

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

Мегафакультет компьютерных технологий и управления  
Факультет программной инженерии и компьютерной техники

**Отчёт**  
**по лабораторной работе №4**  
**«Исследование протоколов, форматов обмена информацией**  
**и языков разметки документов»**  
**по дисциплине «Информатика»**

Вариант 13

Студент: Кожухин Иван Алексеевич, группа Р3118  
Преподаватель: Рыбаков Степан Дмитриевич

Санкт-Петербург  
2022

# Содержание

<b>1</b>	<b>Задание</b>	<b>2</b>
<b>2</b>	<b>Ход работы</b>	<b>3</b>
2.1	Обязательное задание . . . . .	3
2.2	Дополнительное задание №1 . . . . .	8
2.3	Дополнительное задание №2 . . . . .	8
2.4	Дополнительное задание №3 . . . . .	9
2.5	Дополнительное задание №4 . . . . .	9
<b>3</b>	<b>Вывод</b>	<b>11</b>
<b>4</b>	<b>Список использованной литературы</b>	<b>11</b>

# 1 Задание

1. Определить номер варианта как остаток деления на 36 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.

4. Понять устройство страницы с расписанием для своей группы:

<http://itmo.ru/ru/schedule/0/P3110/schedule.htm>

5. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного. При этом необходимо, чтобы в выбранном дне 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.

6. 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.

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

а) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.

б) Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.

с) Сравнить полученные результаты и объяснить их сходство/различие.

8. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).

а) Переписать исходный код, добавив в него использование регулярных выражений.

б) Сравнить полученные результаты и объяснить их сходство/различие.

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

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

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

10. Дополнительное задание №4 (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.

## 2 Ход работы

Исходники кода программ и файлов:

<https://github.com/troublegale/InfLab4>



Рис. 1: QR-код для доступа к репозиторию на GitHub

### 2.1 Обязательное задание

Мой вариант предполагал создание программы, осуществляющей парсинг и конвертацию XML-файла в файл формата JSON.

Содержимое исходного XML-файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedule>
  <wednesday>
    <lesson>
      <subject>FPA</subject>
      <type>Lab</type>
      <professor>Tkeshelashvili N. M.</professor>
      <start_time>11:40</start_time>
      <end_time>13:10</end_time>
      <building>Kronverksky 49</building>
      <classroom>1330</classroom>
      <form>In-class + remote</form>
      <weeks>Odd</weeks>
    </lesson>
    <lesson>
      <subject>FPA</subject>
      <type>Lab</type>
      <professor>Tkeshelashvili N. M.</professor>
      <start_time>13:30</start_time>
      <end_time>15:00</end_time>
      <building>Kronverksky 49</building>
      <classroom>1330</classroom>
      <form>In-class + remote</form>
      <weeks>Odd</weeks>
    </lesson>
  </wednesday>

```

```

        <subject>CS</subject>
        <type>Lab</type>
        <professor>Rybakov S. D.</professor>
        <start_time>11:40</start_time>
        <end_time>13:10</end_time>
        <building>Kronverksky 49</building>
        <classroom>2332</classroom>
        <form>In-class + remote</form>
        <weeks>Even</weeks>
    </lesson>
</lesson>
    <subject>CS</subject>
    <type>Lab</type>
    <professor>Rybakov S. D.</professor>
    <start_time>13:30</start_time>
    <end_time>15:00</end_time>
    <building>Kronverksky 49</building>
    <classroom>2332</classroom>
    <form>In-class + remote</form>
    <weeks>Even</weeks>
</lesson>
</wednesday>
</schedule>

```

Для выполнения задания я создал несколько функций, что позволило мне с большим удобством работать с полученными при парсинге данными.

```

def xml_to_array(name):
    file = open(name)
    file.readline()
    separator = 'Death Grips is online'
    separated = file.read().replace('\n', ' ').replace('\t', '')
    separated = separated.replace('> <', '>' +
                                separator + '<').replace('><', '>' + separator + '<')
    return separated.split(separator)

def separate(data):
    separator = 'Death Grips is online'
    return data.replace('>', '<').replace('<', '>' + separator +
                                         '<').split(separator)[1:-1]

```

```

def get_tag(data):
    return separate(data)[0]

def just_tag(data):
    tag = separate(data)[0]
    return tag.replace('<', '').replace('>', '')

def get_value(data):
    if len(separate(data)) > 1:
        return separate(data)[1].replace('<', '').replace('>', '')
    return 0

def find_array_tags(arr):
    array_tags = []
    for i in arr:
        if i == get_tag(i) and arr.count(get_tag(i)) > 1 and just_tag(i) not in array_tags:
            array_tags.append(just_tag(i))
    return array_tags

```

Непосредственно процесс парсинга:

```
json_file = open('timetable_str.json', 'w')
xml_array = xml_to_array('timetable.xml')
array_tags = find_array_tags(xml_array)

in_arr = False

for i in range(len(xml_array)):

    if i == 0:
        json_file.write('{')
    elif i == len(xml_array) - 1:
        json_file.write('}')
    else:

        data = xml_array[i]
        next_data = xml_array[i + 1]
        tag = get_tag(data)
        only_tag = just_tag(data)
        next_only_tag = just_tag(next_data)
        value = get_value(data)

        if only_tag in array_tags and not in_arr:
            json_file.write('"' + only_tag + '": [')
            in_arr = True

        if in_arr:
            if data == get_tag(data) and '/' not in data:
                json_file.write('{')
            if '/' in only_tag:
                if '/' + next_only_tag == only_tag:
                    json_file.write(', ')
                else:
                    json_file.write(']')
                    in_arr = False
            elif value:
                json_file.write('"' + only_tag + '": "' + value + ('", '
                                                                    if next_only_tag not in array_tags else '"}'))
        else:
            if '/' not in data:
                json_file.write('"' + only_tag + '": {')
            else:
                json_file.write('}')
```

Содержимое полученного файла в формате JSON (пропущенное через JSON-beautifier):

```
{
  "wednesday": {
    "lesson": [
      {
        "subject": "FPA",
        "type": "Lab",
        "professor": "Tkeshelashvili N. M.",
        "start_time": "11:40",
        "end_time": "13:10",
        "building": "Kronverksky 49",
        "classroom": "1330",
        "form": "In-class + remote",
        "weeks": "Odd"
      },
      {
        "subject": "FPA",
        "type": "Lab",
        "professor": "Tkeshelashvili N. M.",
        "start_time": "13:30",
        "end_time": "15:00",
        "building": "Kronverksky 49",
        "classroom": "1330",
        "form": "In-class + remote",
        "weeks": "Odd"
      },
      {
        "subject": "CS",
        "type": "Lab",
        "professor": "Rybakov S. D.",
        "start_time": "11:40",
        "end_time": "13:10",
        "building": "Kronverksky 49",
        "classroom": "2332",
        "form": "In-class + remote",
        "weeks": "Even"
      },
      {
        "subject": "CS",
        "type": "Lab",
        "professor": "Rybakov S. D.",
        "start_time": "13:30",
        "end_time": "15:00",
        "building": "Kronverksky 49",
        "classroom": "2332",
        "form": "In-class + remote",
        "weeks": "Even"
      }
    ]
  }
}
```



## 2.2 Дополнительное задание №1

Для выполнения этого задания я использовал библиотеки `xmldict` и `json`. Код программы:

```
xml_file = open('timetable.xml')
json_file = open('timetable_add1.json', 'w')

# parsing process
timetable = xmldict.parse(xml_file.read())

# converting to json
json_str = json.dumps(timetable)[: -1]
json_str = json_str.replace('}', '', 1)
json_str = json_str[: -1]
root = json_str.split()[0] + ' '
json_str = json_str.replace(root, '', 1)
json_file.write(json_str)
```

Содержимое результирующего файла полностью совпадает с содержимым файла, полученного по завершению работы программы для основного задания (форматом в одну строку).

## 2.3 Дополнительное задание №2

Для выполнения этого задания я скопировал код для основного задания и добавил в некоторых функциях работу с регулярными выражениями. Измененные фрагменты кода:

```
def get_tag(data):
    tag = re.search(r'</*\w+>', data)[0]
    return tag

def get_value(data):
    info = re.findall(r'[^<>\t]+' , data)
    if len(info) == 3:
        return info[1]
    else:
        return 0
```

Содержимое результирующего файла полностью совпадает с содержимым файла, полученного по завершению работы программы для основного задания (форматом в одну строку).

## 2.4 Дополнительное задание №3

Для измерения времени работы программ я использовал библиотеку `time`. Я поместил части программ, в которых происходит парсинг и конвертация, в цикл, который выполнялся 100 раз. Перед этим циклом я присвоил переменной `start_time` значение `time.time()`, зафиксировав время начало работы. После цикла я присвоил переменной `end_time` значение `time.time()`, зафиксировав время завершения работы. Затем я вычислил разность `end_time` и `start_time`, округлив её до 3 знаков после запятой, и вывел это значение на экран. В результате замеров были получены следующие данные:

Метод выполнения задания	Среднее время работы программы
Без использования сторонних библиотек и рег. выражений	≈ 0.05 сек
С использованием сторонних библиотек	≈ 0.045 сек
С использованием регулярных выражений	≈ 0.063 сек

Вариант с использованием сторонних библиотек оказался самым быстродейственным, обогнав стандартный вариант на 0,005 секунды. Предположу, что это можно объяснить многократным использованием конструкции `if/else` в стандартном варианте, а также моей некомпетентностью в сфере оптимизации кода на языке Python. Вариант с использованием регулярных выражений оказался наименее быстродейственным, уступив стандартному целых 0,013 секунды. Регулярные выражения показали себя во всей красе.

## 2.5 Дополнительное задание №4

Для выполнения этого задания я выбрал формат TSV (tab-separated values). Большинство функций я скопировал из кода программы для основного задания, а вот фрагмент кода с конвертацией файла:

```
def xml_to_array(name):
    file = open(name)
    file.readline()
    separator = 'Death Grips is online'
    separated = file.read().replace('\n', ' ').replace('\t', '')
    separated = separated.replace('> <', '>' + separator + '<').replace('><', '>' + separator + '<')
    return separated.split(separator)

def separate(data):
    separator = 'Death Grips is online'
    return data.replace('>', '<').replace('<', '>' + separator + '<').split(separator)[1:-1]

def get_tag(data):
    return separate(data)[0]

def just_tag(data):
    tag = separate(data)[0]
    return tag.replace('<', '').replace('>', '')
```

```

def get_value(data):
    if len(separate(data)) > 1:
        return separate(data)[1].replace('<', '>').replace('>', '<')
    return 0

# parsing process
tsv_file = open('timetable.tsv', 'w')
xml_array = xml_to_array('timetable.xml')

# converting to tsv
for i in range(len(xml_array) - 1):
    if get_value(xml_array[i]):
        tsv_file.write(just_tag(xml_array[i]) + ('\t' if get_value(xml_array[i + 1]) else '\n'))
    if not get_value(xml_array[i + 1]):
        break
for i in range(len(xml_array) - 1):
    if get_value(xml_array[i]):
        tsv_file.write(get_value(xml_array[i]) + ('\t' if get_value(xml_array[i + 1]) else '\n'))

# result representation
tsv_file.close()
result = open('timetable.tsv')
print(result.read())

```

Содержимое полученного TSV-файла (элементы каждой строки разделены знаком табуляции):

```

subject type professor start_time end_time building classroom form weeks
FPA Lab Tkeshelashvili N. M. 11:40 13:10 Kronverksky 49 1330 In-class + remote Odd
FPA Lab Tkeshelashvili N. M. 13:30 15:00 Kronverksky 49 1330 In-class + remote Odd
CS Lab Rybakov S. D. 11:40 13:10 Kronverksky 49 2332 In-class + remote Even
CS Lab Rybakov S. D. 13:30 15:00 Kronverksky 49 2332 In-class + remote Even

```

Конвертация в TSV-формат оказалась проще, чем в формат JSON, во многом потому, что в формате TSV данные разделены только знаками табуляции и по строчкам.

### 3 Вывод

В ходе лабораторной работы я изучил форму Бэкуса–Наура, изучил особенности языков разметки XML, JSON и TSV, научился парсингу данных с использованием языка Python, узнал о различиях работы без использования дополнительных библиотек и с ним.

### 4 Список использованной литературы

1. Балакшин П. В. Информатика: лабораторные работы и тесты / П.В. Балакшин, В.В. Соснин, И.В. Калинин, Т.А. Малышева, С.В. Раков, Н.Г. Рущенко, А.М. Дергачев – СПб.: Университет ИТМО, 2019. - 59 с.
2. Сервис JSONOnlineTools [электронный ресурс]. – Режим доступа: <https://onlinejsontools.com/> (дата обращения: 27.11.2022)
3. Сервис Hilite.me [электронный ресурс]. – Режим доступа: <http://hilite.me/> (дата образования: 27.11.2022)
4. Лямин А.В., Череповская Е.Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб.: Университет ИТМО, 2017. - 143 с. – Режим доступа: <https://books.ifmo.ru/file/pdf/2256.pdf> (дата обращения: 27.11.2022)