": "!"}}, {"time": {"$": "10:00 - 11:30"}, "week\_parity": {"$": "Нечётная неделя"}, "adress": {"$": "Кронверкский пр., д.49, лит.А"}, "lesson\_name": {"$": "Информатика(Лек): Актовый зал"}, "lesson\_lector": {"$": "Балакшин Павел Валерьевич"}, "lesson\_format": {"$": "Очно - дистанционный"}}, {"time": {"$": "10:00 - 11:30"}, "week\_parity": {"$": "Чётная неделя"}, "adress": {"$": "Кронверкский пр., д.49, лит.А"}, "lesson\_name": {"$": "Программирование(Лек): Актовый зал"}, "lesson\_lector": {"$": "Письмак Алексей Евгеньевич"}, "lesson\_format": {"$": "Очно - дистанционный"}}, {"time": {"$": "11:40 - 13:10"}, "week\_parity": {"$": "Чётная неделя"}, "adress": {"$": "Кронверкский пр., д.49, лит.А"}, "lesson\_name": {"$": "Основы профессиональной деятельности(Лек): Актовый зал"}, "lesson\_lector": {"$": "Клименков Сергей Викторович"}, "lesson\_format": {"$": "Очно - дистанционный"}}]}}

При сравнении работы двух парсеров выяснилось:

1. Они дают одинаковый результат, но в разном форматировании.
2. Время работы моего парсера равно 0.0019936561584472656 секунды, то парсер на основе библиотеки работает за 0.001994609832763672 секунды. Видно, что различие начинается в шестом знаке после запятой и мой парсер работает быстрее. Так получается, потому что мой код не учитывает полного синтаксиса XML и работает не для любого.xml файла, в то время как парсер на основе библиотеки работает для любого .xml файла. Парсер бы работал порядка на два быстрее, если бы в нём не использовалась классы.

**Выводы по работе**

В ходе лабораторной работы я познакомилась с языками разметки JSON, YAML, XML, PROTOBUF, базовым синтаксисом языка Phyton (версии 3) и библиотекой xmljson и формой Бэкуса-Наура.