Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный Исследовательский Университет ИТМО»

Факультет Программной Инженерии и Компьютерной техники

Лабораторная работа № 4
Исследование протоколов, форматов обмена информацией и языков разметки документов
Вариант № 51

Выполнил:
Михальченков Александр Николаевич
P3109
Проверила:
Малышева Татьяна Алексеевна

Санкт-Петербург 2025

Оглавление

Задание

Вариант:	51	HCL	TOML	Среда, суббота

1. Понять устройство страницы с расписанием на примере расписания лектора: https://itmo.ru/ru/schedule/3/125598/raspisanie_zanyatiy.htm

2. 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.

3. 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.

4. 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.

5. 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.

6. Дополнительное задание № 3 (позволяет набрать +20 процентов от максимального числа баллов БаРС за данную лабораторную).

Переписать код из дополнительного задания №1, чтобы серализация производилась в XML файл.

7. 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.

Основные этапы выполнения

1. Обязательное задание

Исходный файл в формате HCL: mandatory_task/data/schedule.hcl

Реализация механизма парсинга формата HCL: formats/hcl_parser.py

Реализация механизма бинарной сериализации и десериализации: formats/bin_codec.py

Реализация парсинга и конвертации исходного файла в формате HCL в бинарный объект: mandatory_task/convert.py

2. Дополнительное задание № 1

Реализация механизма бинарной сериализации и десериализации: formats/bin_codec.py

Реализация механизма сериализации в формат TOML: formats/toml_serializer.py

Реализация парсинга и конвертации бинарного объекта, полученного в обязательном задании, в формат TOML: optional_task_1/convert.py

Полученный файл в формате TOML: optional_task_1/data/schedule.toml

3. Дополнительное задание № 2

Библиотека, осуществляющая парсинг формата HCL: https://pypi.org/project/python-hcl2/

Стандартный модуль Python, осуществляющий механизм бинарной сериализации и десериализации: https://docs.python.org/3/library/pickle.html

Библиотека, осуществляющая механизм сериализации в формат TOML: https://docs.python.org/3/library/tomllib.html

Исходный файл в формате HCL: optional_task_2/data/schedule.hcl

Реализация механизма парсинга и конвертации исходного файла в формате HCL в бинарный объект, а также обратной конвертации бинарного объекта в формат TOML: optional_task_2/convert.py

Полученный файл в формате TOML: optional_task_2/data/schedule.toml

Полученные файлы в формате TOML в дополнительном задании № 1 и в дополнительном задании № 2 совпадают, так как при написании собственных конвертеров я ориентировался на принятые стандарты представления данных и реализовал их преобразование с использованием алгоритмов, схожих с подходами, используемыми в широко распространённых библиотечных решениях. Однако код программы стал лаконичнее, поскольку парсинг HCL формата, механизм бинарной сериализации и десериализации, а также механизм сериализации в формат TOML реализованы с помощью соответствующих библиотек и модулей Python, а не написаны вручную.

4. Дополнительное задание № 3

Реализация механизма бинарной сериализации и десериализации: formats/bin_codec.py

Реализация механизма сериализации в формат XML: formats/xml_serializer.py

Реализация парсинга и конвертации бинароного объекта, полученного в обязательном задании, в формат XML: optional_task_3/convert.py

Полученный файл в формате XML: optional_task_3/data/schedule.xml

5. Дополнительное задание № 4

Реализация алгоритма подсчёта стократного времени выполнения:

- парсинга и конвертации формата HCL в формат TOML с помощью рукописных алгоритмов
- парсинга и конвертации формата HCL в формат TOML с использованием библиотек и модулей Python
- парсинга и конвертации формата HCL в формат XML с помощью рукописных алгоритмов

в цикле: optional_task_4/benchmark.py

Полученный результат (см. рисунок 1):

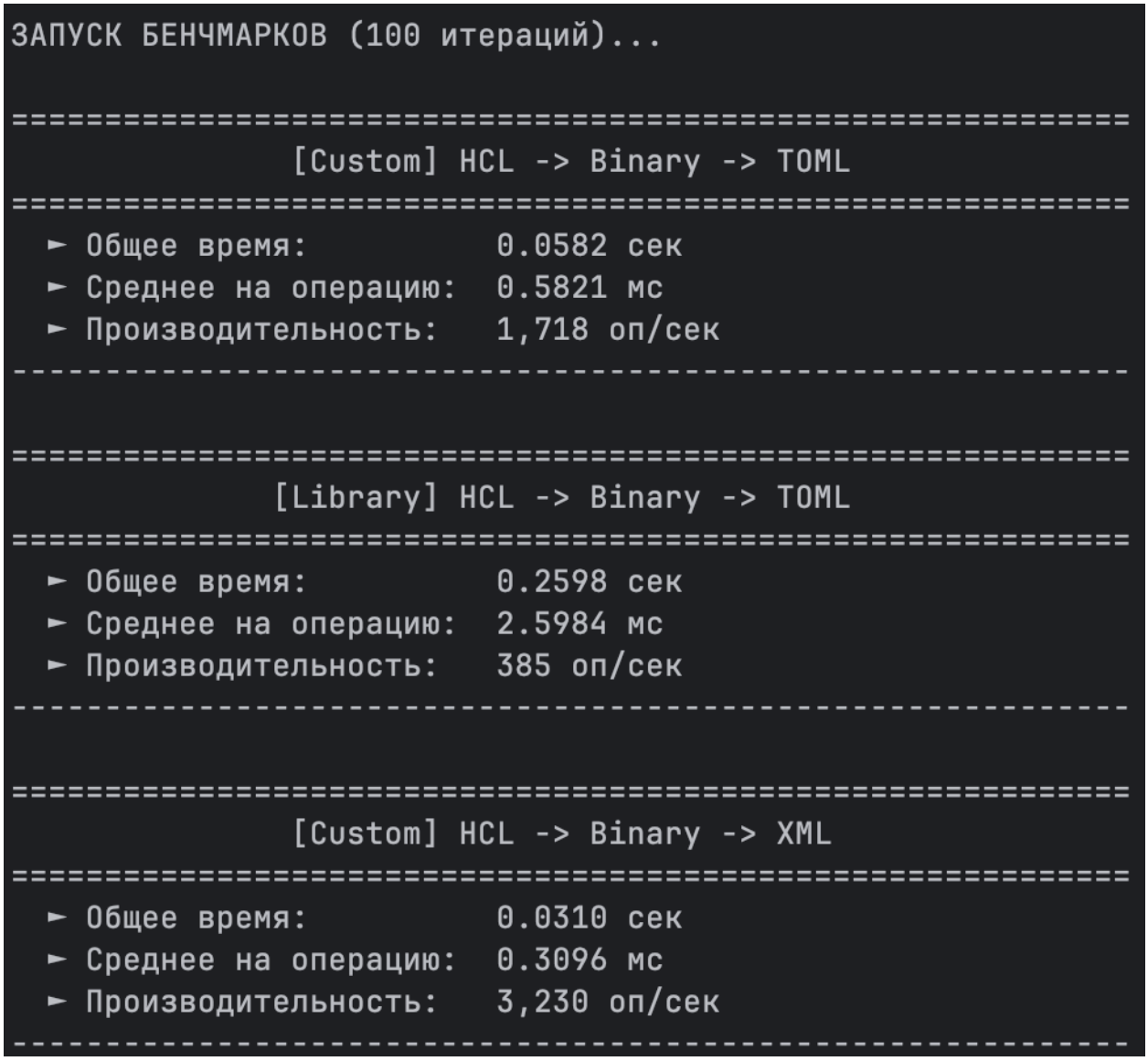

Рисунок 1. Результат тестирования различных парсеров и конвертаций

Сравнение полученных результатов:

На основе полученных экспериментальных данных (100 итераций) были зафиксированы следующие показатели времени выполнения и пропускной способности (см. таблицу 1):

Алгоритм / Метод реализации	Общее время (сек)	Среднее время (мс)	Производительность (оп/сек)
Custom: HCL → Binary → TOML (Рукописный)	0.0582	0.5821	1 718

Library: HCL → Binary → TOML (Библиотечный)	0.2598	2.5984	385
Custom: HCL → Binary → XML (Рукописный)	0.0310	0.3096	3230

Таблица 1. Сравнение производительности алгоритмов конвертации

Анализ полученных результатов:

1. Сравнение рукописной реализации и библиотечного решения (HCL → TOML)

Наблюдение: рукописная реализация (Custom) оказалась быстрее библиотечной (Library) примерно в 4.5 раза (1718 оп/сек против 385 оп/сек).

Причины:

• Архитектурная легковесность: сторонние библиотеки (особенно для HCL) часто построены на базе сложных генераторов парсеров, которые сначала создают громоздкое абстрактное синтаксическое дерево (AST) и проводят множество внутренних проверок целостности грамматики. Рукописный алгоритм, являясь универсальным, реализован более эффективно: он работает напрямую с потоком токенов, преобразуя их сразу в структуры Python (словари и списки), минуя стадию создания тяжелых промежуточных объектов.

• Накладные расходы библиотек: библиотечные решения обычно содержат многоуровневую архитектуру классов и абстракций для поддержки совместимости с разными версиями стандартов и экосистем. Собственная реализация лишена этого «инфраструктурного веса» (overhead), выполняя ровно ту логику парсинга и сериализации, которая требуется алгоритмом.

• Эффективность бинарного формата: библиотечный вариант использовал стандартный модуль pickle, который сохраняет много метаинформации об объектах Python. Рукописный BinCodec реализует компактный TLV-подобный (Type-Length-Value) формат, оперируя байтами напрямую, что значительно быстрее при записи и чтении.

2. Сравнение форматов вывода в рукописных реализациях (TOML vs XML)

Наблюдение: конвертация в XML выполняется быстрее, чем в TOML, примерно в 1.9 раза (3230 оп/сек против 1718 оп/сек), при том что оба алгоритма реализованы вручную.

Причины:

- Алгоритмическая сложность записи:
  - XML: структура XML иерархична и линейна. Сериализатор просто обходит дерево данных рекурсивно и пишет теги последовательно в поток.
  - TOML: спецификация формата требует строгого порядка записи: сначала примитивы, затем таблицы и массивы таблиц. Это вынуждает алгоритм на каждом уровне вложенности сортировать или группировать ключи перед записью, что добавляет лишние проходы по данным.
- Строковые операции: генерация TOML требует более сложной логики для построения полных путей к ключам (например, [day.wednesday.lesson]), в то время как XML формируется простой конкатенацией открывающих и закрывающих тегов с учетом отступов.

Вывод:

Рукописные решения показали значительно более высокую производительность за счёт прямой реализации алгоритмов парсинга и отсутствия архитектурных излишеств, свойственных универсальным библиотекам. При этом формат XML оказался алгоритмически проще и быстрее для генерации, чем TOML, из-за особенностей структуры последнего (требование группировки типов данных).

Заключение

В ходе выполнения лабораторной работы мной были закреплены теоретические знания о языках разметки данных (HCL, TOML, XML) и освоены практические навыки их программной обработки. Кроме того, мной было изучено устройство механизмов парсинга и конвертации, реализован собственный протокол бинарной сериализации, а также проведено исследование эффективности рукописных и библиотечных решений. Работа позволила развить навыки написания оптимизированного кода на Python, внимание к деталям реализации форматов и углубить понимание принципов трансформации данных, что является важной основой для дальнейшего профессионального развития в сфере IT.

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. — экз. — Режим доступа: https://books.ifmo.ru/book/2248/informatika:_laboratornye_raboty_i_testy:_uchebno-metodicheskoe_posobie_/_recenzent:_polyakov_v.i..htm

2. Brikman Y. Terraform: Up and Running: Writing Infrastructure as Code / Y. Brikman. — 3rd ed. — Sebastopol: O'Reilly Media, 2022. — 430 p. — (HCL syntax overview). — Режим доступа: https://www.oreilly.com/library/view/terraform-up-and/9781098116736/

3. HashiCorp. HCL (HashiCorp Configuration Language) Specification [Электронный ресурс]. — Режим доступа: https://github.com/hashicorp/hcl (дата обращения: 23.11.2025).

4. Preston-Werner T. TOML: Tom's Obvious, Minimal Language. Version 1.0.0 [Электронный ресурс]. — Режим доступа: https://toml.io/en/ (дата обращения: 25.11.2025).

5. Bray T. et al. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Электронный ресурс] / W3C Recommendation. — 2008. — Режим доступа: https://www.w3.org/TR/xml/ (дата обращения: 24.11.2025).