

Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**Отчёт по лабораторной работе 4.**

Тема: Исследование протоколов,

форматов обмена информацией и языков разметки

документов.

Вариант: 30

Выполнил**:** студент группы Р3115 Суворов Станислав Денисович

Проверила**:** Белокон Юлия Алексеевна

Дата сдачи: 17.09.2024

**2024**

[Задание 3](#_Toc129651014)

[Основные этапы вычисления 5](#_Toc129651015)

[Дополнительное задание №1 9](#_Toc129651016)

[Дополнительное задание №2 12](#_Toc129651017)

[Дополнительное задание №3 17](#_Toc129651018)

[Дополнительное задание №4 23](#_Toc129651019)

[Дополнительное задание №5 29](#_Toc129651019)

[Заключение 30](#_Toc129651020)

[Список использованной литературы 31](#_Toc129651021)

# Задание

**Составить файл с расписанием занятий своей учебной группы в указанный день в исходном формате, написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый**.

День недели: Понедельник

Исходный формат: JSON

Результирующий формат: XML

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

**Дополнительное задание №1:**

1. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.

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

3. Сравнить полученные результаты и объяснить их сходство/различие.

**Дополнительное задание №2:**

1. Переписать исходный код, добавив в него использование регулярных выражений.

2. Сравнить полученные результаты и объяснить их сходство/различие.

**Дополнительное задание №3:**

1. 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.

2. Проверку осуществить как минимум для расписания с

двумя учебными днями по два занятия в каждом.

3. Сравнить полученные результаты и объяснить их

сходство/различие. Объяснение должно быть отражено в

отчёте.

**Дополнительное задание №4:**

1. 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.

2. Проанализировать полученные результаты и объяснить их сходство/различие.

**Дополнительное задание №5:**

1. 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.

2. Проанализировать полученные результаты, объяснить особенности использования формата.

# Основные этапы вычисления

Сначала был создан JSON-файл, содержащий расписание предметов на среду:

{"Понедельник":[{"period":{"time":"11:40-13:10","even-week":"true"},"place":{"room":"1330","building":"Кронверкский пр., д.49, лит.А"},"class-info":{"lesson":"Информатика","teacher":"Белокон Юлия Алексеевна"},"lesson-format":"Очно"},{"period":{"time":"13:30-15:00","even-week":"true"},"place":{"room":"1330","building":"Кронверкский пр., д.49, лит.А"},"class-info":{"lesson":"Информатика","teacher":"Белокон Юлия Алексеевна"},"lesson-format":"Очно"}]}

Конвертация данных из формата JSON в формат XML будет выполнена с помощью следующих шагов:

1. Чтение JSON-файла и сохранение его в виде «сырого» текста

2. Парсинг этого текста и его интерпретация с помощью объектов языка Python.

3. Сериализация полученной совокупности объектов в структуру, соответствующую XML-файлу.

4. Создание нового XML-файла и запись в него.

def DictToXML(d):  
 global xml  
 global counter  
 global maxcounter  
 for k, v in d.items():  
 xml += "\t" \* (counter + 1) + "<" + k + ">"  
 if isinstance(v, list):  
 for i in v:  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(i)  
 counter -= 1  
 elif isinstance(v, dict):  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(v)  
 counter -= 1  
 else:  
 maxcounter = counter  
 xml += str(v)  
 if counter == maxcounter:  
 xml += "</" + k + ">\n"  
 else:  
 xml += "\t" \* (counter + 1) + "</" + k + ">\n"  
  
  
maxcounter = 0  
counter = 0  
xml = '<?xml version="1.0" encoding="UTF-8" ?>\n<main>\n'  
with open("mon\_shedule.json", "r", encoding="utf-8") as f:  
 text = f.read()  
true = True  
New\_text = eval(text)  
DictToXML(New\_text)  
xml += "</main>"  
print(xml)  
with open("../Outputs/output1.xml", "w", encoding="utf-8") as f:  
 f.write(xml)

После запуска программа создаст файл output.xml в своей директории. Его содержимое:

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

<main>

<mon\_schedule>

<even\_week>True</even\_week>

<num\_of\_group>P3115</num\_of\_group>

<day>Monday</day>

<lessons>

<lesson1>

<title>Основы профессиональной деятельности</title>

<time>8:20-9:50</time>

<teacher>Блохина Елена Николаевна</teacher>

<room>1330</room>

<address>Кронверский пр., д.49, лит.А</address>

</lesson1>

<lesson2>

<title>Основы профессиональной деятельности</title>

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

<teacher>Блохина Елена Николаевна</teacher>

<room>1330</room>

<address>Кронверский пр., д.49, лит.А</address>

</lesson2>

<lesson3>

<title>Информатика</title>

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

<teacher>Белокон Юлия Алексеевна</teacher>

<room>1330</room>

<address>Кронверский пр., д.49, лит.А</address>

</lesson3>

<lesson4>

<title>Информатика</title>

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

<teacher>Белокон Юлия Алексеевна</teacher>

<room>1330</room>

<address>Кронверский пр., д.49, лит.А</address>

</lesson4>

</lessons>

</mon\_schedule>

</main>

## Дополнительное задание №1

Для сериализации данных в XML-файл можно использовать библиотеку json2xml. Её требуется дополнительно установить следующим образом:

pip install json2xml

from json2xml import json2xml  
from json2xml.utils import readfromjson  
  
json = readfromjson("sat\_shedule.json")  
xml = json2xml.Json2xml(json).to\_xml()  
print(xml)  
with open("../Outputs/output2.xml", "w", encoding="utf-8") as f:  
 f.write(xml)

Результат конвертации доступен в файле second\_task/output2.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<all>  
 <sat\_schedule type="dict">  
 <even\_week type="bool">true</even\_week>  
 <group type="str">P3115</group>  
 <day type="str">Saturday</day>  
 <lessons type="dict">  
 <lesson1 type="dict">  
 <title type="str">Математический анализ</title>  
 <time type="str">10:00-11:30</time>  
 <teacher type="str">Блейхер Оксана Александровна</teacher>  
 <format type="str">Дистанционный ZOOM</format>  
 </lesson1>  
 <lesson2 type="dict">  
 <title type="str">Программирование</title>  
 <time type="str">13:30-15:00</time>  
 <teacher type="str">Кулинич Ярослав Вадимович</teacher>  
 <room type="int">1338</room>  
 <address type="str">Кронверский пр., д.49, лит.А</address>  
 </lesson2>  
 <lesson3 type="dict">  
 <title type="str">Программирование</title>  
 <time type="str">15:20-16:50</time>  
 <teacher type="str">Кулинич Ярослав Вадимович</teacher>  
 <room type="int">1338</room>  
 <address type="str">Кронверский пр., д.49, лит.А</address>  
 </lesson3>  
 </lessons>  
 </sat\_schedule>  
</all>

Отличие этого результата от результата программы, написанной вручную, состоит в следующем:

1. Тег main заменен на all.

2. У объектов указывается их тип.

3. Добавлен тег <item> для разных блоков.

Проанализировав различия, видно, что выводы обеих программ отличаются в некоторых стилистических моментах, которые не влияют на корректность их результатов, а также в новом теге, который может облегчить дальнейшее разделение блоков.

## Дополнительное задание №2

Напишем новый файл third\_task.py, в котором для определения типов данных будут использоваться регулярные выражения.

import re  
  
  
def JsonToDict(d):  
 for k, v in d.items():  
 array = []  
 nd = {}  
 regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"  
 for i in re.finditer(regex, v):  
 if i.group(1) in nd.keys():  
 array.append(JsonToDictPart2(nd))  
 nd = {}  
 if i.group(2) == None:  
 nd[i.group(1)] = i.group(3)  
 else:  
 nd[i.group(1)] = i.group(2)  
 array.append(JsonToDictPart2(nd))  
 d[k] = array  
 return d  
  
  
def JsonToDictPart2(d):  
 for k, v in d.items():  
 a = re.search("\":", v)  
 if re.search("\":", v):  
 nd = {}  
 regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"  
 for i in re.finditer(regex, v):  
 if i.group(2) == None:  
 ch = str(i.group(3)).replace('"', "")  
 nd[i.group(1)] = ch  
 else:  
 ch = str(i.group(2)).replace('"', "")  
 nd[i.group(1)] = ch  
 d[k] = nd  
 else:  
 v = v.replace('"', "")  
 d[k] = v  
 return d  
  
  
def DictToXML(d):  
 global xml  
 global counter  
 global maxcounter  
 for k, v in d.items():  
 xml += "\t" \* (counter + 1) + "<" + k + ">"  
 if isinstance(v, list):  
 for i in v:  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(i)  
 counter -= 1  
 elif isinstance(v, dict):  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(v)  
 counter -= 1  
 else:  
 maxcounter = counter  
 xml += str(v)  
 if counter == maxcounter:  
 xml += "</" + k + ">\n"  
 else:  
 xml += "\t" \* (counter + 1) + "</" + k + ">\n"  
  
  
maxcounter = 0  
counter = 0  
xml = '<?xml version="1.0" encoding="UTF-8" ?>\n<main>\n'  
with open("mon\_shedule.json", "r", encoding="utf-8") as f:  
 text = f.read()  
text = text[1:-1]  
d = {}  
regex = r'(?<=")[А-Яа-яёЁ\w]+(?=":\[)'  
km = re.findall(regex, text)  
regex = r'(?<=:\[{)[^\]]+(?=})'  
vm = re.findall(regex, text)  
for i in range(len(km)):  
 d[km[i]] = vm[i]  
a = JsonToDict(d)  
DictToXML(a)  
xml += "</main>"  
print(xml)  
with open("../Outputs/output3.xml", "w", encoding="utf-8") as f:  
 f.write(xml)

Результат сохранён в файл output3.xml.

<?xml version="1.0" encoding="UTF-8" ?>  
<main>  
 <Понедельник>  
 <period>  
 <time>11:40-13:10</time>  
 <even-week>true</even-week>  
 </period>  
 <place>  
 <room>1330</room>  
 <building>Кронверкский пр., д.49, лит.А</building>  
 </place>  
 <class-info>  
 <lesson>Информатика</lesson>  
 <teacher>Белокон Юлия Алексеевна</teacher>  
 </class-info>  
 <lesson-format>Очно</lesson-format>  
 <period>  
 <time>13:30-15:00</time>  
 <even-week>true</even-week>  
 </period>  
 <place>  
 <room>1330</room>  
 <building>Кронверкский пр., д.49, лит.А</building>  
 </place>  
 <class-info>  
 <lesson>Информатика</lesson>  
 <teacher>Белокон Юлия Алексеевна</teacher>  
 </class-info>  
 <lesson-format>Очно</lesson-format>  
 </Понедельник>  
</main>

## Дополнительное задание №3

1. 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.

2. Проверку осуществить как минимум для расписания с двумя учебными днями по два занятия в каждом.

3. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

from lark import Lark, Transformer  
  
# Грамматика JSON  
json\_grammar = r"""  
 start: value  
  
 value: object  
 | array  
 | string  
 | number  
 | "true" -> true  
 | "false" -> false  
 | "null" -> null  
  
 object : "{" [members] "}"  
 members: pair ("," pair)\*  
 pair : string ":" value  
  
 array : "[" [elements] "]"  
 elements: value ("," value)\*  
  
 string : ESCAPED\_STRING  
  
 number : SIGNED\_NUMBER  
  
 %import common.ESCAPED\_STRING  
 %import common.SIGNED\_NUMBER  
 %import common.WS\_INLINE  
  
 %ignore WS\_INLINE  
"""  
  
  
class JsonToXmlTransformer(Transformer):  
 def \_\_init\_\_(self):  
 self.counter = 0  
 self.max\_counter = 0  
 self.xml = ""  
  
 def start(self, items):  
 return self.xml  
  
 def object(self, items):  
 self.counter += 1  
 self.max\_counter = max(self.counter, self.max\_counter)  
 self.xml += "\t" \* self.counter + "<object>\n"  
 for item in items:  
 # Преобразуем Tree в строку  
 item\_str = ''.join(str(child) for child in item.children)  
 self.xml += item\_str  
 self.counter -= 1  
 self.xml += "\t" \* self.counter + "</object>\n"  
 return self.xml  
  
 def pair(self, items):  
 key, value = items  
 # Преобразуем Tree в строку  
 value\_str = ''.join(str(child) for child in value.children)  
 self.xml += "\t" \* (self.counter + 1) + "<" + key + ">" + value\_str + "</" + key + ">\n"  
 return self.xml  
  
 def array(self, items):  
 self.counter += 1  
 self.max\_counter = max(self.counter, self.max\_counter)  
 self.xml += "\t" \* self.counter + "<array>\n"  
 for item in items:  
 # Преобразуем Tree в строку  
 item\_str = ''.join(str(child) for child in item.children)  
 self.xml += item\_str  
 self.counter -= 1  
 self.xml += "\t" \* self.counter + "</array>\n"  
 return self.xml  
  
 def string(self, items):  
 return items[0]  
  
 def number(self, items):  
 return items[0]  
  
 def true(self, \_):  
 return "true"  
  
 def false(self, \_):  
 return "false"  
  
 def null(self, \_):  
 return "null"  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Чтение файла JSON  
 with open("mon\_shedule.json", "r", encoding="utf-8") as f:  
 json\_data = f.read()  
  
 # Парсинг JSON  
 parser = Lark(json\_grammar, parser='lalr', transformer=JsonToXmlTransformer())  
 tree = parser.parse(json\_data)  
  
 # Генерация XML  
 xml = '<?xml version="1.0" encoding="UTF-8"?>\n' + tree + '\n</main>'  
  
 # Запись результата в файл  
 with open("../Outputs/output4.xml", "w", encoding="utf-8") as f:  
 f.write(xml)

<?xml version="1.0" encoding="UTF-8" ?>  
<main>  
 <mon\_schedule>  
 <even\_week>True</even\_week>  
 <num\_of\_group>P3115</num\_of\_group>  
 <day>Monday</day>  
 <lessons>  
 <lesson1>  
 <title>Основы профессиональной деятельности</title>  
 <time>8:20-9:50</time>  
 <teacher>Блохина Елена Николаевна</teacher>  
 <room>1330</room>  
 <address>Кронверский пр., д.49, лит.А</address>  
 </lesson1>  
 <lesson2>  
 <title>Основы профессиональной деятельности</title>  
 <time>10:00-11:30</time>  
 <teacher>Блохина Елена Николаевна</teacher>  
 <room>1330</room>  
 <address>Кронверский пр., д.49, лит.А</address>  
 </lesson2>  
 <lesson3>  
 <title>Информатика</title>  
 <time>11:40-13:10</time>  
 <teacher>Белокон Юлия Алексеевна</teacher>  
 <room>1330</room>  
 <address>Кронверский пр., д.49, лит.А</address>  
 </lesson3>  
 <lesson4>  
 <title>Информатика</title>  
 <time>13:30-15:00</time>  
 <teacher>Белокон Юлия Алексеевна</teacher>  
 <room>1330</room>  
 <address>Кронверский пр., д.49, лит.А</address>  
 </lesson4>  
 </lessons>  
 </mon\_schedule>  
</main>

Этот код определяет грамматику JSON и использует её для разбора данных из файла mon\_schedule.json. Затем он генерирует соответствующий XML-код и записывает его в файл output3.xml.

## Дополнительное задание №4

Для замера времени многократного запуска функций конвертации будем использовать функцию time из встроенной библиотеки time.

from time import time  
from json2xml import json2xml  
from json2xml.utils import readfromjson  
import re  
  
st\_time1 = time()  
  
  
#№1  
def DictToXML(d):  
 global xml  
 global counter  
 global maxcounter  
 for k, v in d.items():  
 xml += " " \* counter + "<" + k + ">"  
 if isinstance(v, list):  
 for i in v:  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(i)  
 elif isinstance(v, dict):  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(v)  
 else:  
 maxcounter = counter  
 xml += str(v)  
 if counter == maxcounter:  
 xml += "</" + k + ">\n"  
 else:  
 xml += " " \* counter + "</" + k + ">\n"  
 counter -= 1  
 if counter == 0:  
 maxcounter = 0  
  
  
maxcounter = 0  
counter = 0  
xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'  
with open("mon\_shedule.json", "r") as f:  
 text = f.read()  
true = True  
false = False  
null = None  
l = eval(text)  
DictToXML(l)  
#print(xml)  
t1 = (time() - st\_time1) \* (10)  
print("--- №1 %s секунд \* 10 ---" % (t1))  
st\_time2 = time()  
  
#№2  
data = readfromjson("mon\_shedule.json")  
xml = json2xml.Json2xml(data).to\_xml()  
# print(xml)  
t2 = (time() - st\_time2) \* (10)  
print("--- №2 %s секунд \* 10 ---" % (t2))  
st\_time3 = time()  
  
  
#№3  
def JsonToDict(d):  
 for k, v in d.items():  
 array = []  
 nd = {}  
 regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"  
 for i in re.finditer(regex, v):  
 if i.group(1) in nd.keys():  
 array.append(JsonToDictPart2(nd))  
 nd = {}  
 if i.group(2) is None:  
 nd[i.group(1)] = i.group(3)  
 else:  
 nd[i.group(1)] = i.group(2)  
 array.append(JsonToDictPart2(nd))  
 d[k] = array  
 return d  
  
  
def JsonToDictPart2(d):  
 for k, v in d.items():  
 if re.search("\":", v):  
 nd = {}  
 regex = r"(?<=\")([\w|-]+)\"(?::)(?:(\".+?\")|\{(\".+?\")\})(?=[, ]?)"  
 for i in re.finditer(regex, v):  
 if i.group(2) is None:  
 ch = str(i.group(3)).replace('"', "")  
 nd[i.group(1)] = ch  
 else:  
 ch = str(i.group(2)).replace('"', "")  
 nd[i.group(1)] = ch  
 d[k] = nd  
 else:  
 v = v.replace('"', "")  
 d[k] = v  
 return d  
  
  
def DictToXML(d):  
 global xml  
 global counter  
 global maxcounter  
 for k, v in d.items():  
 xml += " " \* counter + "<" + k + ">"  
 if isinstance(v, list):  
 for i in v:  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(i)  
 elif isinstance(v, dict):  
 if xml[-1] != "\n":  
 xml += "\n"  
 counter += 1  
 maxcounter = counter  
 DictToXML(v)  
 else:  
 maxcounter = counter  
 xml += str(v)  
 if counter == maxcounter:  
 xml += "</" + k + ">\n"  
 else:  
 xml += " " \* counter + "</" + k + ">\n"  
 counter -= 1  
 if counter == 0:  
 maxcounter = 0  
  
  
maxcounter = 0  
counter = 0  
xml = '<?xml version="1.0" encoding="UTF-8" ?>\n'  
with open("mon\_shedule.json", "r") as f:  
 text = f.read()  
text = text[1:-1]  
d = {}  
regex = r'(?<=")[А-Яа-яёЁ\w]+(?=":\[)'  
km = re.findall(regex, text)  
regex = r'(?<=:\[{)[^\]]+(?=})'  
vm = re.findall(regex, text)  
for i in range(len(km)):  
 d[km[i]] = vm[i]  
a = JsonToDict(d)  
DictToXML(a)  
#print(xml)  
t3 = (time() - st\_time3) \* 10  
print("--- №3 %s секунд \* 10 ---" % t3)

Вывод программы представлен на рисунке 1.

A black background with white numbers

Description automatically generated

Рисунок 1 – Вывод программы

Выходит, что самая быстрая реализация – конвертация вручную без регулярных выражений. Можно предположить, что рекурсивный алгоритм, который в ней реализован, оптимален для небольшого файла, а также не требует дополнительных обращений к библиотекам, таким как re.

## Дополнительное задание №5

Будем конвертировать исходный JSON-файл в формат Markdown. Он позволяет создавать структурированные документы с использованием минимального количества символов и форматирования текста с помощью некоторых символов. Markdown применяется для создания простых веб-страниц, блогов, документации и прочего.

def DictToMarkdown(d):  
 global md  
 global rekdeep  
 for k, v in d.items():  
 md += ("#" \* rekdeep) + " " + k + "\n"  
 if isinstance(v, list):  
 for i in v:  
 md += "\n------\n"  
 rekdeep += 1  
 DictToMarkdown(i)  
 rekdeep -= 1  
 elif isinstance(v, dict):  
 rekdeep += 1  
 DictToMarkdown(v)  
 rekdeep -= 1  
 else:  
 md += v + "\n"  
  
  
rekdeep = 1  
md = ''  
with open("mon\_shedule.json", "r", encoding="utf-8") as f:  
 text = f.read()  
true = True  
false = False  
null = None  
l = eval(text)  
DictToMarkdown(l)  
print(md)  
  
with open("../Outputs/markdown.md", "w", encoding="utf-8") as f:  
 f.write(md)

Листинг кода именно такой, если скопировать и вставить в IDE все будет klass.

После выполнения блока с сериализацией считанных данных из JSON-файла в директории появится файл markdown.md. Его содержимое:

# Понедельник  
  
------  
## period  
### time  
11:40-13:10  
### even-week  
true  
## place  
### room  
1330  
### building  
Кронверкский пр., д.49, лит.А  
## class-info  
### lesson  
Информатика  
### teacher  
Белокон Юлия Алексеевна  
## lesson-format  
Очно  
  
------  
## period  
### time  
13:30-15:00  
### even-week  
true  
## place  
### room  
1330  
### building  
Кронверкский пр., д.49, лит.А  
## class-info  
### lesson  
Информатика  
### teacher  
Белокон Юлия Алексеевна  
## lesson-format  
Очно

# 

# Заключение

В результате выполнения данной лабораторной работы были изучены следующие вещи:

* Форматы JSON, XML и Markdown;
* Понятие парсинга данных и его практическая реализация для считывания JSON-файла;
* Понятие сериализации данных и её практическая реализация для записи данных в XML-файл;
* Библиотеки для работы с JSON- и XML-файлами;
* Применение регулярных выражений при парсинге файла.

# Список использованной литературы

1. Балакшин Е.А., Соснин П.В., Машина В.В. Информатика. –   
СПб: Университет ИТМО, 2020.

2. Лямин А. В., Череповская Е. Н. Объектно-ориентированное программирование. Компьютерный практикум. – СПб: Университет ИТМО, 2017.