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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

По информатике

Исследование протоколов, форматов обмена информацией и языков разметки документов

Вариант №6

Выполнил:

Ступин Тимур Русланович

Группа № P3108

Проверил:

Балакшин Павел Валерьевич

Кандидат технических наук

Доцент факультета ПИиКТ

Санкт-Петербург 2023

Содержание

Задание ... 3

Основные этапы вычисления ... 4

Определение номера варианта ... 4

Формирование файла с расписанием ... 5

Обязательное задание ... 6

Дополнительное задание 1 ... 8

Дополнительное задание 2 ... 10

Дополнительное задание 3 ... 13

Дополнительное задание 4 ... 21

Дополнительно задание 5 ... 22

Заключение ... 23

Список использованных источников ... 23

Задание

1) Определить номер варианта как остаток деления на 36 последних двух цифр своего идентификационного номера в ISU. В случае, если в данный день недели нет занятий, то увеличить номер варианта 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 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.
7) Обязательное задание (позволяет набрать до 45 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 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

Основные этапы вычисления

Определение номера варианта

Номер ИСУ: 409642 => номер варианта = 42 % 36 = 6

День – Вторник(Рисунок 1). Перевод: JSON→XML

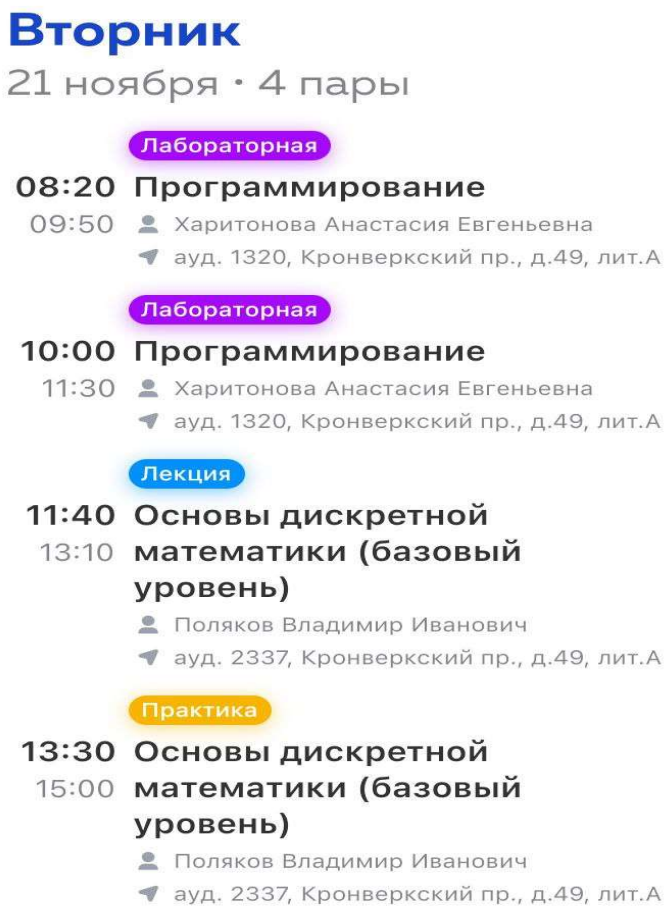

Рисунок 1

Формирование файла с расписанием

На основание расписания на день из варианта(Рисунок 1) составлено расписание в формате JSON:


```
{
  "dayTable1": {
    "day": "Вторник",
    "lesson1": {
      "time": {
        "start": "8:20",
        "end": "9:50"
      },
      "fiber": "Прогр 2.6",
      "room": "ауд. 1320",
      "address": "Кронверкский пр., д.49, лит.А",
      "lesson": "Программирование",
      "lecturer": "Харитонова Анастасия Евгеньевна",
      "lesson-format": "Очный"
    },
    "lesson2": {
      "time": {
        "start": "10:00",
        "end": "11:30"
      },
      "fiber": "Прогр 2.6",
      "room": "ауд. 1320",
      "address": "Кронверкский пр., д.49, лит.А",
      "lesson": "Программирование",
      "lecturer": "Харитонова Анастасия Евгеньевна",
      "lesson-format": "Очный"
    },
    "lesson3": {
      "time": {
        "start": "11:40",
        "end": "13:10"
      },
      "fiber": "ДИСКР МАТ 2",
      "room": "ауд. 2337",
      "address": "Кронверкский пр., д.49, лит.А",
      "lesson": "Основы дискретной математики (базовый уровень)",
      "lecturer": "Поляков Владимир Иванович",
      "lesson-format": "Очный"
    },
    "lesson4": {
      "time": {
        "start": "13:30",
        "end": "15:00"
      },
```

```
      "fiber": "ДИСКР МАТ 2",
      "room": "ауд. 2337",
      "address": "Кронверкский пр., д.49, лит.А",
      "lesson": "Основы дискретной математики (базовый уровень)",
      "lecturer": "Поляков Владимир Иванович",
      "lesson-format": "Очный"
    }
  }
}
```

Обязательное задание

На я языке программирование python была написана программа для конвертации файла расписания в формате JSON в файл формата XML. Данная программа основана на построчном анализе JSON файла, и поддерживает перевод словарей из формата JSON в формат XML, в том числе вложенных. Стоит также отметить, что из-за вложенной структуры форматов JSON и XML и необходимости сохранять эту структуру при расстановки закрывающих тегов в формате XML в программе был использован стек, в котором хранятся значения открытых в настоящих момент тегов, что даёт возможность корректно обрабатывать вложенные словари.

Программа представляет собой функцию, которая принимает на вход имена входного и выходного файлов, после чего производится конвертирование.

Листинг программы приведён ниже:

```python
#Основная функция для перевода
def JSON2XML0(inputFile, outputFile):
    # Читаем данные из JSON файла построчно в массив
    with open(inputFile, mode='r', encoding='utf8') as f:
        lines = f.readlines()
    res = '<?xml version="1.0" encoding="UTF-8" ?>\n<root>\n'
    stack = ['root']
    for line in lines[1:]:
        #Запоминаем пробелы
        tab = ''
        for i in line:
            if i != ' ':
                break
            tab += i
        #Убираем пробелы
        s = line.strip()
        #Закрывающие тег
        if s[0] == '}':
            res += f'{tab}</{stack[-1]}>\n'
            stack.pop()
            continue
```

```
            #Получаем ключ
            key = ''
            ind = 1
            while s[ind] != '"':
                key += s[ind]
                ind += 1
            ind += 3
            #Открывающий тег
            if s[ind] == '{':
                res += f'{tab}<{key}>\n'
                stack.append(key)
                continue
            #Получаем значение одинрного тега
            val = ''
            ind += 1
            while s[ind] != '"':
                val += s[ind]
                ind += 1
            res += f'{tab}<{key}>{val}</{key}>\n'

    # Записываем в файл
    output_file = open(outputFile, "w", encoding='utf8')
    output_file.write(res)
```

Выходной файл в формате XML имеет следующий вид:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <dayTable1>
    <day>Вторник</day>
    <lesson1>
      <time>
        <start>8:20</start>
        <end>9:50</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson1>
    <lesson2>
      <time>
        <start>10:00</start>
        <end>11:30</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
```

```
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson2>
    <lesson3>
      <time>
        <start>11:40</start>
        <end>13:10</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson3>
    <lesson4>
      <time>
        <start>13:30</start>
        <end>15:00</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson4>
  </dayTable1>
</root>
```

Дополнительное задание 1

Для перевода между форматами были использованы следующие библиотеки:

- json – для чтение входного файла и преобразования его в python словарь
- dicttoxml – для перевода словаря в строку в формате xml
- модуль parseString модуль xml.dom.minidom – для преобразования однострочного представления xml файла в форматированный вид(расстановка переводов строк и табуляции)

Программа представляет собой функцию, которая принимает на вход имена входного и выходного файлов, после чего производится конвертирование.

Листинг программы приведён ниже:

```
import json
from dicttoxml import dicttoxml
from xml.dom.minidom import parseString

def JSON2XML1(inputFile, outputFile):
    # Открываем son файл
    f = open(inputFile, encoding='utf8')
    # Преобразуем его в словарь
    json_obj = json.load(f)
    # Получаем строку xml
    xml_obj = dicttoxml(json_obj, attr_type=False)
    # Открыли файл на запись
    output_file = open(outputFile, "w", encoding='utf8')
    # Получем отворматированную строку
    parsed_string = parseString(xml_obj).toprettyxml()
    # Записываем в файл
    output_file.write(parsed_string)
```

Выходной файл в формате XML имеет следующий вид:

```
<?xml version="1.0" ?>
<root>
    <dayTable1>
        <day>Вторник</day>
        <lesson1>
            <time>
                <start>8:20</start>
                <end>9:50</end>
            </time>
            <fiber>Прогр 2.6</fiber>
            <room>ауд. 1320</room>
            <address>Кронверкский пр., д.49, лит.А</address>
            <lesson>Программирование</lesson>
            <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
            <lesson-format>Очный</lesson-format>
        </lesson1>
        <lesson2>
            <time>
                <start>10:00</start>
                <end>11:30</end>
            </time>
            <fiber>Прогр 2.6</fiber>
            <room>ауд. 1320</room>
            <address>Кронверкский пр., д.49, лит.А</address>
            <lesson>Программирование</lesson>
            <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
            <lesson-format>Очный</lesson-format>
```

```
        </lesson2>
        <lesson3>
            <time>
                <start>11:40</start>
                <end>13:10</end>
            </time>
            <fiber>ДИСКР МАТ 2</fiber>
            <room>ауд. 2337</room>
            <address>Кронверкский пр., д.49, лит.А</address>
            <lesson>Основы дискретной математики (базовый
уровень)</lesson>
            <lecturer>Поляков Владимир Иванович</lecturer>
            <lesson-format>Очный</lesson-format>
        </lesson3>
        <lesson4>
            <time>
                <start>13:30</start>
                <end>15:00</end>
            </time>
            <fiber>ДИСКР МАТ 2</fiber>
            <room>ауд. 2337</room>
            <address>Кронверкский пр., д.49, лит.А</address>
            <lesson>Основы дискретной математики (базовый
уровень)</lesson>
            <lecturer>Поляков Владимир Иванович</lecturer>
            <lesson-format>Очный</lesson-format>
        </lesson4>
    </dayTable1>
</root>
```

С точки зрения содержания данный файл полностью идентичен предыдущему Отличие заключается в размере табуляции(здесь она из 4 пробелов вместо 2), что является особенность форматирования библиотеки parseString, а также содержание первой строчки: при переводе с помощью библиотек данная строчка содержит только информацию о версии XML, без кодировки, что также является особенностью библиотеки.

Дополнительное задание 2

Данная программа написана с использование регулярных выражений. Теперь json файл читается сразу в одну строку, после чего осуществляется замена метасимволов с использованием регулярных выражений. В начале из строки выделяются все заголовки словарей, после чего производится замена закрывающий скобок на закрывающие теги, с сохранением исходной вложенности, что достигается использованием стека. После этого при помощи метода `re.sub` осуществляется замена названий словарей на теги.

Программа представляет собой функцию, которая принимает на вход имена входного и выходного файлов, после чего производится конвертирование.

Листинг программы приведён ниже:

```
import re

#Основная функция для перевода
def JSON2XML2(inputFile, outputFile):
    # Читаем данные из JSON файла построчно в массив
    with open(inputFile, mode='r', encoding='utf8') as f:
        s = f.read()
    #Добавляем тег root и стандартную шапку
    s = '<?xml version="1.0" encoding="UTF-8" ?>\n<root>' + s[1:]
    # Получаем все заголоки словарей
    dicts = []
    for i in re.findall(pattern=r'(\")(.*)(\")(\s*\:\s*\{)',
string=s):
        dicts.append(i[1])
    # Убираем запятые
    s = re.sub(pattern=r'},', repl=r'}', string=s)
    #Заменяем все закрывающие скобки на закрывающие теги
    ptr = 0
    stack = ['root']
    b = ''
    for i in range(len(s)):
        if s[i] == '}':
            b += f'</{stack[-1]}>'
            stack.pop()
            continue
        if s[i] == '{':
            stack.append(dicts[ptr])
            ptr += 1
        b += s[i]
    s = b
    # Заменяем заголовки словарей
    s = re.sub(pattern=r'(\")(.*)(\")(\s*\:\s*\{)', repl=r'<\2>',
string=s)
    # Заменяем основные поля
    s =
re.sub(pattern=r'(\")(.*)(\")(\s*\:\s*\"?)(.+?)(\"?\s*\,?\n)',
repl=r'<\2>\5</\2>\n', string=s)
    # Записываем в файл
    output_file = open(outputFile, "w", encoding='utf8')
    output_file.write(s)
```

Выходной файл в формате XML имеет следующий вид:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<root>
  <dayTable1>
    <day>Вторник</day>
    <lesson1>
      <time>
        <start>8:20</start>
        <end>9:50</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson1>
    <lesson2>
      <time>
        <start>10:00</start>
        <end>11:30</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson2>
    <lesson3>
      <time>
        <start>11:40</start>
        <end>13:10</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson3>
    <lesson4>
      <time>
        <start>13:30</start>
        <end>15:00</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
```

```
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson4>
  </dayTable1>
</root>
```

Данный файл полностью идентичен файлу, полученному в результате работа изначальной программы из основного задания, так как принцип работы программы очень похож, отличается лишь способ реализации.

Дополнительное задание 3

В данном задание мною был реализован полноценный универсальный парсер для формата JSON, который дает на выходе python словарь, а также написан алгоритм для перевода словаря в файл xml, с сохранением форматирования.

Файл JSON читается в строку, после чего при помощи прохода производится удаление лишних пробелов, то есть тех, которые находятся вне строк. Знаки перевода строки также удаляются. Дальнейшая обработка строки производится при помощи рекурсивных функций, которые производят посимвольный анализ строки и выделяют из нее базовые объекты языка разметки JSON: словарь, список, строки, число, null.

После получения словаря при помощи похожих рекурсивных функций производится его перевод в строку XML. Благодаря рекурсии достигается возможность правильное расстановки табуляций, а также корректная расстановка тегов в соответствие со структурой исходного файла. В процессе создания результирующей строки при помощи регулярных выражений производится фильтрация и замета спецсимволов, которые допустимы в формате JSON, но требуют особой обратки в формате XML.

Так как формат JSON описывается контекстно-свободной формальной грамматикой LL(k), то при посимвольном разборе строки, для определения дальнейшего действия необходимы иметь информацию не более чем о k символах впереди. Это значит что для парсинге формата JSON можно использовать набор рекурсивных функций которые и производят посимвольный разбор файла, спускаясь всё глубже по его древовидной структуре. Это также является обоснование того, что рекурсивный парсер является полным и универсальным.

Листинг программы приведён ниже:

```
import re

#Размер табуляции
tabVal = '  '
#Стандартный текст в начале
standartStr = '<?xml version="1.0" encoding="UTF-8" ?>'
#Словарь со специальными символами которые необходимо заменять
specialSymbolsDict = {
    '<': '&lt;',
    '>': '&gt;',
    '&': '&amp;',
    '"': '&quot;',
    "'": "&apos;"
}
#Список со всеми спецсимволами
allSpecialSymbols = ':! "#$%&\'()*+,/;<=>? @[\]^`{|}~'

ind = 0
#Символы которые можно пропускать
separators = [',', ' ', ':', '\n']

#Получение строки
def getString(s):
    global ind
    #Скип открывающих ковычек
    ind += 1
    #Ищем первое строку с помощью регулярки обрабатываея
жкранирование
    res = re.search(r'(.*?[^\\])(\")', s[ind:]).groups()[0]
    #Смещаем индекс для корректной дальнейшей работы
    ind += len(res) + 1
    return res
#Получение литерала
def getLiteral(s):
    global ind
    #Ищем первое строку с помощью регулярки обрабатываея
жкранирование
    res = re.search(r'(\w*)(\}|\]|\,|$)', s[ind:]).groups()[0]
    #Смещаем индекс для корректной дальнейшей работы
    ind += len(res)

    if res == 'true': return True
    if res == 'false': return False
    if res == 'null': return None
#Получение целого числа
def getNum(s):
    global ind
    # Ищем первое строку с помощью регулярки обрабатываея
жкранирование
```

```
    res = re.match(r'(.+?)(\}|\]|\,|$)', s[ind:]).groups()[0]
    # Смещаем индекс для корректной дальнейшей работы
    ind += len(res)
    #nan
    if res == 'NaN':
        return float('nan')
    #inf
    if res == 'Infinity':
        return float('inf')
    if res == '-Infinity':
        return -float('inf')
    #Целое число
    if re.fullmatch(r'\-?\d+', res):
        return int(res)
    return float(res)

#Получение словаря из json строки
def getDict(s):
    global ind
    dict = {}
    while ind < len(s):
        skipSeps(s)
        #Конец словаря
        if s[ind] == '}':
            ind += 1
            break
        skipSeps(s)
        #Получаем имя ключа
        key = getString(s)
        #Получаем значние
        dict[key] = parse(s)
    return dict
#Получение списка из json строки
def getList(s):
    global ind
    list = []
    while ind < len(s):
        # Конец списка
        if s[ind] == ']':
            ind += 1
            break
        # Получаем значние
        list.append(parse(s))
    return list

#Определение чего надо парсить(для начала парсинга)
def parse(s):
    global ind
    #Пропускаем ненужные символы
```

```
    skipSeps(s)
    #Словарь
    if s[ind] == '{':
        ind += 1
        return getDict(s)
    #Список
    if s[ind] == '[':
        ind += 1
        return getList(s)
    # Ключ задает строковой значение
    if s[ind] == '"':
        return getString(s)
    # Ключ задает литерал
    if s[ind] in ['t', 'f', 'n']:
        return getLiteral(s)
    # Ключ задает число
    return getNum(s)
#Пропуск разделителей
def skipSeps(s):
    global ind
    while s[ind] in separators: ind += 1

#Проверяет строку на наличие специальных символов
def ckeckSpecialSymbols(s):
    if re.fullmatch(r'\d+\w+', s): return True
    if s[0] == '-': return True
    for c in allSpecialSymbols:
        if c in s:
            return True
    return False
#заменяет спецсимволы на читаемые XML форматом
def replaceSpecialSymbols(s):
    res = ''
    #Проверяем на особые символы
    for c in s:
        if c in specialSymbolsDict:
            res += specialSymbolsDict[c]
        else:
            res += c
    return res
#Нормализует название тега
def normalizeKey(s):
    #Убираем экранирование
    s = re.sub(r'([^\\])(\\)([^\\])', r'\1\3', s)
    s = re.sub(r'\\\\', r'\\', s)
    #Заменяем спецсимволы
    s = replaceSpecialSymbols(s)
    #Проверяем на число
    if re.fullmatch(r'\d*', s):
```

```
            return 'n' + s
        return s

def getTag(item, curTab, openTag, closeTag):
    tab = tabVal * curTab
    # Строка
    if type(item) == type(''):
        return
f'{tab}<{openTag}>{replaceSpecialSymbols(item)}</{closeTag}>\n'
    #Число
    if type(item) == type(0) or type(item) == type(0.0):
        return f'{tab}<{openTag}>{item}</{closeTag}>\n'
    # Словрь
    if type(item) == type({}):
        return f'{tab}<{openTag}>\n{dictToXml(item, curTab +
1)}{tab}</{closeTag}>\n'
    # Список
    if type(item) == type([]):
        return f'{tab}<{openTag}>\n{listToXML(item, curTab +
1)}{tab}</{closeTag}>\n'
    # литералы
    if item == True:
        return f'{tab}<{openTag}>True</{closeTag}>\n'
    if item == False:
        return f'{tab}<{openTag}>False</{closeTag}>\n'
    if item == None:
        return f'{tab}<{closeTag}/>\n'

#Переводит список в строку формата XML
#На вход получает словарь и текущий отступ
def dictToXml(dict, curTab):
    res = ''

    for item in dict:
        #Определяем имя тега
        openTag = item
        closeTag = item
        #Проводим проверку на спецсимволы
        if ckeckSpecialSymbols(item):
            openTag = f'key name="{normalizeKey(item)}"'
            closeTag = 'key'

        res += getTag(dict[item], curTab, openTag, closeTag)
    return res

#Переводит список в строку формата XML
def listToXML(list, curTab):
    res = ''
```

```
    for item in list:
        res += getTag(item, curTab, 'item', 'item')

    return res

#Основная функция для перевода
def JSON2XML3(inputFile, outputFile):
    # Читаем данные из JSON файла построчно в массив
    with open(inputFile, mode='r', encoding='utf8') as f:
        s = f.read()
    #Перводим глобальный индекс на второй символ строки
    global ind
    ind = 0
    # Парсим строку в словарь
    dict = parse(s)
    # Добавляем тег root
    dict = {'root': dict}
    # Переводим словарь в XML
    res = f'{standartStr}\n{dictToXml(dict, 0)}'
    # Записываем в файл
    output_file = open(outputFile, "w", encoding='utf8')
    output_file.write(res)

def main():
    JSON2XML3(inputFile='day.json', outputFile='day3.xml')

if __name__ == '__main__':
    main()
```

Выходной файл в формате XML имеет следующий вид:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <dayTable1>
    <day>Вторник</day>
    <lesson1>
      <time>
        <start>8:20</start>
        <end>9:50</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson1>
    <lesson2>
```

```
      <time>
        <start>10:00</start>
        <end>11:30</end>
      </time>
      <fiber>Прогр 2.6</fiber>
      <room>ауд. 1320</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Программирование</lesson>
      <lecturer>Харитонова Анастасия Евгеньевна</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson2>
    <lesson3>
      <time>
        <start>11:40</start>
        <end>13:10</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson3>
    <lesson4>
      <time>
        <start>13:30</start>
        <end>15:00</end>
      </time>
      <fiber>ДИСКР МАТ 2</fiber>
      <room>ауд. 2337</room>
      <address>Кронверкский пр., д.49, лит.А</address>
      <lesson>Основы дискретной математики (базовый
уровень)</lesson>
      <lecturer>Поляков Владимир Иванович</lecturer>
      <lesson-format>Очный</lesson-format>
    </lesson4>
  </dayTable1>
</root>
```

Результат полностью идентичен результату работы первой и второй программ.

Для проверки универсальности данного решения был подготовлен тестовый файл:

```
{
  "InfTag": -Infinity,
  "expTag": 1e-9,
  "-InfTag": -Infinity,
  "Na-N": NaN,
```

```
  "nums": [1, -1, 1.3156e9, ["something", true],
  "!&@#&@)!sdf> <dpoiu", {"dv s": {"ins": NaN, "null": null}}]
}
```

Результат перевода при помощи моего решения:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <InfTag>-inf</InfTag>
  <expTag>1e-09</expTag>
  <key name="-InfTag">-inf</key>
  <Na-N>nan</Na-N>
  <nums>
    <item>1</item>
    <item>-1</item>
    <item>1315600000.0</item>
    <item>
      <item>something</item>
      <item>True</item>
    </item>
    <item>!&amp;@#&amp;@)!sdf&gt; &lt;dpoiu</item>
    <item>
      <key name=" dv s">
        <ins>nan</ins>
        <null/>
      </key>
    </item>
  </nums>
</root>
```

Результат перевода при помощи библиотек:

```
<?xml version="1.0" ?>
<root>
    <InfTag>-inf</InfTag>
    <expTag>1e-09</expTag>
    <key name="-InfTag">-inf</key>
    <Na-N>nan</Na-N>
    <nums>
        <item>1</item>
        <item>-1</item>
        <item>1315600000.0</item>
        <item>
            <item>something</item>
            <item>True</item>
        </item>
        <item>!&amp;@#&amp;@)!sdf&gt; &lt;dpoiu</item>
        <item>
            <dv_s>
                <ins>nan</ins>
                <null/>
```

```
            </dv_s>
        </item>
    </nums>
</root>
```

Как можно видеть, результаты работы практически полностью идентичны, и отличия заключаются только в незначительных особенностях обработки спецсимволов, которые являются особенностью библиотеки.

Дополнительное задание 4

Для сравнения стократного времени выполнения программ я написал следующий код на языке python:

```
from task0 import JSON2XML0
from task1 import JSON2XML1
from task2 import JSON2XML2
from task3 import JSON2XML3
from time import time

def getTime(inputFile, outputFile, f):
    startTime = time()
    for i in range(100):
        f(inputFile, outputFile)
    endTime = time()
    return endTime - startTime

def main():
    funcs = [JSON2XML0, JSON2XML1, JSON2XML2, JSON2XML3]
    for i in range(len(funcs)):
        fileName = f'day{i}.xml'
        print(f'Время стократного выполнения для task{i}:
{round(getTime("day.json", fileName, funcs[i]), 9)} сек')
if __name__ == '__main__':
    main()
```

В результате работы программы имеем следующий вывод:

```
Время стократного выполнения для task0: 0.038094521 сек

Время стократного выполнения для task1: 0.312334299 сек

Время стократного выполнения для task2: 0.078078508 сек

Время стократного выполнения для task3: 0.072746992 сек
```

Программа, использующая библиотеки работала дольше всего так как она использует множество сторонних функций и структур данных, которые выполняются дольше

элементарных операций. Также стоит учитывать время на передачу данных между модулями библиотек и их подключение.

Программа, использующая простую замену метасимволов, отработала быстрее всего, так как она имеет линейную сложность и выполняет только базовые операции со строками.

Программы, использующие регулярные выражение и рекурсивный разбор формальной грамматики имеют примерно одинаково время выполнения так как их сложность в среднем линейная и они обе используют комбинацию итеративного прохода по строке и регулярных выражений, которые и увеличивают время работы.

Мною также был построен график(Рисунок 2) зависимости времени выполнения от количества итераций для 1000 итераций, на котором наглядно видно различие между скоростью алгоритмов

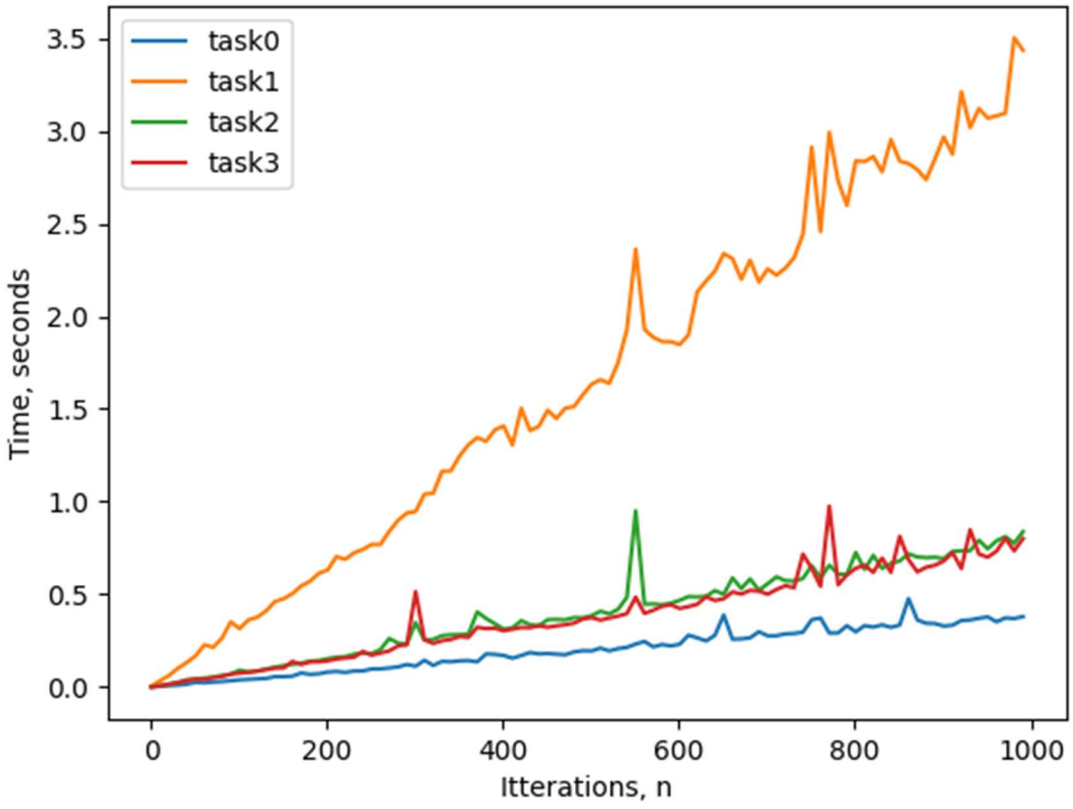

Рисунок 2

Дополнительно задание 5

Для перевода файла из формата JSON в формат CSV я использовал библиотеки pandas и json.

Листинг программы приведён ниже:

```
import pandas as pd
import json
def main():
    with open('day.json', mode='r', encoding='utf-8') as f:
        json_data = json.load(f)
    df = pd.json_normalize(json_data)
    df.to_csv('day5.csv', encoding='utf8', index=False)

if __name__ == '__main__':
    main()
```

В результате работы программы получается csv файл из 2 строк, в котором в первой строке записан иерархический путь до базового элемента json файла, а во второй строке записано его содержимое.

Заключение

В ходе работы я познакомился с различными форматами разметки, реализовал на языке python парсинг и конвертацию файла из формата JSON в формат XML различными способами. Научился использовать ряд библиотек для работы с файлами JSON, XML и CSV. Познакомился с формальными грамматиками и реализовал их использование на языке python. В ходе работы я закрепил навыки использования регулярных выражений, а также чтения и записи файлов в языке python. Также я сравнил время работы написанных программ и научился использовать модуль time.

Список использованных источников

1. Формальные грамматики и языки. Элементы теории трансляции: Учебное пособие для студентов II курса (издание третье, переработанное и дополненное). / И. А. Волкова, А. А. Вылиток, Т. В. Руденко — М.: Издательский отдел факультета ВМиК МГУ им. М.В.Ломоносова, 2009 — 115 с.
2. Лекция 10 «Формальные грамматики»: конспект лекции для студентов Механико-Математического факультета МГУ имени М. В. Ломоносова [Электронный ресурс] – Режим доступа: http://mech.math.msu.su/~vvb/2course/lect10.html