Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

Вариант №18
Лабораторная работа №4
по дисциплине
Информатика

Выполнил Студент группы P3115
Владимир Мацюк
Преподаватель:
Малышева Татьяна Алексеевна

г. Санкт-Петербург
2022г.

Содержание

1 Задание

1.1 Вариант

1. Определить номер варианта как остаток деления на 36 порядкового номера в списке группы в ISU. В случае, если в данный день недели нет занятий, то увеличить номер варианта на восемь.

18	JSON	XML	Четверг

2. Изучить форму Бэкуса-Наура.
3. Изучить особенности языков разметки/форматов JSON, YAML, XML.
4. Понять устройство страницы с расписанием для своей группы: http://itmo.ru/ru/schedule/0/P3110/schedule.htm

1.2 Исходный файл

5. 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.

```
{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to create markup
                ↪ languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
```

```
  }
}
```

1.3 Обязательное задание

7. Обязательное задание (позволяет набрать до 65 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.

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

task1.py

```python
from dataclasses import dataclass
from enum import Enum, auto
from pathlib import Path
from typing import Iterator


class TokenType(Enum):
    NONE = auto()
    BEGIN_OBJ = auto()
    END_OBJ = auto()
    BEGIN_ARR = auto()
    END_ARR = auto()
    STR = auto()
    COL = auto()
    COMMA = auto()


@dataclass(init=True)
class Token:
    val: str
    type: TokenType


def tokenise(s: str):
    tokens: list[Token] = []

    it = iter(s)
    while True:
        try:
            i = next(it)
        except StopIteration:
            break
        cur = Token(i, TokenType.NONE)
        match i:
            case '{': cur.type = TokenType.BEGIN_OBJ
            case '}': cur.type = TokenType.END_OBJ
            case '[': cur.type = TokenType.BEGIN_ARR
            case ']': cur.type = TokenType.END_ARR
            case ':': cur.type = TokenType.COL
            case ',': cur.type = TokenType.COMMA
            case '"':
                cur.type = TokenType.STR
                cur.val = ''
                while True:
```

```
                try:
                    i = next(it)
                except StopIteration:
                    break
                if i == "\\":
                    i = next()
                if i == '"':
                    break
                cur.val += i

        if cur.type != TokenType.NONE:
            tokens.append(cur)

    return tokens


def parse_tokens(it: Iterator[Token]):
    i = next(it)
    match i.type:
        case TokenType.STR:
            return i.val
        case TokenType.BEGIN_OBJ:
            res = {}
            i = next(it)
            while True:
                if i.type != TokenType.STR:
                    raise RuntimeError('expected str key')
                key = i.val
                i = next(it)
                if i.type != TokenType.COL:
                    raise RuntimeError('expected ':'')
                val = parse_tokens(it)
                res[key] = val
                i = next(it)

                if i.type == TokenType.COMMA:
                    i = next(it)
                elif i.type == TokenType.END_OBJ:
                    break
                else:
                    raise RuntimeError('unexpected token')
            return res
        case TokenType.BEGIN_ARR:
            res = []
            while True:
                val = parse_tokens(it)
                res.append(val)
                i = next(it)
                if i.type == TokenType.COMMA:
                    continue
                elif i.type == TokenType.END_ARR:
                    break
                else:
                    raise RuntimeError('unexpected token')
            return res


def parse_json(s: str):
```

```
    tokens = tokenise(s)
    res = parse_tokens(iter(tokens))
    return res


def obj2xml(obj, deep=0, parent='root') -> str:
    sp = ' ' * deep
    match obj:
        case str(): return obj
        case dict():
            res = ''
            for (key, val) in obj.items():
                match val:
                    case list():
                        res += obj2xml(val, deep+1, key)
                    case str():
                        res += f'{sp}<{key}>{val}</{key}>\n'
                    case _:
                        res += f'{sp}<{key}>\n{obj2xml(val, deep+1, key)
                            ↪ }{sp}</{key}>\n'
            return res
        case list():
            return ''.join(f'{sp}<{parent}>{obj2xml(val, deep+1)}</{
                ↪ parent}>\n' for val in obj)


def task1(s: str):
    res = parse_json(s)
    return obj2xml(res)
```

out1.xml

```
<glossary>
  <title>example glossary</title>
  <GlossDiv>
    <title>S</title>
    <GlossList>
      <GlossEntry>
        <ID>SGML</ID>
        <SortAs>SGML</SortAs>
        <GlossTerm>Standard Generalized Markup Language</GlossTerm>
        <Acronym>SGML</Acronym>
        <Abbrev>ISO 8879:1986</Abbrev>
        <GlossDef>
          <para>A meta-markup language, used to create markup
              ↪ languages such as DocBook.</para>
          <GlossSeeAlso>GML</GlossSeeAlso>
          <GlossSeeAlso>XML</GlossSeeAlso>
        </GlossDef>
        <GlossSee>markup</GlossSee>
      </GlossEntry>
    </GlossList>
  </GlossDiv>
</glossary>
```

1.4 Дополнительное задание №1

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.

task2.py

```python
import json
from dict2xml import dict2xml


def parse_json(s: str):
    return json.loads(s)


def obj2xml(s: str):
    return dict2xml(s)


def task2(s: str):
    res = parse_json(s)
    return obj2xml(res)
```

out2.xml

```xml
<glossary>
  <GlossDiv>
    <GlossList>
      <GlossEntry>
        <Abbrev>ISO 8879:1986</Abbrev>
        <Acronym>SGML</Acronym>
        <GlossDef>
          <GlossSeeAlso>GML</GlossSeeAlso>
          <GlossSeeAlso>XML</GlossSeeAlso>
          <para>A meta-markup language, used to create markup
            ↪ languages such as DocBook.</para>
        </GlossDef>
        <GlossSee>markup</GlossSee>
        <GlossTerm>Standard Generalized Markup Language</GlossTerm>
        <ID>SGML</ID>
        <SortAs>SGML</SortAs>
      </GlossEntry>
    </GlossList>
    <title>S</title>
  </GlossDiv>
  <title>example glossary</title>
</glossary>
```

1.5 Дополнительное задание №2

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

(a) Переписать исходный код, добавив в него использование 2 регулярных выражений.
(b) Сравнить полученные результаты и объяснить их сходство/различие.

task3.py

```python
import re
from task1 import Token, TokenType, obj2xml, parse_tokens


def tokenise(s: str):
    return [
        Token(match.group(match.lastindex), TokenType(match.
            ↪ lastindex + 1))
        for match in re.finditer('|'.join([
            r'(\{)',
            r'(\})',
            r'(\[)',
            r'(\])',
            r'"((?:\.|[^"])*)"',
            r'(:)',
            r'(,)',
        ]), s)
    ]


def parse_json(s: str):
    tokens = tokenise(s)
    res = parse_tokens(iter(tokens))
    return res


def task3(s: str):
    res = parse_json(s)
    return obj2xml(res)
```

out3.xml

```xml
<glossary>
  <title>example glossary</title>
  <GlossDiv>
    <title>S</title>
    <GlossList>
      <GlossEntry>
        <ID>SGML</ID>
        <SortAs>SGML</SortAs>
        <GlossTerm>Standard Generalized Markup Language</GlossTerm>
        <Acronym>SGML</Acronym>
        <Abbrev>ISO 8879:1986</Abbrev>
        <GlossDef>
          <para>A meta-markup language, used to create markup
              ↪ languages such as DocBook.</para>
          <GlossSeeAlso>GML</GlossSeeAlso>
          <GlossSeeAlso>XML</GlossSeeAlso>
        </GlossDef>
        <GlossSee>markup</GlossSee>
      </GlossEntry>
    </GlossList>
  </GlossDiv>
</glossary>
```

1.6 Дополнительное задание №3

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

    (a) 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.

    (b) Проанализировать полученные результаты и объяснить их сходство/различие.

task4.py

```python
from pathlib import Path


def read():
    return open(Path(__file__).with_name('input.json')).read()


def write(idx: int, out: str, ext='xml'):
    open(Path(__file__).with_name(f'out{idx}.{ext}'), mode="w+").write(out)


def task4():
    from task1 import task1
    from task2 import task2
    from task3 import task3
    from timeit import timeit
    s = read()

    write(1, task1(s))
    write(2, task2(s))
    write(3, task3(s))

    res = '\n'.join(
        s + str(t) for (s, t) in
        [
            ('no lib + no regex: ', timeit("task1(s)", globals=locals(), number=100)),
            ('lib: ', timeit("task2(s)", globals=locals(), number=100)),
            ('regex: ', timeit("task3(s)", globals=locals(), number=100))
        ]
    )
    open(Path(__file__).with_name(f'out4.txt'), mode="w+").write(res)
    print(res)


if __name__ == '__main__':
    task4()
```

out4.txt

```
no lib + no regex: 0.02160260699929495
lib: 0.009555570999509655
regex: 0.010486863000551239
```

1.7 Дополнительное задание №4

12. Дополнительное задание №4 (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.

task5.py

```python
from task2 import parse_json
from task4 import read, write


def val2toml(item: any):
    match item:
        case str():
            return item.__repr__()
        case list():
            return f'[{", ".join(val2toml(i) for i in item)}]'
        case _:
            raise RuntimeError('forbidden type')


def obj2toml(item: dict, parent=''):
    res = []

    def tables_at_end(item):
        _, value = item
        return isinstance(value, dict)

    for (key, val) in sorted(item.items(), key=tables_at_end):
        match val:
            case dict():
                path = f"{parent}.{key}" if parent else key
                res.append(f'\n[{path}]\n{obj2toml(val, path)}')
            case _:
                res.append(f'{key} = {val2toml(val)}')
    return '\n'.join(res)


s = read()
d = parse_json(s)
res = obj2toml(d)
write(5, res, 'toml')
```

out5.toml

```toml

[glossary]
title = 'example glossary'

[glossary.GlossDiv]
title = 'S'

[glossary.GlossDiv.GlossList]
```

```
 9
10 [glossary.GlossDiv.GlossList.GlossEntry]
11 ID = 'SGML'
12 SortAs = 'SGML'
13 GlossTerm = 'Standard Generalized Markup Language'
14 Acronym = 'SGML'
15 Abbrev = 'ISO 8879:1986'
16 GlossSee = 'markup'
17
18 [glossary.GlossDiv.GlossList.GlossEntry.GlossDef]
19 para = 'A meta-markup language, used to create markup languages
       ↪ such as DocBook.'
20 GlossSeeAlso = ['GML', 'XML']
```

1.8 Итог

13. Проверить, что все пункты задания выполнены и выполнены верно.
14. Написать отчёт о проделанной работе.
15. Подготовиться к устным вопросам на защите

2 Вывод

sdfsdf