Федеральное государственное автономное образовательное учреждение

высшего образования «Национальный исследовательский университет

ИТМО»

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №4**

**Исследование протоколов, форматов обмена информацией и языков разметки документов**

**Вариант 21**

Выполнила: Леонтьева Арина Николаевна Р3113

Проверила: к.п.н., доцент Авксентьева Елена Юрьевна

Санкт-Петербург 2024

**Содержание**

[Задание 2](#_Toc179236708)

[Основные этапы вычисления 3](#_Toc179236709)

[Задание 1 3](#_Toc179236710)

[Задание 2 4](#_Toc179236711)

[Задание 3 5](#_Toc179236712)

[Задание 4 6](#_Toc179236713)

[Задание 5 7](#_Toc179236714)

[Задание 6 8](#_Toc179236715)

[Задание 7 9](#_Toc179236716)

[Заключение 10](#_Toc179236717)

[Список использованных источников 11](#_Toc179236718)

## Задание

1. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного. При этом необходимо, чтобы хотя бы в одной из выбранных дней 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.
2. Обязательное задание (позволяет набрать до 45 процентов от максимального числа баллов БаРС за данную лабораторную): написать программу на языке Python 3.x или любом другом, которая бы осуществляла парсинг и конвертацию исходного файла в новый путём простой замены метасимволов исходного формата на метасимволы результирующего форматa. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.

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

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

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

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

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

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

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

5. Дополнительное задание №3 (позволяет набрать +25 процентов от максимального числа баллов БаРС за данную лабораторную).

- 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.

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

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

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

- Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.

- Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

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

- 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. Объяснение должно быть отражено в отчёте.

## Основные этапы вычисления

## Задание 1

week\_days:

tuesday:

- lesson:

subject: "Lineal Algebra"

type: "Practice"

classroom: "2430"

time: "8:20-9:50"

teacher: "Rvanova Alla"

building: "Kronva"

- lesson:

subject: "Linear Algebra"

type: "Lecture"

classroom: "2426"

time: "10:00-11:30"

teacher: "Rvanova Alla"

building: "Kronva"

thursday:

- lesson:

subject: "Russia in the history of international relations"

type: "Practice"

classroom: "2426"

time: "13:30-15:00"

teacher: "Pavlovsky Alexey"

building: "Kronva"

- lesson:

subject: "Russia in the history of international relations"

type: "Lecture"

classroom: "Orange Classroom (1229)"

time: "17:00-18:30"

teacher: "Bogomazov Nikolai"

building: "Kronva"

## Задание 2

def putValue(dictionary, path, value):

if len(path) != 1:

if "- " in path[0]:

return putValue(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::], value)

else:

return putValue(dictionary[path[0]], path[1::], value)

else:

dictionary[path[0]] = value

def indentDict(dictionary, path):

if len(path) != 1:

if "- " in path[0]:

return indentDict(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::])

else:

return indentDict(dictionary[path[0]], path[1::])

else:

dictionary[path[0]] = dict()

def addList(dictionary, path, key):

if len(path) != 1:

return addList(dictionary[path[0]], path[1::], key)

else:

if not (dictionary[path[0]]):

dictionary[path[0]] = [dict()]

if (int(key[-1]) == 0):

dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

else:

dictionary[path[0]].append(dict())

dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

def yamlToDict (filename):

result = dict()

with open(filename, 'r', encoding='utf-8') as file:

lines = file.readlines()

path = []

last\_level = 0

listIndex = 0

inList = False

for line in lines:

if (line.strip() == ""):

continue

line = line.rstrip()

level = len(line) - len(line.lstrip())

line = line.lstrip()

line = line.split(":", 1)

key = line[0]

value = ""

if (len(line) > 1):

value = line[1]

value = value.strip().replace('"', '')

if last\_level > level:

for i in range((last\_level-level) // 2):

path.pop()

if (last\_level - level > 2):

listIndex = 0

path.append(key)

if value:

putValue(result, path, value)

path.pop()

else:

if "-" in key:

key = key.lstrip("- ")

inList = True

key = key + str(listIndex)

path.pop()

if (listIndex == 0):

addList(result, path, key)

else:

addList(result, path, key)

path.append("- " + key)

listIndex += 1

else:

indentDict(result, path)

if (inList):

last\_level = level - 2

else:

last\_level = level

return result

filw = open("output.json", "w")

print(yamlToDict("orig.yaml"))

def DictToJSON(dictionary, level=0):

json = "{\n"

for key, value in dictionary.items():

if isinstance(value, dict):

json += level \* ' ' + '"' + key + '":' + DictToJSON(value, level + 1) + ",\n"

elif isinstance(value, list):

json += level \* ' ' + '"' + key + '":'+ ListToJSON(value, level + 1) + ",\n"

else:

json += level \* ' ' + '"' + key + '": "'+ str(value) + '",\n'

json = json.rstrip(",\n") + "\n"+ level\*' '+"}"

return json

def ListToJSON(list,level):

json = "["

for item in list:

if isinstance(item, dict):

json += DictToJSON(item, level + 1) + ",\n"

elif isinstance(item, list):

json += ListToJSON(item, level + 1) + ",\n"

else:

json += str(item) + ",\n"

json = json.rstrip(",\n") + "]"

return json

filw.write(DictToJSON(yamlToDict("orig.yaml")))

## Задание 3

import yaml

import json

with open('orig.yaml', 'r', encoding='utf-8') as yaml\_file:

yaml\_data = yaml.safe\_load(yaml\_file)

json\_data = json.dumps(yaml\_data, indent=4, ensure\_ascii=False)

with open('output.json', 'w', encoding='utf-8') as json\_file:

json\_file.write(json\_data)

## Задание 4

## import re

## def putValue(dictionary, path, value):

## if len(path) != 1:

## 

## if "- " in path[0]:

## return putValue(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::], value)

## else:

## return putValue(dictionary[path[0]], path[1::], value)

## else:

## dictionary[path[0]] = value

## def indentDict(dictionary, path):

## if len(path) != 1:

## 

## if "- " in path[0]:

## return indentDict(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::])

## else:

## return indentDict(dictionary[path[0]], path[1::])

## else:

## dictionary[path[0]] = dict()

## def addList(dictionary, path, key):

## if len(path) != 1:

## return addList(dictionary[path[0]], path[1::], key)

## else:

## if not (dictionary[path[0]]):

## dictionary[path[0]] = [dict()]

## if (int(key[-1]) == 0):

## dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

## else:

## dictionary[path[0]].append(dict())

## dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

## def yamlToDict(filename):

## result = dict()

## with open(filename, 'r', encoding='utf-8') as file:

## lines = file.readlines()

## path = []

## last\_level = 0

## listIndex = 0

## inList = False

## for line in lines:

## if (line.strip() == ""):

## continue

## level = len(re.match(r'^\s\*', line).group())

## line = line.strip()

## match = re.match(r'^(.\*?):(.\*)$', line)

## if match:

## key = match.group(1).strip()

## value = match.group(2).strip().replace('"', '')

## else:

## key = line

## value = ""

## if last\_level > level:

## for i in range((last\_level - level) // 2):

## path.pop()

## if (last\_level - level > 2):

## listIndex = 0

## path.append(key)

## if value:

## putValue(result, path, value)

## path.pop()

## else:

## if re.match(r'^- ', key):

## key = key.lstrip("- ") + str(listIndex)

## path.pop()

## addList(result, path, key)

## path.append("- " + key)

## listIndex += 1

## inList = True

## else:

## indentDict(result, path)

## last\_level = level - 2 if inList else level

## return result

## filw = open("output.json", "w")

## print(yamlToDict("orig.yaml"))

## def DictToJSON(dictionary, level=0):

## json = "{\n"

## for key, value in dictionary.items():

## if isinstance(value, dict):

## json += level \* ' ' + '"' + key + '":' + DictToJSON(value, level + 1) + ",\n"

## elif isinstance(value, list):

## json += level \* ' ' + '"' + key + '":'+ ListToJSON(value, level + 1) + ",\n"

## else:

## json += level \* ' ' + '"' + key + '": "'+ str(value) + '",\n'

## json = json.rstrip(",\n") + "\n"+ level\*' '+"}"

## return json

## def ListToJSON(list,level):

## json = "["

## for item in list:

## if isinstance(item, dict):

## json += DictToJSON(item, level + 1) + ",\n"

## elif isinstance(item, list):

## json += ListToJSON(item, level + 1) + ",\n"

## else:

## json += str(item) + ",\n"

## json = json.rstrip(",\n") + "]"

## return json

## filw.write(DictToJSON(yamlToDict("orig.yaml")))

## Задание 5

import re

def parsingFG():

text=open("orig.yaml", encoding="utf-8").read()

ar=text.splitlines()

ar.append(" ")

steck=[]

ktire=0

flug = False

for i in range(len(ar)-1):

if not re.search(r": ", ar[i]):

ar[i]= ar[i] + (" ")

dlin=(len(ar[i]) - len(ar[i].lstrip()))

key, value = ar[i].lstrip().split(': ', 1)

ar[i] = ' ' \* dlin + f'"{key}": {value}'

ar[i] = ar[i].replace('"- ','- "')

if re.search(r"-", ar[i]):

dlin1 = (len(ar[i]) - len(ar[i].lstrip()))

if i<len(ar)-1:

if re.search(r": ", ar[i]) and re.search(r"-", ar[i+1]):

ar[i]=ar[i].replace(': ',': [')

for i in range(len(ar)-1):

if (len(ar[i]) - len(ar[i].lstrip()))<(len(ar[i+1]) - len(ar[i+1].lstrip())):

if re.search(r"\[", ar[i]):

fg=(len(ar[i+1]) - len(ar[i+1].lstrip()))

else:

ar[i]+="{"

fg=(len(ar[i]) - len(ar[i].lstrip()))

if (len(ar[i]) - len(ar[i].lstrip()))>(len(ar[i+1]) - len(ar[i+1].lstrip())):

ar[i]+="\n "+ " "\* fg+"}"

if (len(ar[i]) - len(ar[i].lstrip()))==0:

ar[i-1]=ar[i-1]+"\n ]"

for i in range(len(ar)):

if re.search(r"\[", ar[i]):

flug = True

if re.search(r"]", ar[i]):

flug = False

if flug:

ar[i + 1] += ","

if ": {" in ar[i]:

ar[i]=ar[i].replace(",","")

for i in range(len(ar)):

if re.search(r"\[", ar[i]):

flug = True

if re.search(r"]", ar[i]):

flug = False

ktire=0

if flug and re.search(r"-", ar[i]):

ktire += 1

if re.search(r"-", ar[i]):

if ktire==1:

ar[i]=ar[i].replace("-"," ").replace("{","")

ar[i - 1] += "\n {"

elif ktire>1:

ar[i] = ar[i].replace("-", " ").replace("{", "")

ar[i-1]=ar[i-1].replace(",","")

ar[i-1]+="\n },\n {"

if "},\n" in ar[i] and re.search(r" \{ |' ' ", ar[i+1]):

ar[i]=ar[i].replace("},","}\n },")

for i in range(len(ar)-1):

if "[" in ar[i]:

fg1=(len(ar[i+1]) - len(ar[i+1].lstrip()))

for i in range(len(ar)):

if re.search(r"]", ar[i]):

ar[i]=ar[i].replace("],", " }\n],")

for i in range(len(ar)-1,0,-1):

if re.search(r" \{ | ] ", ar[i]):

ar[i]=ar[i].replace(",","")

else:

break

st=""

ar.append("}")

ar.insert(0,"{")

for i in range(len(ar)):

if i==0 or i==len(ar)-1:

print(ar[i])

else:

ar[i]=" "+ar[i]

print(ar[i])

output\_file = "output.json"

with open(output\_file, 'w', encoding='utf-8') as file:

for line in ar:

file.write(line + "\n")

st+=line + "\n"

print(f"Данные успешно записаны в файл {(output\_file)}")

try:

dict(st)

except Exception as e:

print(e)

parsingFG()

## Задание 6

import time

import yaml

import json

def custom\_parser(yaml\_data):

def putValue(dictionary, path, value):

if len(path) != 1:

if "- " in path[0]:

return putValue(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::], value)

else:

return putValue(dictionary[path[0]], path[1::], value)

else:

dictionary[path[0]] = value

def indentDict(dictionary, path):

if len(path) != 1:

if "- " in path[0]:

return indentDict(dictionary[int(path[0][-1])][path[0][2:-1]], path[1::])

else:

return indentDict(dictionary[path[0]], path[1::])

else:

dictionary[path[0]] = dict()

def addList(dictionary, path, key):

if len(path) != 1:

return addList(dictionary[path[0]], path[1::], key)

else:

if not (dictionary[path[0]]):

dictionary[path[0]] = [dict()]

if (int(key[-1]) == 0):

dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

else:

dictionary[path[0]].append(dict())

dictionary[path[0]][int(key[-1])][key[0:-1]] = dict()

def yamlToDict(yaml\_data):

result = dict()

lines = yaml\_data.splitlines()

path = []

last\_level = 0

listIndex = 0

inList = False

for line in lines:

if (line.strip() == ""):

continue

line = line.rstrip()

level = len(line) - len(line.lstrip())

line = line.lstrip()

line = line.split(":", 1)

key = line[0]

value = ""

if (len(line) > 1):

value = line[1]

value = value.strip().replace('"', '')

if last\_level > level:

for i in range((last\_level - level) // 2):

path.pop()

if (last\_level - level > 2):

listIndex = 0

path.append(key)

if value:

putValue(result, path, value)

path.pop()

else:

if "-" in key:

key = key.lstrip("- ")

inList = True

key = key + str(listIndex)

path.pop()

if (listIndex == 0):

addList(result, path, key)

else:

addList(result, path, key)

path.append("- " + key)

listIndex += 1

else:

indentDict(result, path)

if (inList):

last\_level = level - 2

else:

last\_level = level

return result

return yamlToDict(yaml\_data)

custom\_parser\_test = lambda x: json.dumps(custom\_parser(x))

yaml\_lib\_test = lambda x: json.dumps(yaml.safe\_load(x))

def test(f, data):

start\_time = time.time()

for \_ in range(100):

f(data)

return time.time() - start\_time

if \_\_name\_\_ == "\_\_main\_\_":

with open("orig.yaml", "r", encoding="utf-8") as f:

data = f.read()

print(

f"Custom Parser: {test(custom\_parser\_test, data)} seconds",

f"YAML Library Parser: {test(yaml\_lib\_test, data)} seconds",

sep="\n",

)

## Задание 7

import yaml

import csv

def yaml\_to\_csv(yaml\_file, csv\_file):

with open(yaml\_file, 'r', encoding='utf-8') as file:

yaml\_data = yaml.safe\_load(file)

with open(csv\_file, 'w', encoding='utf-8', newline='') as file:

csv\_writer = csv.writer(file)

if isinstance(yaml\_data, list):

headers = yaml\_data[0].keys()

csv\_writer.writerow(headers)

for entry in yaml\_data:

csv\_writer.writerow(entry.values())

elif isinstance(yaml\_data, dict):

csv\_writer.writerow(["Key", "Value"])

for key, value in yaml\_data.items():

csv\_writer.writerow([key, value])

else:

raise ValueError("Unsupported YAML format for conversion to CSV.")

print(f"Данные успешно преобразованы из {yaml\_file} в {csv\_file}.")

yaml\_to\_csv("orig.yaml", "output.csv")

## Заключение

В данной лабораторной работе я изучила парсинг, конвертацию файлов, различные типы файлов, языки разметки, формальные грамматики, форматы обмена информацией.

## Список использованных источников

1. Балакшин П.В. Информатика: методическое пособие / П.В. Балакшин, В.В. Соснин – Санкт-Петербург: 2015 г.
2. Балакшин П.В. Лекция №4 Тема: «Исследование протоколов, форматов обмена информацией и языков разметки документов»/ П.В. Балакшин – Санкт-Петербург: 2023-2024 гг.