**Федеральное государственное автономное образовательное учреждение высшего образования**
**«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»**

**Факультет Программной Инженерии и Компьютерной**

**ТехникиДисциплина: Информатика**

# Лабораторная работа №4

Выполнил: Кудрявцева Р.С.
Группа: P3117
Преподаватель:Марухленко Д.С.

Санкт-Петербург, 2024г

# Оглавление

## Задание

1. Определить номер варианта как остаток деления на 36 последних двух цифр своего идентификационного номера в ISU: например, 125**98** / 36 = 26. В случае, если в оба указнных дня недели нет занятий, то увеличить номер варианта на восемь. В случае, если занятий нет и в новом наборе дней, то продолжать увеличивать 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 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.
7. **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 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.

| | | | |
|---|---|---|---|
| 8 | JSON | YAML | Вторник, пятница |

## Основные этапы выполнения

### 1. Обязательное задание

Исходный файл json: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/json_file.json

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/osnov/code_osnov.py

```
stdin = open('json_file.json', 'r')
stdout = open('schedule_yaml.yaml', 'w')

def f(file):
    text = file.read()
    ans_yaml = ""
    line = text.split('\n')
    for i in range(1, len(line) - 1):
        left = 0
        right = 0
        counter = line[i].count('\t')
        for j in range(len(line[i])):
```

```
            if left == 0 and line[i][j] == '"':
                left = j + 1
            elif left != 0 and line[i][j] == '"':
                right = j
        ans = line[i][left:right]
        ans = ans.replace('"', '')

        if len(ans) != 0:
            if ',' in line[i - 1]:
                if "_teacher" in ans or "_place" in ans or "_tipe" in ans or
"_lesson" in ans:
                    ans_yaml += (counter - 3) * '  ' + ans + "\n"
            elif '},' in line[i - 2] and '{' in line[i - 1]:
                if '[' in line[i]:
                    ans_yaml += (counter - 3) * '  ' + '- ' + ans + ":\n"
                else:
                    ans_yaml += (counter - 4) * '  ' + '- ' + ans + "\n"
            elif '[' in line[i - 2]:
                if '[' in line[i]:
                    ans_yaml += (counter - 3) * '  ' + '- ' + ans + ":\n"
                else:
                    ans_yaml += (counter - 4) * '  ' + '- ' + ans + "\n"
            elif '{' in line[i - 1]:
                ans_yaml += (counter - 1) * '  ' + ans + ':\n'
    return ans_yaml

stdout.write(f(stdin))
stdout.close()
stdin.close()
```

Результат в yaml: https://github.com/Creature-bot/Ruslana/blob/abf0925f4f90a67a2d4931f439d1f1ae94449469/infa/infa_sem1/laba4/osnov/schedule_yaml.yaml

## 2. Дополнительное задание №1

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop1/code_dop1.py

```
import json
import yaml

with open('json_file.json') as json_file:
  data = json.load(json_file)
with open('schedule_yaml_library.yaml', 'w') as yaml_file:
  yaml.dump(data, yaml_file, allow_unicode=True, sort_keys=False)
```

Результат: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop1/schedule_yaml_library.yaml

Готовые библиотеки: стандартная библиотека Python json для парсинга json и pyyaml для дампа словаря в файл yaml.

Файл результата не отличается от результата обязательного задания. Код программы стал значительно проще – теперь он состоит из нескольких строчек открытия и закрытия файлов, для их преобразования.

## 3. Дополнительное задание №2

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop2/lab4_dop2.py

```
import re

stdin = open('json_file.json', 'r')
stdout = open('schedule_yaml_regex.yaml', 'w')

def json_to_yaml(file):
    text = file.read()
    ans_yaml = ""
    key_value_pair = re.compile(r'"(\w+)":\s*"?([^"\n}]+)"?,*')

    for line in text.splitlines():

        counter = line.count('\t')
        line = line.replace('{', '')
        line = line.replace('[', '')
        match = key_value_pair.search(line)

        if match:
            key, value = match.groups()
            if key == "_teacher" or key == "_place" or key == "_lesson" or
key == "_tipe":
                ans_yaml += '  ' * (counter - 3) + f"{key}: {value}\n"
            elif key == "_time":
                ans_yaml += '  ' * (counter - 4) + '- ' + f"{key}: {value}\n"
            elif 'lesson_' in key:
                ans_yaml += '  ' * (counter - 3) + '- ' + f"{key}: {value}\n"
            else:
                ans_yaml += '  ' * (counter - 1) + f"{key}: {value}\n"

    return ans_yaml
```

```
yaml_content = json_to_yaml(stdin)
stdout.write(yaml_content)

stdin.close()
stdout.close()
```

Результат: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop2/schedule_yaml_regex.yaml

Файл результата полностью совпадает с файлом результата обязательного задания.

Регулярные выражения были использованы для более простого парсинга литералов JSON: строк, чисел, булевых значений и null.

## 4. Дополнительное задание №3

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop3/code_dop3.py

```
from lark import Lark, Transformer
import yaml

json_grammar = """
    ?start: value
    ?value: object
          | array
          | string
          | SIGNED_NUMBER      -> number
          | "true"             -> true
          | "false"            -> false
          | "null"             -> null

    array  : "[" [value ("," value)*] "]"
    object : "{" [pair ("," pair)*] "}"
    pair   : string ":" value
    string : ESCAPED_STRING

    %import common.ESCAPED_STRING
    %import common.SIGNED_NUMBER
    %import common.WS
    %ignore WS
"""

json_parser = Lark(json_grammar, start='value', parser='lalr')

class JSONTransformer(Transformer):
    def object(self, items):
```

```
        return dict(items)

    def array(self, items):
        return items

    def string(self, s):
        return s[0][1:-1]

    def number(self, n):
        if '.' in n[0]:
            return float(n[0])
        else:
            return int(n[0])

    def true(self, _):
        return True

    def false(self, _):
        return False

    def null(self, _):
        return None

    def pair(self, items):
        key = items[0][0:].strip()
        value = items[1]
        return key, value

def json_to_yaml(json_data, yaml_filename):
    tree = json_parser.parse(json_data)
    transformer = JSONTransformer()
    data = transformer.transform(tree)

    with open(yaml_filename, 'w') as yaml_file:
        yaml.dump(data, yaml_file, allow_unicode=True, sort_keys=False)

with open('json_file.json') as file:
    json_content = file.read()
    json_to_yaml(json_content, 'schedule_yaml_grammar.yaml')
```

Результат: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop3/schedule_yaml_grammar.yaml

## 5. Дополнительное задание №4

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop4/code_dop4.py

**import time**

```
start_time = time.perf_counter()
for i in range(100):
    import lab4_osnov
end_time = time.perf_counter()
print(f"Основа - {end_time - start_time:.8f} сек")

start_time = time.perf_counter()
for i in range(100):
    import lab4_dop1
end_time = time.perf_counter()
print(f"Доп 1  - {end_time - start_time:.8f} сек")

start_time = time.perf_counter()
for i in range(100):
    import lab4_dop2
end_time = time.perf_counter()
print(f"Доп 2  - {end_time - start_time:.8f} сек")

start_time = time.perf_counter()
for i in range(100):
    import lab4_dop3
end_time = time.perf_counter()
print(f"Доп 3  - {end_time - start_time:.8f} сек")
```

Результат:

Основа - 0.00091448 сек

Доп 1  - 0.03714072 сек

Доп 2  - 0.00137624 сек

Доп 3  - 0.09534595 сек

Возможное объяснение результатов:

Быстрее всего работает Основное задание, наверное потому, что в нем не используются никакие библиотеки и из-за своей примитивности оно считывает файл посимвольно, не утруждаясь в отдельном сохранение данных для последующей конвертации.

Дольше всего работает Дополнительное задание №3, вероятно потому, что ему приходится использовать несколько библиотек, в том числе lark, работающую с формальными грамматиками. Сами грамматики требуют преобразования из исходного формата в другой, общий. А после уже используется библиотека yaml, то есть надо из общего формата, конвертировать в Yaml. Весь процесс проходит через несколько этапов, что замедляет работу кода.

## 6. Дополнительное задание №5

Исходный код: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop5/code_dop5.py

```
from lark import Lark, Transformer
import toml

json_grammar = """
    ?start: value
    ?value: object
          | array
          | string
          | SIGNED_NUMBER      -> number
          | "true"             -> true
          | "false"            -> false
          | "null"             -> null

    array  : "[" [value ("," value)*] "]"
    object : "{" [pair ("," pair)*] "}"
    pair   : string ":" value
    string : ESCAPED_STRING

    %import common.ESCAPED_STRING
    %import common.SIGNED_NUMBER
    %import common.WS
    %ignore WS
"""

json_parser = Lark(json_grammar, start='value', parser='lalr')

class JSONTransformer(Transformer):
    def object(self, items):
        return dict(items)

    def array(self, items):
        return items

    def string(self, s):
```

```
            return s[0][1:-1]

        def number(self, n):
            if '.' in n[0]:
                return float(n[0])
            else:
                return int(n[0])

        def true(self, _):
            return True

        def false(self, _):
            return False

        def null(self, _):
            return None

        def pair(self, items):
            key = items[0]
            value = items[1]
            return key, value

    def json_to_toml(json_data, toml_filename):
        tree = json_parser.parse(json_data)
        transformer = JSONTransformer()
        data = transformer.transform(tree)

        with open(toml_filename, 'w') as toml_file:
            toml.dump(data, toml_file)

    with open('json_file.json') as file:
        json_content = file.read()
        json_to_toml(json_content, 'schedule_toml.toml')
```

Результат: https://github.com/Creature-bot/Ruslana/blob/ba2270c28dfd838865d39513cbe87d0f2ca5fd0a/infa/infa_sem1/laba4/dop5/schedule_toml.toml

## Список литературы


1. Балакшин П.В., Соснин В.В., Калинин И.В., Малышева Т.А., Раков С.В., Рущенко Н.Г., Дергачев А.М. Информатика: лабораторные работы и тесты: Учебно-методическое пособие / Рецензент: Поляков В.И. - Санкт-Петербург: Университет ИТМО, 2019. - 56 с. - экз. - Режим доступа: https://books.ifmo.ru/book/2248/informatika:_laboratornye_raboty_i_testy:_uchebno-metodicheskoe_posobie_/_recenzent:_polyakov_v.i..htm
2. Грошев А.С. Г89 Информатика: Учебник для вузов / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2010. -470с. -Режим доступа https://narfu.ru/university/library/books/0690.pdf