«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники (ФПИиКТ)

Лабораторная работа

Исследование протоколов, форматов обмена информацией и языков разметки документов

Вариант 32

Выполнил

Григорьев Даниил Александрович

Группа 3116

Принял В.В. Пономарев

Санкт-Петербург 2024

Содержание

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.

Основные этапы вычисления

Основное задание:

```
def parse_input_file():
    file = open("Informatics/fourth_lab/main_task/input.json", "r",
encoding="utf-8")
    yaml_data = file.read()
        # Заменяем ключи на YAML-формат (отступы и двоеточия)
    yaml_data = yaml_data.replace("{", "").replace("}", "")
    yaml_data = yaml_data.replace("[", "").replace("]", "")
    yaml_data = yaml_data.replace(",", "")
    yaml_data = yaml_data.replace(":", ":\n  ")
    yaml_data = yaml_data.replace("\"", "")  # Убираем кавычки
    yaml_data = yaml_data.replace("\n\n", "\n")  # Убираем лишние пустые
строки

    lines = yaml_data.split("\n")
    # print(lines)
    yaml_lines = []
    for line in lines:
        if line.strip() == "":
            continue
        indent_level = line.find(line.strip())  # находим уровень отступа
        yaml_lines.append("  " * indent_level + line.strip())
    final_var = ""
    reserved_words = ["date", "first_day", "second_day", "subject", "lesson",
"lecture", "time", "room", "place", "schedule"]
    print(yaml_lines)
    for line in yaml_lines:

        exist_flag = False
        for i in reserved_words:
            if i in line:
                final_var += "\n" + line
                exist_flag = True
                break
        if exist_flag == False:
            final_var += " " + line.strip()




    output_file = open("Informatics/fourth_lab/main_task/output.yaml", "w",
encoding="utf-8")
    output_file.write(final_var)
    output_file.close()


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

Пример работы:

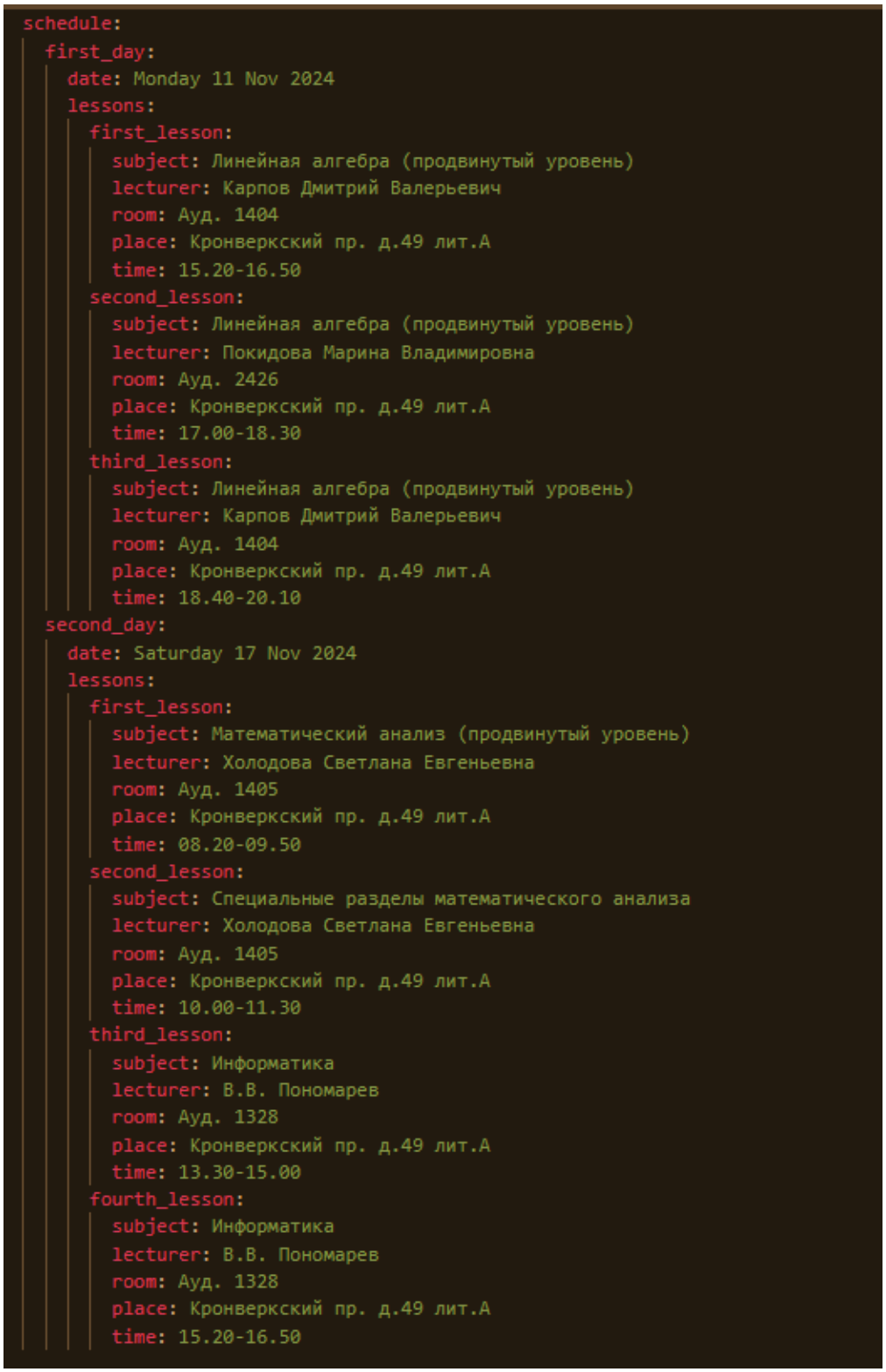
Входной файл input.json:

Первая половина файла

Вторая половина файла


```
        "second_day":
        {
            "date": "Saturday 17 Nov 2024",
            "lessons":
            {
                "first_lesson":
                {
                    "subject": "Математический анализ (продвинутый уровень)",
                    "lecturer": "Холодова Светлана Евгеньевна",
                    "room": "Ауд. 1405",
                    "place": "Кронверкский пр. д.49 лит.А",
                    "time": "08.20-09.50"
                },
                "second_lesson":
                {
                    "subject": "Специальные разделы математического анализа",
                    "lecturer": "Холодова Светлана Евгеньевна",
                    "room": "Ауд. 1405",
                    "place": "Кронверкский пр. д.49 лит.А",
                    "time": "10.00-11.30"
                },
                "third_lesson":
                {
                    "subject": "Информатика",
                    "lecturer": "В.В. Пономарев",
                    "room": "Ауд. 1328",
                    "place": "Кронверкский пр. д.49 лит.А",
                    "time": "13.30-15.00"
                },
                "fourth_lesson":
                {
                    "subject": "Информатика",
                    "lecturer": "В.В. Пономарев",
                    "room": "Ауд. 1328",
                    "place": "Кронверкский пр. д.49 лит.А",
                    "time": "15.20-16.50"
                }
            }
        }
    }
}
```

Выходной файл после работы алгоритма:



```yaml
schedule:
  first_day:
    date: Monday 11 Nov 2024
    lessons:
      first_lesson:
        subject: Линейная алгебра (продвинутый уровень)
        lecturer: Карпов Дмитрий Валерьевич
        room: Ауд. 1404
        place: Кронверкский пр. д.49 лит.А
        time: 15.20-16.50
      second_lesson:
        subject: Линейная алгебра (продвинутый уровень)
        lecturer: Покидова Марина Владимировна
        room: Ауд. 2426
        place: Кронверкский пр. д.49 лит.А
        time: 17.00-18.30
      third_lesson:
        subject: Линейная алгебра (продвинутый уровень)
        lecturer: Карпов Дмитрий Валерьевич
        room: Ауд. 1404
        place: Кронверкский пр. д.49 лит.А
        time: 18.40-20.10
  second_day:
    date: Saturday 17 Nov 2024
    lessons:
      first_lesson:
        subject: Математический анализ (продвинутый уровень)
        lecturer: Холодова Светлана Евгеньевна
        room: Ауд. 1405
        place: Кронверкский пр. д.49 лит.А
        time: 08.20-09.50
      second_lesson:
        subject: Специальные разделы математического анализа
        lecturer: Холодова Светлана Евгеньевна
        room: Ауд. 1405
        place: Кронверкский пр. д.49 лит.А
        time: 10.00-11.30
      third_lesson:
        subject: Информатика
        lecturer: В.В. Пономарев
        room: Ауд. 1328
        place: Кронверкский пр. д.49 лит.А
        time: 13.30-15.00
      fourth_lesson:
        subject: Информатика
        lecturer: В.В. Пономарев
        room: Ауд. 1328
        place: Кронверкский пр. д.49 лит.А
        time: 15.20-16.50
```

Дополнительное задание №1

Код алгоритма:

```
import json
import yaml # используем pyyaml

input_file = open("Informatics/fourth_lab/first_option_task/input.json", "r",
encoding="utf-8").read()

json_data = json.loads(input_file) # парсим json из строки
yaml_data = yaml.dump(json_data, default_flow_style=False,
allow_unicode="True") # конвертируем json в yaml

output_file = open("Informatics/fourth_lab/first_option_task/output.yaml",
"w", encoding="utf-8")
output_file.write(yaml_data)
output_file.close()
```

Входные файлы для обязательного задания и дополнительного задания №1 ничем не отличаются. В выходных файлах глобальных отличий нет, единственное отличие – в выходном коде после выполнения алгоритма дополнительного задания №1 “ключи” сортируются по названию в алфавитном порядке.

Дополнительное задание №2

Код алгоритма:

```
import re
def parse_input_file():
    file = open("Informatics/fourth_lab/second_option_task/input.json", "r",
encoding="utf-8")
    yaml_data = file.read()
    yaml_data = re.sub(r'["}],', "", yaml_data)
    yaml_data = re.sub(r'[{}"]', "", yaml_data)


    lines = yaml_data.split("\n")
    # print(lines)
    yaml_lines = []
    for line in lines:
        if line.strip() == "":
            continue
        indent_level = line.find(line.strip())  # находим уровень отступа
        yaml_lines.append("  " * (indent_level-1) + line.strip())


    output_file =
open("Informatics/fourth_lab/second_option_task/output.yaml", "w",
encoding="utf-8")
    output_file.write("\n".join(yaml_lines))
    output_file.close()

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

Входные файлы для обязательного задания и дополнительного задания №2 ничем не отличаются. В выходных файлах обязательного и дополнительного задания №2 отличий нет.

Дополнительное задание №3

Код алгоритма:

```
class JSONParser: # парсер json формата из строки: строка -> json
    def __init__(self, text):
        self.text = text
        self.index = 0

    def parse(self): #старт парсинга: пропускаем пробелы -> обрабатываем не
пробельный символ -> пропускаем пробелы -> возвращаем json
        self.skip_whitespace()
        value = self.parse_value()
        self.skip_whitespace()
        return value

    def parse_value(self): # обрабатываем не пробельный символ (если пробел,
то пропускаем и работаем с не пробельным символом)
        self.skip_whitespace()
        char = self.current_char()
        if char == '"':
            return self.parse_string()
        elif char == '{':
            return self.parse_object()
        elif char == '[':
            return self.parse_array()
        elif char.isdigit() or char == '-':
            return self.parse_number()
        elif self.text[self.index:self.index+4].lower() == "true":
            self.index += 4
            return True
        elif self.text[self.index:self.index+5].lower() == "false":
            self.index += 5
            return False
        elif self.text[self.index:self.index+4].lower() == "none":
            self.index += 4
            return None
        else:
            print("Че ты сюда вписал вообще, кроме
цифр/чисел/строк/словарей/списков/true/false/none ничего не принимается")

    def parse_object(self): # работаем с объектом (словарь)
        obj = {}
        self.index += 1  # Skip '{'
        self.skip_whitespace()
        while self.current_char() != '}':
            key = self.parse_string()
            self.skip_whitespace()
            if self.current_char() != ':':
                print("Переделывай свой json: ожидался символ : после ключа
словаря")
            self.index += 1  # Skip ':'
            self.skip_whitespace()
            value = self.parse_value()
            obj[key] = value
            self.skip_whitespace()
            if self.current_char() == ',':
                self.index += 1  # Skip ','
                self.skip_whitespace()
            elif self.current_char() != '}':
                print("Переделывай свой json: ожидались символы , или }")
        self.index += 1  # Skip '}'
```

```
        return obj

    def parse_array(self): # работаем со списком
        arr = []
        self.index += 1  # Skip '['
        self.skip_whitespace()
        while self.current_char() != ']':
            value = self.parse_value()
            arr.append(value)
            self.skip_whitespace()
            if self.current_char() == ',':
                self.index += 1  # Skip ','
                self.skip_whitespace()
            elif self.current_char() != ']':
                print("Переделывай свой json, он выполнен не по формату:
нужны , или ]")

        self.index += 1  # Скипаем ']'
        return arr

    def parse_string(self): # работаем со строкой
        self.index += 1  # Скипаем открытие строки '"'
        start = self.index
        while self.current_char() != '"':
            if self.current_char() == '\\':
                self.index += 2  # Скипаем символ \
            else:
                self.index += 1
        result = self.text[start:self.index]
        self.index += 1  # Скипаем закрытие строки '"'
        return result

    def parse_number(self): # работаем с числами
        start = self.index
        if self.current_char() == '-':
            self.index += 1
        while self.current_char().isdigit():
            self.index += 1
        if self.current_char() == '.':
            self.index += 1
            while self.current_char().isdigit():
                self.index += 1
        if self.current_char() in 'eE':
            self.index += 1
            if self.current_char() in '+-':
                self.index += 1
            while self.current_char().isdigit():
                self.index += 1
        return float(self.text[start:self.index])

    def current_char(self): # текущий символ
        if self.index < len(self.text):
            return self.text[self.index]
        return ''

    def skip_whitespace(self): # скипаем пробелы (прибавляем к
рассматриваемому индексу единицу)
        while self.index < len(self.text) and
self.text[self.index].isspace():
            self.index += 1

def json_to_yaml(data, indent = 0, tire = 0): # конвертер json в yaml
    yaml = ""
```

```
    if isinstance(data, dict):
        for key, value in data.items():

            if tire != 1 or key != list(data.keys())[0]:
                if isinstance(value, dict):
                    yaml += "\n"
                elif isinstance(value, list):
                    yaml += "\n"
                elif isinstance(value, str):
                    yaml += "\n"
                else:
                    yaml += "\n"
                yaml += indent * " " + key + ":" + json_to_yaml(value,
indent+2,0)

            elif key == list(data.keys())[0]and tire == 1:

                yaml +=  key + ":" + json_to_yaml(value, indent+2,0)



    elif isinstance(data, list):
        for value in data:
            yaml += "\n" + (indent-2)  * " " + "- " + json_to_yaml(value,
indent, 1)

    elif isinstance(data, str):
        yaml += (f' "{data.strip()}"')
    else:
        yaml += " " + str(data).strip()

    return yaml



# Чтение JSON-файла, разбор и преобразование в YAML
def main():
    with open("Informatics/fourth_lab/third_option_task/example0.json", "r",
encoding="UTF-8") as file:
        json_text = file.read()
    parser = JSONParser(json_text)
    parsed_data = parser.parse()
    yaml_output = json_to_yaml(parsed_data)
    yaml_output = yaml_output.lstrip("\n")
    with open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/third_option_task/output.yaml", "w",
encoding = "UTF-8") as file:
        file.write(yaml_output)
    print("JSON успешно конвертирован в YAML и сохранён в 'output.yaml'.")


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

Дополнительное задание №4

Код алгоритма:

```
import time
import json
import yaml
import os
import re


def main_task():
    input_file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/input.json", "r",
encoding="utf-8")
    output_file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/output.yaml",
"w", encoding="utf-8")
    yaml_data = input_file.read()
        # Заменяем метасимволы для приведения json к yaml формату
    yaml_data = yaml_data.replace('",', "").replace('},', "").replace('"',
"")
    yaml_data = yaml_data.replace("{", "").replace("}", "")
    yaml_data = yaml_data.replace("[", "").replace("]", "")


    lines = yaml_data.split("\n") # список строк, с которым будем работать
    yaml_lines = [] # результирующий список строк
    first = -1
    for line in lines:
        if line.strip() == "":
            continue
        elif first == -1:
            first = line.find(line.strip())
        indent_level = line.find(line.strip())  # находим уровень отступа
путем нахождения индекса первой встречи ключа в строке line
        yaml_lines.append("  " * (indent_level-first) +
line.strip().strip("\t"))


    output_file.write("\n".join(yaml_lines))
    output_file.close()



def first_option():
    input_file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/input.json", "r",
encoding="utf-8")
    output_file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/output.yaml",
"w", encoding="utf-8")
    input_file = input_file.read()
    json_data = json.loads(input_file) # парсим json из строки
    yaml_data = yaml.dump(json_data, default_flow_style=False,
allow_unicode="True") # конвертируем json в yaml
    output_file.write(yaml_data)
    output_file.close()

def second_option():
```

```
    file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/input.json", "r",
encoding="utf-8")
    yaml_data = file.read()
    yaml_data = re.sub(r'["}],', "", yaml_data)
    yaml_data = re.sub(r'[{}"]', "", yaml_data)
    lines = yaml_data.split("\n")

    yaml_lines = []
    for line in lines:
        if line.strip() == "":
            continue
        indent_level = line.find(line.strip())  # находим уровень отступа
        yaml_lines.append("  " * (indent_level-1) + line.strip())


    output_file = open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/output.yaml",
"w", encoding="utf-8")
    output_file.write("\n".join(yaml_lines))
    output_file.close()

#######
class JSONParser: # парсер json формата из строки: строка -> json
    def __init__(self, text):
        self.text = text
        self.index = 0

    def parse(self): #старт парсинга: пропускаем пробелы -> обрабатываем не
пробельный символ -> пропускаем пробелы -> возвращаем json
        self.skip_whitespace()
        value = self.parse_value()
        self.skip_whitespace()
        return value

    def parse_value(self): # обрабатываем не пробельный символ (если пробел,
то пропускаем и работаем с не пробельным символом)
        self.skip_whitespace()
        char = self.current_char()
        if char == '"':
            return self.parse_string()
        elif char == '{':
            return self.parse_object()
        elif char == '[':
            return self.parse_array()
        elif char.isdigit() or char == '-':
            return self.parse_number()
        elif self.text[self.index:self.index+4].lower() == "true":
            self.index += 4
            return True
        elif self.text[self.index:self.index+5].lower() == "false":
            self.index += 5
            return False
        elif self.text[self.index:self.index+4].lower() == "none":
            self.index += 4
            return None
        else:
            print("Че ты сюда вписал вообще, кроме
цифр/чисел/строк/словарей/списков/true/false/none ничего не принимается")

    def parse_object(self): # работаем с объектом (словарь)
        obj = {}
        self.index += 1  # Skip '{'
```

```
        self.skip_whitespace()
        while self.current_char() != '}':
            key = self.parse_string()
            self.skip_whitespace()
            if self.current_char() != ':':
                print("Переделывай свой json: ожидался символ : после ключа
словаря")
            self.index += 1  # Skip ':'
            self.skip_whitespace()
            value = self.parse_value()
            obj[key] = value
            self.skip_whitespace()
            if self.current_char() == ',':
                self.index += 1  # Skip ','
                self.skip_whitespace()
            elif self.current_char() != '}':
                print("Переделывай свой json: ожидались символы , или }")
        self.index += 1  # Skip '}'
        return obj

    def parse_array(self): # работаем со списком
        arr = []
        self.index += 1  # Skip '['
        self.skip_whitespace()
        while self.current_char() != ']':
            value = self.parse_value()
            arr.append(value)
            self.skip_whitespace()
            if self.current_char() == ',':
                self.index += 1  # Skip ','
                self.skip_whitespace()
            elif self.current_char() != ']':
                print("Переделывай свой json, он выполнен не по формату:
нужны , или ]")

        self.index += 1  # Скипаем ']'
        return arr

    def parse_string(self): # работаем со строкой
        self.index += 1  # Скипаем открытие строки '"'
        start = self.index
        while self.current_char() != '"':
            if self.current_char() == '\\':
                self.index += 2  # Скипаем символ \
            else:
                self.index += 1
        result = self.text[start:self.index]
        self.index += 1  # Скипаем закрытие строки '"'
        return result

    def parse_number(self): # работаем с числами
        start = self.index
        if self.current_char() == '-':
            self.index += 1
        while self.current_char().isdigit():
            self.index += 1
        if self.current_char() == '.':
            self.index += 1
            while self.current_char().isdigit():
                self.index += 1
        if self.current_char() in 'eE':
            self.index += 1
            if self.current_char() in '+-':
```

```
                self.index += 1
            while self.current_char().isdigit():
                self.index += 1
        return float(self.text[start:self.index])

    def current_char(self): # текущий символ
        if self.index < len(self.text):
            return self.text[self.index]
        return ''

    def skip_whitespace(self): # скипаем пробелы (прибавляем к
рассматриваемому индексу единицу)
        while self.index < len(self.text) and
self.text[self.index].isspace():
            self.index += 1

def json_to_yaml(data, indent = 0, tire = 0): # конвертер json в yaml
    yaml = ""

    if isinstance(data, dict):
        for key, value in data.items():

            if tire != 1 or key != list(data.keys())[0]:
                if isinstance(value, dict):
                    yaml += "\n"
                elif isinstance(value, list):
                    yaml += "\n"
                elif isinstance(value, str):
                    yaml += "\n"
                else:
                    yaml += "\n"
                yaml += indent * " " + key + ":" + json_to_yaml(value,
indent+2,0)

            elif key == list(data.keys())[0]and tire == 1:

                yaml +=  key + ":" + json_to_yaml(value, indent+2,0)



    elif isinstance(data, list):
        for value in data:
            yaml += "\n" + (indent-2)  * " " + "- " + json_to_yaml(value,
indent, 1)

    elif isinstance(data, str):
        yaml += (f' "{data.strip()}"')
    else:
        yaml += " " + str(data).strip()

    return yaml




def third_option_task():
    with open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/input.json", "r",
encoding="UTF-8") as file:
        json_text = file.read()
    parser = JSONParser(json_text)
    parsed_data = parser.parse()
    yaml_output = json_to_yaml(parsed_data)
```

```
    yaml_output = yaml_output.lstrip("\n")
    with open("C:/Users/danie/OneDrive/Desktop/main
folder/ITMO_files/Informatics/fourth_lab/fourth_option_task/output.yaml",
"w", encoding = "UTF-8") as file:
        file.write(yaml_output)
####




if __name__ == "__main__":



    start_time = time.time()
    for i in range(100):
        main_task()
    end_time = time.time()
    print(f"Время выполнения файла основного задания:{end_time-
start_time:.6f} секунд")

    start_time = time.time()
    for i in range(100):
        first_option()
    end_time = time.time()
    print(f"Время выполнения файла доп задачи 1: {end_time-start_time:.6f}
секунд")

    start_time = time.time()
    for i in range(100):
        second_option()
    end_time = time.time()
    print(f"Время выполнения доп задачи 2: {end_time-start_time:.6f} секунд")

    start_time = time.time()
    for i in range(100):
        third_option_task()
    end_time = time.time()
    print(f"Время выполнения доп задачи 3: {end_time-start_time:.6f} секунд")
```

Заключение

Я научился парсить json и конвертировать его в yaml

Вопросы

Форма Бэкуса-Наура – это формальная система описания синтаксиса, в которой одни синтаксические группы определяются через другие.

<определяемый символ> ::= <посл1> | <посл2>

Принципы организации формальных грамматик

Такие единицы есть:

Терминал – неизменяемый объект, существующий в словах языка

Нетерминал – объект, обозначающий сущность языка (формула, команда …)

Начальный объект

Правила (продукции):

<S> -> <…> -> <…> …

Формальная грамматика – способ описания формального языка

Сравнение: JSON, YAML и XML

Характеристика	JSON	YAML	XML
Читаемость	Хорошо, но более "механично"	Очень хорошая для человека	Средняя, громоздкие теги
Компактность	Очень компактный	Более компактный, чем XML	Более громоздкий
Поддержка комментариев	Нет	Да	Да
Типы данных	Ограниченные (строки, числа, массивы, объекты, null)	Поддерживает сложные структуры (ссылки, многоуровневые словари)	Широкий спектр, поддерживает атрибуты и структуры
Парсинг и генерация	Легче всего	Сложнее, чем JSON	Сложнее, чем JSON и YAML
Применение	Веб-приложения, API, мобильные приложения	Конфигурационные файлы, DevOps	Старые системы, сложные структуры данных
Стандарты и схемы	Нет (но поддерживаются внешние схемы, например, JSON Schema)	Нет схем, можно расширять	Есть (например, XML Schema)
Преимущества	Простой, популярный, компактный	Читаемость, поддержка ссылок, комментариев	Очень гибкий, расширяемый
Недостатки	Не поддерживает комментарии, менее человекочитаем, чем YAML	Сложность парсинга, чувствителен к отступам	Сложность и громоздкость, низкая компактность

Markup – общее название языков разметки (HTML, latex, XML)

Markdown – упрощенный язык разметки для быстрого форматирования текста в HTML

Protobuf – протокол сериализации данных, предложенный Google, как альтернатива xml. Работает быстрее, поскольку осуществляется передача бинарных данных, оптимизированных под минимальный размер сообщения[

CSV – comma separated values – отделяются запятыми

TSV – tab separated values – отделяются табуляцией

Разнообразие требований к данным, оптимизация, гибкость, упрощение работы с данными

< - начало тега, > - конец тега

Сериализация – процесс перевода данных в битовую последовательность

- коммент

Viber: JSON
WhatsApp: JSON
Telegram: TL (Type Language, собственный ML телеграма)/JSON
VK: JSON
Twitter: JSON

SVG — это формат и стандарт для описания двухмерной векторной графики с помощью XML. Scalable vector graphics – масштабируемая векторная графика

Текст ссылки

Объекты, массивы, строки, числа, null, true/false

Список использованных источников

1. [Балакшин, П. В. Информатика / П. В. Балакшин, В. В. Соснин, Е. А. Машина. – СПБ : Университет ИТМО, 2020. – 143 с.]
2. [ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления» : дата введения 2017 25 09. – Москва : Стандартинформ, 2017. – 32 с.]
3. ГОСТ 7.1 — 2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления» : дата введения 01.07.2004. – Москва : ИПК Издательство стандартов, 2003. – 57 с.