# Санкт-Петербургский Национальный Исследовательский

# Университет ИТМО

# Факультет программной инженерии и компьютерной техники

## Лабораторная работа №4

## информатикa

## Вариант : 465716 (16)

Выполнил:

**Захра дарабзадех**

Студент группы: P3130

Преподаватель:

Stepen



Санкт-Петербург 2024

oглавление

[Задание 3](#_Toc182793514)

[Основные этапы выполнения 5](#_Toc182793515)

[1. Обязательное задание 5](#_Toc182793516)

[Дополнительное задание №1 6](#_Toc182793517)

[Дополнительное задание №2 7](#_Toc182793518)

[Дополнительное задание №3 8](#_Toc182793519)

[Дополнительное задание №4 9](#_Toc182793520)

[Дополнительное задание №5 11](#_Toc182793521)

[Вывод 12](#_Toc182793522)

# Задание

1. Определить номер варианта как остаток деления на 36 последних двух цифр своего идентификационного номера в ISU: например, 1255**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).
   * 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. 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).

а) 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.

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

1. 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.
2. 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.
3. 14.Проверить, что все пункты задания выполнены и выполнены верно.
4. 15.Написать отчёт о проделанной работе.
5. 16.Подготовиться к устным вопросам на защите.

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | YAML | XML | Среда, суббота |

# Основные этапы выполнения

### Обязательное задание

def yaml\_to\_xml(yaml\_content):

    def process\_line(line, indent\_level):

        stripped\_line = line.strip()

        if ": " in stripped\_line:  # Ключ-значение

            key, value = stripped\_line.split(": ", 1)

            return f"{'  ' \* indent\_level}<{key}>{value}</{key}>\n"

        elif ":" in stripped\_line:  # Ключ (начало вложенной структуры)

            key = stripped\_line[:-1]

            return f"{'  ' \* indent\_level}<{key}>\n"

        else:

            return ""

    lines = yaml\_content.splitlines()

    xml\_output = "<root>\n"

    indent\_level = 1

    stack = []

    for line in lines:

        if not line.strip():  # Пропуск пустых строк

            continue

        current\_indent = len(line) - len(line.lstrip())

        # Закрытие тегов при уменьшении вложенности

        while stack and stack[-1][1] >= current\_indent:

            last\_key = stack.pop()[0]

            xml\_output += f"{'  ' \* indent\_level}</{last\_key}>\n"

            indent\_level -= 1

        # Обработка строки

        if ": " in line.strip():

            key, \_ = line.strip().split(": ", 1)

            xml\_output += process\_line(line, indent\_level)

        elif ":" in line.strip():

            key = line.strip()[:-1]

            xml\_output += process\_line(line, indent\_level)

            stack.append((key, current\_indent))

            indent\_level += 1

    # Закрытие оставшихся открытых тегов

    while stack:

        last\_key = stack.pop()[0]

        xml\_output += f"{'  ' \* indent\_level}</{last\_key}>\n"

        indent\_level -= 1

    xml\_output += "</root>"

    return xml\_output

# Чтение данных из файла YAML

yaml\_file\_path = "C:/Users/just DO it !/Desktop/lesson/inf/lab4/schedule.yaml"

with open(yaml\_file\_path, "r") as yaml\_file:

    yaml\_content = yaml\_file.read()

    # Конвертация в XML

xml\_result = yaml\_to\_xml(yaml\_content)

# Сохранение результата в файл XML

xml\_file\_path = "result.xml"

with open(xml\_file\_path, "w") as xml\_file:

    xml\_file.write(xml\_result)

....

input yaml :

schedule:

  day:

    - name: "Wednesday"

      subject:

        - name: "Informatics"

          type: "Lecture"

          time: "08:20-09:50"

          instructor: "Balakshin Pavel Valeryevich"

          room: "1216"

          address: "Lomonosova Street, Building 9"

        - name: "Fundamentals of Professional Activity"

          type: "Lecture"

          time: "10:00-11:30"

          instructor: "Klimenkov Sergey Viktorovich"

          room: "1216"

          address: "Lomonosova Street, Building 9"

    - name: "Saturday"

      subject:

        - name: "Linear Algebra"

          type: "Lab Work"

          time: "08:20 - 09:50"

          instructor: "Isaeva Tatyana Timofeevna"

          room: "2113"

          address: "Kronversky Prospect 49"

        - name: "Fundamentals of Professional Activity"

          type: "Lab Work"

          time: "09:50 - 11:30"

          instructor: "Oblyashevsky Sevastian Alexandrovich"

          room: "2308"

          address: "Kronversky Prospect 49"

… output xml :

<root>

  <schedule>

    <day>

      <- name>"Wednesday"</- name>

      <subject>

        <- name>"Informatics"</- name>

        <type>"Lecture"</type>

        <time>"08:20-09:50"</time>

        <instructor>"Balakshin Pavel Valeryevich"</instructor>

        <room>"1216"</room>

        <address>"Lomonosova Street, Building 9"</address>

        <- name>"Fundamentals of Professional Activity"</- name>

        <type>"Lecture"</type>

        <time>"10:00-11:30"</time>

        <instructor>"Klimenkov Sergey Viktorovich"</instructor>

        <room>"1216"</room>

        <address>"Lomonosova Street, Building 9"</address>

        </subject>

      <- name>"Saturday"</- name>

      <subject>

        <- name>"Linear Algebra"</- name>

        <type>"Lab Work"</type>

        <time>"08:20 - 09:50"</time>

        <instructor>"Isaeva Tatyana Timofeevna"</instructor>

        <room>"2113"</room>

        <address>"Kronversky Prospect 49"</address>

        <- name>"Fundamentals of Professional Activity"</- name>

        <type>"Lab Work"</type>

        <time>"09:50 - 11:30"</time>

        <instructor>"Oblyashevsky Sevastian Alexandrovich"</instructor>

        <room>"2308"</room>

        <address>"Kronversky Prospect 49"</address>

        </subject>

      </day>

    </schedule>

</root>

## Дополнительное задание №1

import yaml

import xml.etree.ElementTree as ET

import csv

import time

from typing import Any, Dict

# Updated YAML input data

yaml\_input = """

schedule:

  day:

    - name: "Wednesday"

      subject:

        - name: "Informatics"

          type: "Lecture"

          time: "08:20-09:50"

          instructor: "Balakshin Pavel Valeryevich"

          room: "1216"

          address: "Lomonosova Street, Building 9"

        - name: "Fundamentals of Professional Activity"

          type: "Lecture"

          time: "10:00-11:30"

          instructor: "Klimenkov Sergey Viktorovich"

          room: "1216"

          address: "Lomonosova Street, Building 9"

    - name: "Saturday"

      subject:

        - name: "Linear Algebra"

          type: "Lab Work"

          time: "08:20 - 09:50"

          instructor: "Isaeva Tatyana Timofeevna"

          room: "2113"

          address: "Kronversky Prospect 49"

        - name: "Fundamentals of Professional Activity"

          type: "Lab Work"

          time: "09:50 - 11:30"

          instructor: "Oblyashevsky Sevastian Alexandrovich"

          room: "2308"

          address: "Kronversky Prospect 49"

"""

# YAML to XML Conversion using a Library

def yaml\_to\_xml(yaml\_data: str) -> str:

    """Convert YAML to XML."""

    data = yaml.safe\_load(yaml\_data)

    root = ET.Element("schedule")

    for day\_data in data.get("schedule", {}).get("day", []):

        day\_elem = ET.SubElement(root, "day", name=day\_data["name"])

        for subject in day\_data.get("subject", []):

            subject\_elem = ET.SubElement(day\_elem, "subject")

            for key, value in subject.items():

                key\_elem = ET.SubElement(subject\_elem, key)

                key\_elem.text = str(value)

    return ET.tostring(root, encoding="unicode")

xml\_result = yaml\_to\_xml(yaml\_input)

print("--- Task 1: YAML to XML ---")

print(xml\_result)

## Дополнительное задание №2

import re

from dop1 import yaml\_input

import xml.etree.ElementTree as ET

# Task 2: Adding regex for parsing YAML to XML

def yaml\_to\_xml\_with\_regex(yaml\_data: str) -> str:

    """Convert YAML to XML using regex (simplified)."""

    yaml\_lines = yaml\_data.strip().split("\n")

    root = ET.Element("schedule")

    current\_day = None

    day\_pattern = re.compile(r"^\s\*-\sname:\s\"(.\*?)\"$")

    subject\_pattern = re.compile(r"^\s\*-\sname:\s\"(.\*?)\"$")

    key\_value\_pattern = re.compile(r"^\s\*(\w+):\s\"(.\*?)\"$")

    for line in yaml\_lines:

        day\_match = day\_pattern.match(line)

        if day\_match:

            current\_day = ET.SubElement(root, "day", name=day\_match.group(1))

            continue

        if current\_day:

            subject\_match = subject\_pattern.match(line)

            if subject\_match:

                subject\_elem = ET.SubElement(current\_day, "subject")

                continue

            key\_value\_match = key\_value\_pattern.match(line)

            if key\_value\_match and subject\_elem is not None:

                key\_elem = ET.SubElement(

                    subject\_elem, key\_value\_match.group(1))

                key\_elem.text = key\_value\_match.group(2)

    return ET.tostring(root, encoding="unicode")

# Run Task 2

xml\_result\_regex = yaml\_to\_xml\_with\_regex(yaml\_input)

print("\n--- Task 2: YAML to XML with Regex ---")

print(xml\_result\_regex)

## Дополнительное задание №3

import re

import xml.etree.ElementTree as ET

#from lab2 import file #from folder import file\_name

#from folder.file import name

from dop1 import yaml\_input

from pyparsing import Word, alphas, nums, Group, OneOrMore, Suppress, Literal, Dict, QuotedString

# Task 3: Using formal grammars for parsing YAML to XML

def yaml\_to\_xml\_with\_grammar(yaml\_data: str) -> str:

    """Convert YAML to XML using a formal grammar approach."""

    # Grammar definitions

    colon = Suppress(":")

    dash = Suppress("-")

    string = QuotedString('"')

    key\_value = Group(Word(alphas) + colon + string)

    subject = Group(dash + Dict(OneOrMore(key\_value)))

    day = Group(dash + Suppress("name:") + string("name") + Suppress("subject:") + OneOrMore(subject))

    schedule = Group(Literal("schedule:") + Suppress("day:") + OneOrMore(day))

    # Parse the YAML data

    parsed\_data = schedule.parseString(yaml\_data)

    root = ET.Element("schedule")

    for day in parsed\_data[1:]:

        day\_elem = ET.SubElement(root, "day", name=day["name"])

        for subject in day["subject"]:

            subject\_elem = ET.SubElement(day\_elem, "subject")

            for key, value in subject.items():

                key\_elem = ET.SubElement(subject\_elem, key)

                key\_elem.text = value

    return ET.tostring(root, encoding="unicode")

# Run Task 3

xml\_result\_grammar = yaml\_to\_xml\_with\_grammar(yaml\_input)

print("\n--- Task 3: YAML to XML with Formal Grammar ---")

print(xml\_result\_grammar)

## Дополнительное задание №4

import re

import xml.etree.ElementTree as ET

from dop1 import yaml\_input

from dop1 import yaml\_to\_xml

import time

# Task 4: Benchmarking execution time

def benchmark\_conversion(func, yaml\_data: str, iterations: int = 100) -> float:

    """Benchmark the YAML to XML conversion."""

    start\_time = time.time()

    for \_ in range(iterations):

        func(yaml\_data)

    return time.time() - start\_time

# Run Task 4

execution\_time = benchmark\_conversion(yaml\_to\_xml, yaml\_input)

print("\n--- Task 4: Benchmarking ---")

print(f"Execution time for 100 iterations: {execution\_time:.2f} seconds")

## Дополнительное задание №5

import re

import xml.etree.ElementTree as ET

from dop1 import xml\_result

import time

import csv

# Task 5: Convert YAML to CSV

def xml\_to\_csv(xml\_data: str, csv\_file: str) -> None:

    """Convert XML to CSV."""

    root = ET.fromstring(xml\_data)

    with open(csv\_file, "w", newline="") as file:

        writer = csv.writer(file)

        writer.writerow(["Day", "Subject Name", "Type", "Time", "Instructor", "Room", "Address"])

        for day\_elem in root:

            day = day\_elem.get("name")

            for subject\_elem in day\_elem:

                name = subject\_elem.find("name").text

                type\_ = subject\_elem.find("type").text

                time = subject\_elem.find("time").text

                instructor = subject\_elem.find("instructor").text

                room = subject\_elem.find("room").text

                address = subject\_elem.find("address").text

                writer.writerow([day, name, type\_, time, instructor, room, address])

csv\_file\_path = "output.csv"

xml\_to\_csv(xml\_result, csv\_file\_path)

print("\n--- Task 5: YAML to CSV ---")

print(f"CSV File Created at {csv\_file\_path}")

# Вывод

**В ходе лабораторной работы я** **познакомилась с разными форматами файлов. Напислa тремя разными способами парсер, который из json-файла получает xml-файл. Использовла знания ООП и регулярных выражений на практике.**