Санкт-Петербургский Национальный Исследовательский  
Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №4**

По информатике

Вариант 8

Выполнил:

Студент группы P3106

Николаенко Виктор Олегович

Преподаватель:

Балакшин Павел Валерьевич

Кандидат технических наук

Доцент факультета ПИиКТ

Санкт – Петербург 2023

**Оглавление**

[Задание: 3](#_Toc150985176)

[Выполнение работы: 5](#_Toc150985177)

[Исходный формат 5](#_Toc150985178)

[Результирующий формат 6](#_Toc150985179)

[Обязательное задание 7](#_Toc150985180)

[Программа 7](#_Toc150985181)

[Дополнительное задание №1 7](#_Toc150985182)

[Программа 7](#_Toc150985183)

[Дополнительное задание №2 7](#_Toc150985184)

[Программа 7](#_Toc150985185)

[Сравнение решения №2. 7](#_Toc150985186)

[Дополнительное задание №3 7](#_Toc150985187)

[Программа 7](#_Toc150985188)

[Сравнение решения №3. 10](#_Toc150985189)

[Дополнительное задание №4 10](#_Toc150985190)

[Программа 10](#_Toc150985191)

[Сравнение времени выполнения решений. 11](#_Toc150985192)

[Дополнительное задание №5 11](#_Toc150985193)

[Программа 11](#_Toc150985194)

[Результат 12](#_Toc150985195)

[Заключение: 14](#_Toc150985196)

[Список литературы: 15](#_Toc150985197)

# 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).
   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. Объяснение должно быть отражено в отчёте.
10. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
    1. Переписать исходный код, добавив в него использование регулярных выражений.
    2. 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).
    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.
12. Дополнительное задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
    1. Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
    2. 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).
    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. 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.

# Выполнение работы:

## Исходный формат

{  
 "lessons": [  
 {  
 "day": "Вт",  
 "time": "08:20-09:50",  
 "group": "ОснПрофДеят 1.6",  
 "room": {  
 "number": "2112",  
 "corpus": "Кронверский пр., д.49, лит.А"  
 },  
 "sub": "ОСНОВЫ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ (ПРАКТИКА)",  
 "format": "Очный"  
 },  
 {  
 "day": "Вт",  
 "time": "10:00-11:30",  
 "group": "ОснПрофДеят 1.6",  
 "room": [  
 {  
 "number": "2112",  
 "corpus": "Кронверский пр., д.49, лит.А"  
 }  
 ],  
 "sub": "ОСНОВЫ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ (ПРАКТИКА)",  
 "format": "Очный"  
 },  
 {  
 "day": "Вт",  
 "time": "13:30-15:00",  
 "group": "МАТ АН ПИиКТ 10",  
 "room": [  
 {  
 "number": "1405",  
 "corpus": "Кронверский пр., д.49, лит.А"  
 }  
 ],  
 "sub": "МАТЕМАТИЧЕСКИЙ АНАЛИЗ (ЛЕКЦИЯ)",  
 "format": "Очный"  
 },  
 {  
 "day": "Вт",  
 "time": "15:20-16:50",  
 "group": "МАТ АН ПИиКТ 10.3",  
 "room": [  
 {  
 "number": "2201",  
 "corpus": "Кронверский пр., д.49, лит.А"  
 }  
 ],  
 "sub": "МАТЕМАТИЧЕСКИЙ АНАЛИЗ (ПРАКТИКА)",  
 "format": "Очно-дистанционный"  
 }  
 ]  
}

## Результирующий формат

lessons:  
 - day: Вт  
 time: 08:20-09:50  
 group: ОснПрофДеят 1.6  
 room:  
 number: 2112  
 corpus: Кронверский пр., д.49, лит.А  
 sub: ОСНОВЫ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ (ПРАКТИКА)  
 format: Очный  
 - day: Вт  
 time: 10:00-11:30  
 group: ОснПрофДеят 1.6  
 room:  
 - number: 2112  
 corpus: Кронверский пр., д.49, лит.А  
 sub: ОСНОВЫ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ (ПРАКТИКА)  
 format: Очный  
 - day: Вт  
 time: 13:30-15:00  
 group: МАТ АН ПИиКТ 10  
 room:  
 - number: 1405  
 corpus: Кронверский пр., д.49, лит.А  
 sub: МАТЕМАТИЧЕСКИЙ АНАЛИЗ (ЛЕКЦИЯ)  
 format: Очный  
 - day: Вт  
 time: 15:20-16:50  
 group: МАТ АН ПИиКТ 10.3  
 room:  
 - number: 2201  
 corpus: Кронверский пр., д.49, лит.А  
 sub: МАТЕМАТИЧЕСКИЙ АНАЛИЗ (ПРАКТИКА)  
 format: Очно-дистанционный

## Обязательное задание

### Программа

file = open("json/day.json", "r", encoding="UTF-8")  
txt = file.read()  
txt = txt.split("\n")  
  
yaml = ""  
flag = False  
for i in range(len(txt)):  
 line = txt[i]  
 if line.replace(" ", "") in ["{", "}", "},", "[", "]", "],"]:  
 if "{" in line:  
 flag = True  
 else:  
 if flag:  
 line = line[:line.find('"')][:-2] + "- " + line[line.find('"'):]  
 yaml += line.replace('",', "").replace('"', "")[2:]  
 flag = False  
 else:  
 yaml += line.replace('",', "").replace('"', "")[2:]  
 yaml += "\n"  
  
result = open("result/main.yaml", "w", encoding="UTF-8")  
result.write(yaml.replace("[", ""))

## Дополнительное задание №1

### Программа

import json  
import yaml  
  
with open('json/day.json', "r", encoding="UTF-8") as js:  
 data = json.load(js)  
  
with open('result/dop1.yaml', 'w', encoding="UTF-8") as yml:  
 yaml.dump(data, yml, allow\_unicode=True, encoding="UTF-8")  
  
with open('result/dop1.yaml', 'w', encoding="UTF-8") as yml:  
 yaml.safe\_dump(data, yml, allow\_unicode=True, encoding="UTF-8")

## Дополнительное задание №2

### Программа

import re  
  
file = open("json/day.json", "r", encoding="UTF-8")  
txt = file.read()  
txt = txt.split("\n")  
  
yaml = ""  
flag = False  
for i in range(len(txt)):  
 line = txt[i]  
 if re.match(r"\s\*[{}\[\]][\s\W]\*", line):  
 if "{" in line:  
 flag = True  
 else:  
 if flag:  
 sp = re.findall(r'\s\*".+?"', line)  
 sp[0] = sp[0][:sp[0].rfind(" ")] + "-" + sp[0][sp[0].rfind(" ") + 1:]  
 yaml += ":".join(sp).replace('"', "")[2:]  
 if "[" in line:  
 yaml += ":"  
 flag = False  
 else:  
 yaml += ":".join(re.findall(r'\s\*".+?"', line)).replace('"', "")[2:]  
 if "[" in line:  
 yaml += ":"  
 yaml += "\n"  
  
result = open("result/dop2.yaml", "w", encoding="UTF-8")  
result.write(yaml.replace("[", ""))

### Сравнение решения №2.

Использование библиотеки позволяет учесть все конструкции JSON и YAML. Хотя и работает медленнее.

## Дополнительное задание №3

### Программа

import re  
import pprint  
  
  
class Convertor:  
 def \_\_init\_\_(self):  
 self.n = 0  
 self.yaml = ""  
  
 def clear\_el(self, i):  
 if '"' in i:  
 i = i[i.find('"') + 1:i.rfind('"')]  
 return i  
 else:  
 i = i.replace(" ", "").replace("\n", "")  
 if '"' == i[0] or ":" in i:  
 if ',' in i:  
 return i[1: i.rfind(",") - 1]  
 else:  
 return i[1:-1]  
 else:  
 if i[0].isdigit():  
 if ',' in i:  
 return int(i.replace(",", ""))  
 else:  
 return int(i)  
 elif i[:-1] == "null" or i == "null":  
 return None  
 elif i[:-1] == "true" or i == "true":  
 return True  
 elif i[:-1] == "false" or i == "false":  
 return False  
  
 def parse(self, txt, construction, new=False):  
 if new: self.n = 0  
 i = txt[self.n]  
  
 if len(txt) == 1:  
 return self.clear\_el(txt[0])  
  
 while re.fullmatch("\s\*\n", i):  
 self.n += 1  
 i = txt[self.n]  
  
 # Список  
 if re.findall('"[\w\d]+": \[', i):  
 key = re.findall('"[\w\d]+"', i)[0].replace('"', "")  
 self.n += 1  
 if type(construction) is list:  
 construction.append(self.parse(txt, []))  
 else:  
 construction[key] = self.parse(txt, [])  
  
 elif i.replace(" ", "").replace("\n", "") == "[":  
 if construction is None:  
 self.n += 1  
 construction = self.parse(txt, [])  
  
 # Ключ + Элемент  
 elif re.findall('"[\w\d]+": {', i):  
 key = re.findall('"[\w\d]+"', i)[0].replace('"', "")  
 self.n += 1  
 if type(construction) is list:  
 sp = {}  
 sp[key] = self.parse(txt, {})  
 construction.append(sp)  
 else:  
 construction[key] = self.parse(txt, {})  
  
 # Элемент  
 elif i.replace(" ", "").replace("\n", "") == "{":  
 self.n += 1  
 if type(construction) is list:  
 construction.append(self.parse(txt, {}))  
 else:  
 return self.parse(txt, {})  
  
 # Ключ - Значение  
 elif re.findall('"[\w\d]+": "[\w\d\W\s]+"', i) or re.findall('"[\w\d]+": [\w\d]+', i):  
 key = self.clear\_el(re.findall('"[\w\d]+"', i)[0])  
 i.replace(key, "")  
 if len(re.findall('"[\w\d]+": "[\w\d\W\s]+"', i)) == 1:  
 value = re.findall('"[\w\d\W\s]+"', i[i.find(":"):])[0]  
 else:  
 value = re.findall('[\w\d]+', i[i.find(":"):])[0]  
  
 value = self.clear\_el(value)  
 if type(construction) is list:  
 sp = {}[key] = value  
 construction.append(sp)  
 else:  
 construction[key] = value  
  
 # Значение  
 elif re.findall('[\w\d\W\s]+', i):  
 if re.findall('[}\]]', i):  
 return construction  
  
 for x in i:  
 if x in ["{", "[", "]", "}"]:  
 break  
 else:  
 construction.append(self.clear\_el(i))  
  
 # Выход из рекурсии  
 if len(txt) == self.n + 1:  
 self.n = 0  
 return construction  
  
 # Конец конструкции  
 if re.findall('[}\]]', i):  
 return construction  
  
 self.n += 1  
 return self.parse(txt, construction)  
  
 def to\_yaml(self, dictionary, tab=0, flag=False, new=False):  
 if new:  
 if type(dictionary) is dict:  
 if len(dictionary.keys()) == 0:  
 return "{}"  
 if type(dictionary) is list:  
 if len(dictionary) == 0:  
 return "[]"  
 elif type(dictionary) is bool or type(dictionary) is str or type(dictionary) is int:  
 return str(dictionary)  
 self.n = 0  
 self.yaml = ""  
  
 if type(dictionary) is dict:  
 for i in dictionary.keys():  
 if type(dictionary[i]) is not list and type(dictionary[i]) is not dict:  
 if flag:  
 self.yaml += " " \* (tab - 2) + "- " + i + ": " + str(dictionary[i]) + "\n"  
 flag = False  
 else:  
 self.yaml += " " \* tab + i + ": " + str(dictionary[i]) + "\n"  
 else:  
 if flag:  
 self.yaml += " " \* (tab - 2) + "- " + i + ":\n"  
 flag = False  
 else:  
 self.yaml += " " \* tab + i + ":\n"  
 tab += 2  
 self.to\_yaml(dictionary[i], tab)  
 tab -= 2  
 else:  
 if type(dictionary) is list:  
 tab += 2  
 for i in dictionary:  
 self.to\_yaml(i, tab, True)  
 tab -= 2  
 else:  
 self.yaml += " " \* (tab - 2) + "- " + str(dictionary) + "\n"  
 return self.yaml  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 data = open("json/days.json", "r", encoding="UTF-8").readlines()  
 convertor = Convertor()  
 pp = pprint.PrettyPrinter(indent=2)  
  
 dictionary = convertor.parse(data, None)  
 with open('result/3\_parser\_json\_to\_yaml.yaml', 'w', encoding="UTF-8") as file:  
 file.write(convertor.to\_yaml(dictionary, 0, False, True))

### Сравнение решения №3.

Вариант с использованием регулярных выражений выполняется медленнее, чем вариант без их использования, так как регулярные выражения не очень эффективны при обработке небольших файлов. Регулярные выражения лучше подходят для работы с большими объемами данных.

## Дополнительное задание №4

### Программа

from rep\_json\_to\_yaml import no\_reg  
from lib\_json\_to\_yaml import lib  
from reg\_json\_to\_yaml import reg  
from parser\_json\_to\_yaml import Convertor  
  
import time  
import json  
import yaml  
import os  
import re  
  
start\_time = time.time()  
for i in range(1000):  
 no\_reg('json/day.json')  
end\_time = time.time()  
print("Время без регулярых выражений (1000) -", end\_time-start\_time)  
  
start\_time = time.time()  
for i in range(1000):  
 lib('json/day.json')  
end\_time = time.time()  
print("Время для библиотеки (1000) -", end\_time-start\_time)  
  
start\_time = time.time()  
for i in range(1000):  
 reg('json/day.json')  
end\_time = time.time()  
print("Время с регулярными выражениями (1000) -", end\_time-start\_time)  
  
con = Convertor()  
start\_time = time.time()  
for i in range(1000):  
 data = open('json/day.json', "r", encoding="UTF-8").readlines()  
 data = con.parse(data, None)  
 with open('result/time.yaml', 'w', encoding="UTF-8") as file:  
 file.write(con.to\_yaml(data, 0, False, True))  
end\_time = time.time()  
print("Время для моего парсера (1000) -", end\_time-start\_time)  
  
os.remove('result/time.yaml')

### Сравнение времени выполнения решений.

Для 1000 итераций

|  |  |
| --- | --- |
| Решение №0 | 0.36862707138061523 |
| Решение №1 | 4.522734880447388 |
| Решение №2 | 0.4427640438079834 |
| Решение №3 | 1.8532090187072754 |

Решение №0 (без использования каких-либо библиотек и регулярных выражений)  
Самое быстрое решение. Это связано с тем, что функционал данного самый ограниченный.

Решение №1 (с использованием библиотеки)  
Самое долгое, так как учитывает все особенности конструкций JSON.

Решение №2 (с использованием регулярных выражений)  
Регулярные выражения оказываются менее эффективными при разборе тегов и значений короткой длины, когда их производительность не так очевидна и простые строчные методы не уступают им в скорости.

Решение №3 (parser)  
Решение с парсингом данных значительно медленнее №0 и №1, так как более универсальна, а также позволяет спарсить данные в словарь. Такое решение выигрывает по скорости в сравнении с библиотекой (в силу того, что библиотека учитывает все особенности, конструкции, производит проверки)

## Дополнительное задание №5

### Программа

def no\_reg(name):  
 file = open(name, "r", encoding="UTF-8")  
 txt = file.read()  
 txt = txt.split("\n")  
  
 yaml = ""  
 flag = False  
 for i in range(len(txt)):  
 line = txt[i]  
 if line.replace(" ", "") in ["{", "}", "},", "[", "]", "],"]:  
 if "{" in line:  
 flag = True  
 else:  
 if flag:  
 line = line[:line.find('"')][:-2] + line[line.find('"'):]  
 yaml += line.replace('",', "").replace('"', "")  
 flag = False  
 else:  
 yaml += line.replace('",', "").replace('"', "")[2:]  
 yaml += "\n"  
  
 result = open("result/parser\_json\_to\_md.md", "w", encoding="UTF-8")  
 result.write(yaml.replace("{", "").replace("[", "").replace("}", "").replace("]", "").replace(" ", "&nbsp;"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 no\_reg("json/day.json")

### Результат

Конвертация в md (markdown)

Изображение выглядит как текст, снимок экрана, документ, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Ключевое отличие MD от других языков разметки заключается в его предназначении: он используется для визуализации данных и не подходит для их хранения или передачи. В противоположность этому, JSON используется для хранения и передачи информации, например, на сервере.  
Преобразование JSON в MD сохраняет наглядность и понятность представления данных. Этот процесс может быть полезным при написании статей, позволяя преобразовать данные с сервера в MD и включить их в статью для более наглядного и красивого представления.

# Заключение:

В ходе работы я познакомился с основными типами данных, понятием “формальные грамматики”, устройством структур данных. Научился парсить данные

# Список литературы:

1. Форма Бэкуса-Наура (<https://ru.wikipedia.org/wiki/Форма_Бэкуса_—_Наура>)
2. Балакшин П.В, доцент факультета ПИиКТ «Презентация по информатике» (<https://t.me/balakshin_students>)