Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Отчёт по лабораторной работе №4

По дисципление: Информатика

Тема: «Исследование протоколов, форматов обмена информацией и языков разметки документов»

Вариант (по ИСУ) 13

Выполнил: Разыграев Кирилл Сергеевич

Группа: P3115

Преподаватель: Белокон Юлия Алексеевна

Санкт-Петербург, 2024

Содержание

1 Задание

- **Обязательное задание**. 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

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

- **Дополнительное задание №1**

  – 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

- **Дополнительное задание №2**

  – Переписать исходный код, добавив в него использование регулярных выражений

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

- **Дополнительное задание №3**

  – 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

- **Дополнительное задание №4**
  - Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле
  - Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте

- **Дополнительное задание №5**
  - Переписать исходную программу, чтобы она осуществляла парсинг и конвертацию файла в любой другой формат (кроме JSON, YAML, XML, HTML): PROTOBUF, TSV, CSV, WML и т.п.
  - Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте

2 Исходный файл

```
<schedule>
    <weekday>
        <name>Среда</name>
        <lesson>
            <name>Информатика</name>
            <type>Лекция</type>
```

<location>Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит. М</location>
<teacher>Балакшин Павел Валерьевич</teacher>
<start_at>8:20</start_at>
<end_at>9:50</end_at>
</lesson>
<lesson>
<name>Основы профессиональной деятельности</name>
<type>Лекция</type>
<location>Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит. М</location>
<teacher>Клименков Сергей Викторович</teacher>
<start_at>10:00</start_at>
<end_at>11:30</end_at>
</lesson>
<lesson>
<name>Основы дискретной математики</name>
<type>Лекция</type>
<location>Ауд. 1428, Кронверкский пр., д.49, лит.А</location>
<teacher>Поляков Владимир Иванович</teacher>
<start_at>13:30</start_at>
<end_at>15:00</end_at>
</lesson>
<lesson>
<name>Основы дискретной математики</name>
<type>Практика</type>
<location>Ауд. 1428, Кронверкский пр., д.49, лит.А</location>
<teacher>Поляков Владимир Иванович</teacher>
<start_at>15:20</start_at>
<end_at>16:50</end_at>
</lesson>
</weekday>
<weekday>
<name>Суббота</name>
<lesson>
<name>Математический анализ</name>
<type>Лекция</type>
<location>zoom</location>
<teacher>Блейхер Оксана Владимировна</teacher>
<start_at>10:00</start_at>
<end_at>11:30</end_at>
</lesson>

```
        <lesson>
            <name>Программирование</name>
            <type>Лабораторная</type>
            <location>Ауд. 1338, Кронверкский пр., д.49, лит.А</location>
            <teacher>Кулинич Ярослав Вадимович</teacher>
            <start_at>13:30</start_at>
            <end_at>15:00</end_at>
        </lesson>
        <lesson>
            <name>Программирование</name>
            <type>Лабораторная</type>
            <location>Ауд. 1338, Кронверкский пр., д.49, лит.А</location>
            <teacher>Кулинич Ярослав Вадимович</teacher>
            <start_at>15:20</start_at>
            <end_at>16:50</end_at>
        </lesson>
    </weekday>
</schedule>
```

3 Основные этапы вычисления

3.1 Обязательное задание

3.1.1 Исходный код

```
from pathlib import Path
from typing import Any


def xml2dict(xml: str) -> dict[str, Any]:
    def get_tag_content(xml_str, tag) -> tuple[str | None, str]:
        start_tag = f"<{tag}>"
        end_tag = f"</{tag}>"
        start_idx = xml_str.find(start_tag) + len(start_tag)
        end_idx = xml_str.find(end_tag)
        if start_idx == -1 or end_idx == -1:
            return None, xml_str

        content = xml_str[start_idx:end_idx].strip()
```

```
        remaining_xml = xml_str[end_idx + len(end_tag):]

        return content, remaining_xml

    def parse_element(xml_str) -> dict[str, Any]:
        result = {}
        while "<" in xml_str and ">" in xml_str:
            start = xml_str.find("<") + 1
            end = xml_str.find(">", start)
            tag = xml_str[start:end]
            content, xml_str = get_tag_content(xml_str, tag)

            if "<" in content and ">" in content:
                content = parse_element(content)

            if tag in result:
                if not isinstance(result[tag], list):
                    result[tag] = [result[tag]]
                result[tag].append(content)
            else:
                result[tag] = content
        return result

    root_tag = xml[xml.find("<") + 1:xml.find(">")]
    root_content, _ = get_tag_content(xml, root_tag)

    return {root_tag: parse_element(root_content)}


def dict2json(data: dict[str, Any]) -> str:
    def convert(value):
        if isinstance(value, dict):
            items = [f'"{key}": {convert(val)}' for key, val in value.items()]
            return "{" + ", ".join(items) + "}"
        elif isinstance(value, list):
            items = [convert(item) for item in value]
            return "[" + ", ".join(items) + "]"
        elif isinstance(value, str):
            return f'"{value}"'
        elif isinstance(value, (int, float)):
```

```
            return str(value).lower()
        elif value is None:
            return "null"
        elif isinstance(value, bool):
            return "true" if value else "false"
        else:
            raise TypeError(f"Unsupported data type: {type(value)}")

    return convert(data)


def xml2json(xml: str) -> str:
    parsed_data = xml2dict(xml)
    json_data = dict2json(parsed_data)

    return json_data


def main() -> None:
    xml_path = Path(__file__).parent.joinpath("schedule.xml")
    json_path = Path(__file__).parent.joinpath("schedule_1.json")

    xml_schedule = xml_path.read_text(encoding="utf-8")
    json_schedule = xml2json(xml_schedule)

    json_path.write_text(json_schedule, encoding="utf-8")


if __name__ == "__main__":
    main()
```

3.1.2 Результат


```
{
  "schedule": {
    "weekday": [
      {
        "name": "Среда",
        "lesson": [
          {
            "name": "Информатика",
```

```
      "type": "Лекция",
      "location": "Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит.
      ↪  М",
      "teacher": "Балакшин Павел Валерьевич",
      "start_at": "8:20",
      "end_at": "9:50"
    },
    {
      "name": "Основы профессиональной деятельности",
      "type": "Лекция",
      "location": "Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит.
      ↪  М",
      "teacher": "Клименков Сергей Викторович",
      "start_at": "10:00",
      "end_at": "11:30"
    },
    {
      "name": "Основы дискретной математики",
      "type": "Лекция",
      "location": "Ауд. 1428, Кронверкский пр., д.49, лит.А",
      "teacher": "Поляков Владимир Иванович",
      "start_at": "13:30",
      "end_at": "15:00"
    },
    {
      "name": "Основы дискретной математики",
      "type": "Практика",
      "location": "Ауд. 1428, Кронверкский пр., д.49, лит.А",
      "teacher": "Поляков Владимир Иванович",
      "start_at": "15:20",
      "end_at": "16:50"
    }
  ]
},
{
  "name": "Суббота",
  "lesson": [
    {
      "name": "Математический анализ",
      "type": "Лекция",
```

```
            "location": "zoom",
            "teacher": "Блейхер Оксана Владимировна",
            "start_at": "10:00",
            "end_at": "11:30"
          },
          {
            "name": "Программирование",
            "type": "Лабораторная",
            "location": "Ауд. 1338, Кронверкский пр., д.49, лит.А",
            "teacher": "Кулинич Ярослав Вадимович",
            "start_at": "13:30",
            "end_at": "15:00"
          },
          {
            "name": "Программирование",
            "type": "Лабораторная",
            "location": "Ауд. 1338, Кронверкский пр., д.49, лит.А",
            "teacher": "Кулинич Ярослав Вадимович",
            "start_at": "15:20",
            "end_at": "16:50"
          }
        ]
      }
    ]
  }
}
```


3.2 Дополнительное задание №1

3.2.1 Исходный код

```python
import json
from pathlib import Path

import xmltodict


def xml2json(xml: str) -> str:
    parsed_data = xmltodict.parse(xml)
    json_data = json.dumps(parsed_data, ensure_ascii=False)
```

```
    return json_data


def main() -> None:
    xml_path = Path(__file__).parent.joinpath("schedule.xml")
    json_path = Path(__file__).parent.joinpath("schedule_2.json")

    xml_schedule = xml_path.read_text(encoding="utf-8")
    json_schedule = xml2json(xml_schedule)

    json_path.write_text(json_schedule, encoding="utf-8")


if __name__ == "__main__":
    main()
```

3.2.2 Результат


```
{
  "schedule": {
    "weekday": [
      {
        "name": "Среда",
        "lesson": [
          {
            "name": "Информатика",
            "type": "Лекция",
            "location": "Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит.
            ↪  М",
            "teacher": "Балакшин Павел Валерьевич",
            "start_at": "8:20",
            "end_at": "9:50"
          },
          {
            "name": "Основы профессиональной деятельности",
            "type": "Лекция",
            "location": "Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит.
            ↪  М",
            "teacher": "Клименков Сергей Викторович",
            "start_at": "10:00",
```

```
      "end_at": "11:30"
    },
    {
      "name": "Основы дискретной математики",
      "type": "Лекция",
      "location": "Ауд. 1428, Кронверкский пр., д.49, лит.А",
      "teacher": "Поляков Владимир Иванович",
      "start_at": "13:30",
      "end_at": "15:00"
    },
    {
      "name": "Основы дискретной математики",
      "type": "Практика",
      "location": "Ауд. 1428, Кронверкский пр., д.49, лит.А",
      "teacher": "Поляков Владимир Иванович",
      "start_at": "15:20",
      "end_at": "16:50"
    }
  ]
},
{
  "name": "Суббота",
  "lesson": [
    {
      "name": "Математический анализ",
      "type": "Лекция",
      "location": "zoom",
      "teacher": "Блейхер Оксана Владимировна",
      "start_at": "10:00",
      "end_at": "11:30"
    },
    {
      "name": "Программирование",
      "type": "Лабораторная",
      "location": "Ауд. 1338, Кронверкский пр., д.49, лит.А",
      "teacher": "Кулинич Ярослав Вадимович",
      "start_at": "13:30",
      "end_at": "15:00"
    },
    {
```

```
            "name": "Программирование",
            "type": "Лабораторная",
            "location": "Ауд. 1338, Кронверкский пр., д.49, лит.А",
            "teacher": "Кулинич Ярослав Вадимович",
            "start_at": "15:20",
            "end_at": "16:50"
          }
        ]
      }
    ]
  }
}
```

3.2.3 Сравнение

Результаты работы обеих программ совпадают, однако остаются сомнения, что ручной парсинг корректно обработает все возможные случаи. Использование готовых библиотек делает код более надежным и упрощает его поддержку.

3.3 Дополнительное задание №2

3.3.1 Исходный код

```python
import re
from pathlib import Path


def xml2dict(xml):
    def parse_element(element):
        result = {}
        pattern = r"<(\w+)>(.*?)</\1>|<(\w+)>(.*?)</\3>|<(\w+)/>"

        for match in re.finditer(pattern, element, re.DOTALL):
            tag = match.group(1) or match.group(3) or match.group(5)
            content = match.group(2) or match.group(4) or ''
            if re.search(r"<\w+>", content):
                content = parse_element(content)
```

```
            if tag in result:
                if not isinstance(result[tag], list):
                    result[tag] = [result[tag]]
                result[tag].append(content)
            else:
                result[tag] = content
        return result

    root_match = re.match(r"<(\w+)>(.*?)</\1>", xml, re.DOTALL)
    if root_match:
        root_tag = root_match.group(1)
        root_content = root_match.group(2)
        return {root_tag: parse_element(root_content)}
    else:
        return {}


def dict2json(data):
    def convert(value):
        if isinstance(value, dict):
            items = [f'"{key}": {convert(val)}' for key, val in value.items()]
            return "{" + ", ".join(items) + "}"
        elif isinstance(value, list):
            items = [convert(item) for item in value]
            return "[" + ", ".join(items) + "]"
        elif isinstance(value, str):
            # Экранирование кавычек внутри строк
            escaped_str = re.sub(r'(["\\])', r'\\\1', value)
            return f'"{escaped_str}"'
        elif isinstance(value, (int, float)):
            return str(value).lower()
        elif value is None:
            return "null"
        elif isinstance(value, bool):
            return "true" if value else "false"
        else:
            raise TypeError(f"Unsupported data type: {type(value)}")

    return convert(data)
```

```python
def xml2json(xml: str) -> str:
    parsed_data = xml2dict(xml)
    json_data = dict2json(parsed_data)

    return json_data


def main() -> None:
    xml_path = Path(__file__).parent.joinpath("schedule.xml")
    json_path = Path(__file__).parent.joinpath("schedule_3.json")

    xml_schedule = xml_path.read_text(encoding="utf-8")
    json_schedule = xml2json(xml_schedule)

    json_path.write_text(json_schedule, encoding="utf-8")


if __name__ == "__main__":
    main()
```

3.3.2 Результат


```
{
        "schedule": {
                "weekday": [
                        {
                                "name": "Среда",
                                "lesson": [
                                        {
                                                "name": "Информатика",
                                                "type": "Лекция",
                                                "location": "0 (усл)), ул.Ломоносова,
                                                ↪  д.9, лит. М",
                                                "teacher": "Балакшин Павел
                                                ↪  Валерьевич",
                                                "start_at": "8:20",
                                                "end_at": "9:50"
                                        },
                                        {
```

```
                        "name": "Основы профессиональной
                        ↪  деятельности",
                        "type": "Лекция",
                        "location": "0 (усл)), ул.Ломоносова,
                        ↪  д.9, лит. М",
                        "teacher": "Клименков Сергей
                        ↪  Викторович",
                        "start_at": "10:00",
                        "end_at": "11:30"
                    },
                    {
                        "name": "Основы дискретной
                        ↪  математики",
                        "type": "Лекция",
                        "location": "Ауд. 1428, Кронверкский
                        ↪  пр., д.49, лит.А",
                        "teacher": "Поляков Владимир
                        ↪  Иванович",
                        "start_at": "13:30",
                        "end_at": "15:00"
                    },
                    {
                        "name": "Основы дискретной
                        ↪  математики",
                        "type": "Практика",
                        "location": "Ауд. 1428, Кронверкский
                        ↪  пр., д.49, лит.А",
                        "teacher": "Поляков Владимир
                        ↪  Иванович",
                        "start_at": "15:20",
                        "end_at": "16:50"
                    }
                ]
            },
            {
                "name": "Суббота",
                "lesson": [
                    {
                        "name": "Математический анализ",
                        "type": "Лекция",
```

```
                "location": "zoom",
                "teacher": "Блейхер Оксана
                ↪  Владимировна",
                "start_at": "10:00",
                "end_at": "11:30"
            },
            {
                "name": "Программирование",
                "type": "Лабораторная",
                "location": "Ауд. 1338, Кронверкский
                ↪  пр., д.49, лит.А",
                "teacher": "Кулинич Ярослав
                ↪  Вадимович",
                "start_at": "13:30",
                "end_at": "15:00"
            },
            {
                "name": "Программирование",
                "type": "Лабораторная",
                "location": "Ауд. 1338, Кронверкский
                ↪  пр., д.49, лит.А",
                "teacher": "Кулинич Ярослав
                ↪  Вадимович",
                "start_at": "15:20",
                "end_at": "16:50"
            }
        ]
    }
]
}
}
```


Результаты работы ручного парсинга и парсинга с использованием регулярных выражений совпадают. Применение регулярных выражений позволило сократить объем кода, но может затруднить его понимание для тех, кто не знаком с их синтаксисом.

3.4 Дополнительное задание №3

3.4.1 Исходный код

```
import re
from abc import ABC, abstractmethod
from dataclasses import dataclass
from enum import StrEnum, auto
from pathlib import Path
from typing import Any


class JsonItemType(StrEnum):
    OBJECT = auto()
    INTEGER = auto()
    FLOAT = auto()
    BOOLEAN = auto()
    STRING = auto()


@dataclass
class IJsonItem[T](ABC):
    name: str
    value: T

    @classmethod
    def parse(cls, content: list[str]) -> "IJsonItem[T]":
        name = get_tag_name(content[0])
        value = cls.parse_value(content[1])

        return cls(name=name, value=value)

    @staticmethod
    @abstractmethod
    def parse_value(value: Any) -> T: ...

    @abstractmethod
    def to_json(self, depth: int = 1) -> str: ...


@dataclass
```

```python
class JsonIntegerNumber(IJsonItem[int]):
    @staticmethod
    def parse_value(value: Any) -> int:
        return int(value)

    def to_json(self, depth: int = 1) -> str:
        return "\t" * depth + f"\"{self.name}\": {self.value}"


@dataclass
class JsonFloatNumber(IJsonItem[float]):
    @staticmethod
    def parse_value(value: Any) -> float:
        return float(value)

    def to_json(self, depth: int = 1) -> str:
        return "\t" * depth + f"\"{self.name}\": {self.value}"


@dataclass
class JsonBoolean(IJsonItem[bool]):
    @staticmethod
    def parse_value(value: Any) -> bool:
        return value == "true"

    def to_json(self, depth: int = 1) -> str:
        return "\t" * depth + f"\"{self.name}\": {self.value}"


@dataclass
class JsonString(IJsonItem[str]):
    @staticmethod
    def parse_value(value: Any) -> str:
        return value

    def to_json(self, depth=1):
        return "\t" * depth + f"\"{self.name}\": \"{self.value}\""


@dataclass
```

```
class JsonArray(IJsonItem[list]):
    @staticmethod
    def parse_value(value: Any) -> list[Any]:
        return value

    def to_json(self, depth: int = 1) -> str:
        json_str = "\t" * depth + f"\"{self.name}\": [\n"
        array_len = len(self.value)

        for i in range(array_len):
            item: IJsonItem = self.value[i]

            if isinstance(item, JsonObject):
                tmp = item.to_json(depth + 1)
                json_str += "\t" * (depth + 1) + tmp[tmp.index("{"):]
            else:
                json_str += "\t" * (depth + 1) + str(item.value)

            if i < array_len - 1:
                json_str += ","
            json_str += "\n"

        json_str += "\t" * depth + "]"

        return json_str


@dataclass
class JsonObject(IJsonItem[list]):
    @classmethod
    def parse(cls, content: list[str]) -> "JsonObject":
        name = get_tag_name(content[0])
        value = cls.parse_value(content[1:-1])

        return cls(name=name, value=value)

    @staticmethod
    def parse_value(content: Any) -> list:
        result = []
```

```
        i = 0
        current_content = {}
        while i < len(content):
            if is_opening_tag(content[i]):
                opening_tag_name = get_tag_name(content[i])

                j = i + 1
                while j < len(content):
                    if is_closing_tag(content[j]) and get_tag_name(content[j]) ==
                    ↪  opening_tag_name:
                        break
                    j += 1

                if opening_tag_name in current_content:

                    ↪  current_content[opening_tag_name].append(parse_content(content[i:j
                    ↪  + 1]))
                else:
                    current_content[opening_tag_name] = [parse_content(content[i:j +
                    ↪  1])]
                i = j
            i += 1

        for key, value in current_content.items():
            if len(current_content[key]) > 1:
                result.append(JsonArray(name=key, value=value))
            else:
                result.append(value[0])

        return result

    def to_json(self, depth: int = 1) -> str:
        json_str = "\t" * depth + f"\"{self.name}\": " + "{\n"
        content_len = len(self.value)

        for i in range(content_len):
            json_str += self.value[i].to_json(depth + 1)
            if i < content_len - 1:
                json_str += ","
            json_str += "\n"
```

```
        json_str += "\t" * depth + "}"

        return json_str


def get_tag_name(tag: str) -> str:
    return re.match(r"</?(.*)>", tag).group(1)


def is_opening_tag(tag: str) -> bool:
    return is_tag(tag) and tag[1] != "/"


def is_closing_tag(tag: str) -> bool:
    return is_tag(tag) and tag[1] == "/"


def is_tag(tag: str) -> bool:
    return tag[0] == "<" and tag[-1] == ">"


def get_content_type(content: list[str]) -> JsonItemType:
    if len(content) != 3:
        return JsonItemType.OBJECT

    value = content[1]
    if value in ("true", "false"):
        return JsonItemType.BOOLEAN
    elif re.match(r"^-?\d+$", value):
        return JsonItemType.INTEGER
    elif re.match(r"^-?\d+\.\d+$", value):
        return JsonItemType.FLOAT
    else:
        return JsonItemType.STRING


def parse_content(content: list[str]) -> IJsonItem:
    match get_content_type(content):
        case JsonItemType.INTEGER:
```

```
            return JsonIntegerNumber.parse(content)
        case JsonItemType.FLOAT:
            return JsonFloatNumber.parse(content)
        case JsonItemType.BOOLEAN:
            return JsonBoolean.parse(content)
        case JsonItemType.STRING:
            return JsonString.parse(content)
        case JsonItemType.OBJECT:
            return JsonObject.parse(content)


def split_into_tokens(xml: str) -> list[str]:
    return re.findall(r"</?[^<>/]*>|(?!\s)[^<>/]+?(?=\s*</)", xml)


def xml2json(xml: str) -> str:
    tokens = split_into_tokens(xml)
    json_object = JsonObject.parse(tokens).to_json()

    return "{\n" + json_object + "\n}"


def main() -> None:
    xml_path = Path(__file__).parent.joinpath("schedule.xml")
    json_path = Path(__file__).parent.joinpath("schedule_4.json")

    xml_schedule = xml_path.read_text(encoding="utf-8")
    json_schedule = xml2json(xml_schedule)

    json_path.write_text(json_schedule, encoding="utf-8")

if __name__ == "__main__":
    main()
```

3.4.2 Результат

```
{
        "schedule": {
                "weekday": [
                        {
                                "name": "Среда",
```

```
"lesson": [
        {
            "name": "Информатика",
            "type": "Лекция",
            "location": "0 (усл)), ул.Ломоносова,
            ↪  д.9, лит. М",
            "teacher": "Балакшин Павел
            ↪  Валерьевич",
            "start_at": "8:20",
            "end_at": "9:50"
        },
        {
            "name": "Основы профессиональной
            ↪  деятельности",
            "type": "Лекция",
            "location": "0 (усл)), ул.Ломоносова,
            ↪  д.9, лит. М",
            "teacher": "Клименков Сергей
            ↪  Викторович",
            "start_at": "10:00",
            "end_at": "11:30"
        },
        {
            "name": "Основы дискретной
            ↪  математики",
            "type": "Лекция",
            "location": "Ауд. 1428, Кронверкский
            ↪  пр., д.49, лит.А",
            "teacher": "Поляков Владимир
            ↪  Иванович",
            "start_at": "13:30",
            "end_at": "15:00"
        },
        {
            "name": "Основы дискретной
            ↪  математики",
            "type": "Практика",
            "location": "Ауд. 1428, Кронверкский
            ↪  пр., д.49, лит.А",
```

```
                        "teacher": "Поляков Владимир
                        ↪  Иванович",
                        "start_at": "15:20",
                        "end_at": "16:50"
                    }
                ]
            },
            {
                "name": "Суббота",
                "lesson": [
                    {
                        "name": "Математический анализ",
                        "type": "Лекция",
                        "location": "zoom",
                        "teacher": "Блейхер Оксана
                        ↪  Владимировна",
                        "start_at": "10:00",
                        "end_at": "11:30"
                    },
                    {
                        "name": "Программирование",
                        "type": "Лабораторная",
                        "location": "Ауд. 1338, Кронверкский
                        ↪  пр., д.49, лит.А",
                        "teacher": "Кулинич Ярослав
                        ↪  Вадимович",
                        "start_at": "13:30",
                        "end_at": "15:00"
                    },
                    {
                        "name": "Программирование",
                        "type": "Лабораторная",
                        "location": "Ауд. 1338, Кронверкский
                        ↪  пр., д.49, лит.А",
                        "teacher": "Кулинич Ярослав
                        ↪  Вадимович",
                        "start_at": "15:20",
                        "end_at": "16:50"
                    }
                ]
```

```
                    }
                ]
        }
}
```

3.4.3 Сравнение

Результаты парсинга с использованием формальных грамматик совпали с результатами, полученными ранее. Применение формальных грамматик привело к увеличению объема кода, однако это дало существенные преимущества. Теперь появилась возможность детально управлять процессом парсинга для каждого отдельного компонента, что повышает гибкость и упрощает внесение изменений или расширение функциональности в будущем.

3.5 Дополнительное задание №4

3.5.1 Исходный код

```python
import time
from collections.abc import Callable
from functools import partial
from pathlib import Path

from task_1 import xml2json as manual_converter
from task_2 import xml2json as lib_converter
from task_3 import xml2json as regexp_converter
from task_4 import xml2json as gram_converter

REPEATS = 100


def benchmark(func: Callable[[], str]) -> float:
    start = time.perf_counter()

    for i in range(REPEATS):
        func()
```

```
    return time.perf_counter() - start


def main() -> None:
    xml_schedule =
    ↪  Path(__file__).parent.joinpath("schedule.xml").read_text(encoding="utf-8")

    print("Без использования библиотек:", benchmark(partial(manual_converter,
    ↪  xml_schedule)))
    print("С использование библиотек:", benchmark(partial(lib_converter,
    ↪  xml_schedule)))
    print("С использованием регулярных выражений:",
    ↪  benchmark(partial(regexp_converter, xml_schedule)))
    print("С использованием формальных грамматик:", benchmark(partial(gram_converter,
    ↪  xml_schedule)))


if __name__ == "__main__":
    main()
```

3.5.2 Результат

- Без использования библиотек: 0.006938100035768002
- С использование библиотек: 0.012030399986542761
- С использованием регулярных выражений: 0.014387500006705523
- С использованием формальных грамматик: 0.034521599999869764

3.5.3 Сравнение

Ручной парсинг оказался самым быстрым (*0.0069 с*), но менее универсальным. Использование библиотек (*0.012 с*) немного увеличило время, обеспечив удобство и надёжность. Регулярные выражения (*0.014 с*) добавили гибкость, но усложнили понимание. Формальные грамматики оказались самыми медленными (*0.034 с*), однако предоставили максимальный контроль и расширяемость.

3.6 Дополнительное задание №5

3.6.1 Исходный код

```
import csv
from io import StringIO
from pathlib import Path

from xml.etree import ElementTree


def xml2csv(xml: str) -> str:
    root = ElementTree.fromstringlist(xml)
    io = StringIO()
    writer = csv.writer(io)

    headers = [
        "weekday",
        "lesson_name",
        "lesson_type",
        "location",
        "teacher",
        "start_at",
        "end_at"
    ]
    writer.writerow(headers)

    for weekday in root.iter("weekday"):
        day_name = weekday.find("name").text

        for lesson in weekday.iter("lesson"):
            lesson_name = lesson.find('name').text
            lesson_type = lesson.find('type').text
            location = lesson.find('location').text
            teacher = lesson.find('teacher').text
            start_at = lesson.find('start_at').text
            end_at = lesson.find('end_at').text

            writer.writerow([
                day_name,
                lesson_name,
```

```
                lesson_type,
                location,
                teacher,
                start_at,
                end_at
            ])


    return io.getvalue()



def main() -> None:
    xml_path = Path(__file__).parent.joinpath("schedule.xml")
    csv_path = Path(__file__).parent.joinpath("schedule.csv")

    xml_schedule = xml_path.read_text(encoding="utf-8")
    csv_schedule = xml2csv(xml_schedule)

    csv_path.write_text(csv_schedule, encoding="utf-8", newline="")


if __name__ == "__main__":
    main()
```

3.6.2 Результат

```
weekday,lesson_name,lesson_type,location,teacher,start_at,end_at
Среда,Информатика,Лекция,"Ауд. Актовый зал (1216/0 (усл)), ул.Ломоносова, д.9, лит.
↪  М",Балакшин Павел Валерьевич,8:20,9:50
Среда,Основы профессиональной деятельности,Лекция,"Ауд. Актовый зал (1216/0 (усл)),
↪  ул.Ломоносова, д.9, лит. М",Клименков Сергей Викторович,10:00,11:30
Среда,Основы дискретной математики,Лекция,"Ауд. 1428, Кронверкский пр., д.49,
↪  лит.А",Поляков Владимир Иванович,13:30,15:00
Среда,Основы дискретной математики,Практика,"Ауд. 1428, Кронверкский пр., д.49,
↪  лит.А",Поляков Владимир Иванович,15:20,16:50
Суббота,Математический анализ,Лекция,zoom,Блейхер Оксана Владимировна,10:00,11:30
Суббота,Программирование,Лабораторная,"Ауд. 1338, Кронверкский пр., д.49,
↪  лит.А",Кулинич Ярослав Вадимович,13:30,15:00
Суббота,Программирование,Лабораторная,"Ауд. 1338, Кронверкский пр., д.49,
↪  лит.А",Кулинич Ярослав Вадимович,15:20,16:50
```

3.7 Сравнение

Изначальный JSON-файл был преобразован в CSV с использованием разделителя ;. Формат CSV удобен для представления данных в табличной форме, что облегчает их чтение, анализ и обработку в большинстве стандартных инструментов для работы с таблицами

4 Вывод

В ходе лабораторной работы был приобретён практический опыт преобразования форматов данных с использованием как готовых библиотек, так и самописных алгоритмов. Это позволило сравнить их производительность и удобство в различных сценариях.

5 Список использованных источников

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

2. Грошев А.С. Г89 Информатика: Учебник для вузов / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2010. -470с.