Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский университет ИТМО»

# Отчёт по лабораторной работе №4

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

Вариант 47

Выполнил**:** студент группы P3132 Салпагаров Элдар Эльбрусович

Проверил**:** Бострикова Дарья Константиновна, преподаватель практики

Дата сдачи: 4 декабря

Санкт-Петербург 2025

## Содержание

# Порядок выполнения работы

1. Определить номер варианта как остаток деления на 132 своего идентификационного номера в ISU: например, 125598 / 132 = 16. В случае, если в обоих указанных днях 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, RON, HCL, YAML, TOML, INI, 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 50 процентов от максимального числа баллов БаРС за данную лабораторную). Написать программу на языке Python 3.x или любом другом, которая:

• осуществляет парсинг и конвертацию исходного файла в бинарный объект (=десериализацию);

• для решения задачи использует 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 №2);

• не использует готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки файлов.

8. *Дополнительное задание №1* (позволяет набрать +15 процентов от максимального числа баллов БаРС за данную лабораторную). Написать программу на языке Python 3.x или любом другом, которая:

• осуществляет парсинг и конвертацию бинарного объекта, полученного в обязательном задании, в новый формат (=сериализацию);

• для решения задачи использует формальные грамматики;

• не использует готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки файлов.

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

a) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов (десериализацию и сериализацию).

b) Переписать исходный код и код из дополнительного задания №1, 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 №3* (позволяет набрать +20 процентов от максимального числа баллов БаРС за данную лабораторную). Переписать код из дополнительного задания №1, чтобы серализация происводилась в XML файл.

11.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.

12.Проверить, что все пункты задания выполнены и выполнены верно.

13.Написать отчёт о проделанной работе.

14.Подготовиться к устным вопросам на защите.

# Выполнение задачи

Обязательное задание:

Исходный/входной RON-файл:

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/input.ron

Исходный код программы (включая доп №1 и №3):

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/main.py

Выходной YAML-файл (Доп №1):

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/output.yaml

Выходной XML-файл (Доп №3):

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/output.xml

Дополнительное задание №2:

Исходный переписанный с библиотекой код:

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/dop_2.py

Выходной YAML-файл:

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/output_lib.yaml

Выходной XML-файл:

https://github.com/d6vmc/My-labs/blob/bf9a99bd02d3afbc3e944a5d979b94fac58d91a1/Inf_Lab3/output_lib.xml

Сторонние библиотеки:

Lark – для парсинга RON (так как нет готового решения, пришлось расписывать грамматику самому).

PyYaml (модуль yaml) – для работы с форматом YAML.

dicttoxml – для удобной конвертации словарей в XML.

Стандартные библиотеки:

xml.dom.minidom - для работы с XML-структурой (DOM).

Результаты почти не имеют отличий, мой парсер использует Tab для секций, а подключенные модули используют спец. пробел. С библиотеками код стал намного более читаемым и чистым, а также сократился в 3 раза (считаю сериализацию из доп №1 и доп №3, так как ее я также реализовал и с модулями).

Дополнительное задание №4:

Исходный код программы-бенчмарка:

https://github.com/d6vmc/My-labs/blob/e3f07942ae3e7c60f7d4991ea4516a1cdf4a0a58/Inf_Lab3/dop_4.py

В результате проверки скорости 100-кратной работы парсеров замечу, что написанный без библиотек парсер работает быстрее в ~6,5 раз. Ручной парсер оказался быстрее, так как специализирован под определенный формат данных и отсутствию избыточных проверок.

Lark же тратит ресурсы на построение синтаксического дерева (AST), множество вложенных вызовов функций для обработки правил грамматики, но обеспечивает универсальность и легкость создания парсера на нем

Для обучения создание ручного парсера будет безусловно лучше, для продакшена – библиотечное решение.

# Заключение

В ходе работы я узнал много нового о классах, библиотеках и парсинге в Python, научился проводить бенчмарки программ, познакомился с формальными грамматиками и языками разметки.

# Литература

1) Грошев А.С. Г89 Информатика: Учебник для вузов / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2010. – 470с.
URL: *https://narfu.ru/university/library/books/0690.pdf*
2) Балакшин П.В., Соснин В.В., Машина Е.А. Информатика. – СПб: Университет ИТМО, 2020. – 122 с.