МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И  
КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине

«ИНФОРМАТИКА»

Исследование протоколов,

форматов обмена информацией и языков разметки

документов

Вариант № 2

***Выполнил:***Студент группы P3107  
Добрышкин Владимир Александрович

***Проверил:***Балакшин Павел Валерьевич

кандидат технических наук, доцент факультета ПИиКТ

Содержание

[Задание 3](#_Toc178174515)

[Основные этапы вычисления 5](#_Toc178174516)

[Заключение 10](#_Toc178174517)

[Список литературы 11](#_Toc178174518)

# 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.

1. Изучить форму Бэкуса-Наура.
2. Изучить основные принципы организации формальных грамматик.
3. Изучить особенности языков разметки/форматов JSON, YAML, XML.
4. Понять устройство страницы с расписанием на примере расписания лектора: https://itmo.ru/ru/schedule/3/125598/raspisanie\_zanyatiy.htm
5. 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 одном 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.
6. 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.
7. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
8. 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.
9. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
10. Переписать исходный код, добавив в него использование регулярных выражений.
11. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
12. Дополнительное задание № 3 (позволяет набрать +25 процентов от максимального числа баллов БаРС за данную лабораторную).
13. 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.
14. Проверку осуществить как минимум для расписания с двумя учебными днями по два занятия в каждом.
15. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
16. Дополнительное задание № 4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
17. Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
18. Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
19. Дополнительное задание № 5 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
20. 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.
21. Проанализировать полученные результаты, объяснить особенности использования формата. Объяснение должно быть отражено в отчёте.
22. Проверить, что все пункты задания выполнены и выполнены верно.
23. Написать отчёт о проделанной работе.
24. Подготовиться к устным вопросам на защите.

# Основные этапы вычисления

*Таблица 1 - обобщённые требования к парсерам*

|  |  |
| --- | --- |
| № Задания | Требования к заданию |
| 1 | Написать парсер, не используя никаких внешних и встроенных библиотек. |
| 1(доп.) | Написать парсер с использованием внешних библиотек для парсинга. |
| 2(доп.) | Написать парсер с использованием регулярных выражений. |
| 3(доп.) | Написать парсер с использованием формальных грамматик. |
| 5(доп.) | Написать парсер с одним из требований к предыдущим парсерам. |

Исходный файл с расписанием:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/data/self_schedule.json>

**Обязательное задание:**

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task1.py>

Результат парсинга:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/output/self_schedule1.yaml>

Инструменты для парсинга:

* Стандартная библиотека Python.

**Дополнительное задание №1:**

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task2.py>

Результат парсинга:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/output/self_schedule2.yaml>

Инструменты для парсинга:

* Встроенный модуль json.
* Внешний модуль pyYAML.

Различия результатов с результатами предыдущих реализаций:

* Модуль pyYAML автоматически сортирует ключи в алфавитном порядке в итоговом файле, что можно исправить, поменяв значение флага внутри функции.
* При переходе к словарю внутри массива, библиотека не ставит дополнительный отступ слева, что делает файл менее наглядным.

**Дополнительное задание №2:**

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task3.py>

Результат парсинга:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/output/self_schedule3.yaml>

Инструменты для парсинга:

* Встроенный модуль re.

Различия результатов с результатами предыдущих реализаций:

* В отличие от двух предыдущих способов парсинга, данный метод не является универсальным и подогнан конкретно под изначальный файл.
* Итоговый файл аналогичен файлу при обработке первым методом и имеет такие же различия со вторым, как и первый.

**Дополнительное задание №3:**

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task4.py>

Результат парсинга:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/output/self_schedule4.yaml>

Инструменты для парсинга:

* Конечные автоматы для (ре) парсинга.
* Стандартная библиотека Python.
* Встроенный модуль re.

Различия результатов с результатами предыдущих реализаций:

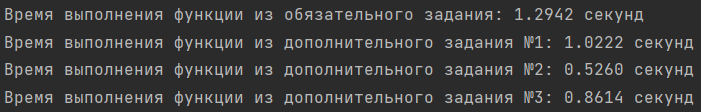
* В данном методе используются две БНФ (одна для json, другая для yaml), а так же реализовано два конечных автомата для парсинга данных грамматик.
* Итоговый файл аналогичен файлу при обработке первым или третьим методом и имеет такие же различия со вторым, как и первый и третий.

**Дополнительное задание №4:**

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task5.py>

Результат работы программы представлен на .



*Рисунок 1* *– результат работы программы для измерения времени работы парсеров*

Быстрее всего работает изначально подогнанная под конкретный файл функция из дополнительного задания №3, использующая регулярные выражения.

Из универсальных способов быстрее всего работает вариант с использованием формальных грамматик, опережая функцию, которая использует сторонние библиотеки.

Это может быть связано с тем, что внешние библиотеки, такие как pyYAML и json, предлагают универсальные решения, которые должны учитывать множество различных случаев и форматов, что может добавлять накладные расходы на обработку. В библиотеке pyYAML по умолчанию стоит сортировка ключей полученного YAML-файла. Использование собственных алгоритмов может уменьшить количество промежуточных шагов и операций.

**Дополнительное задание №5:**

Выбранный формат: TOML (Tom’s Obvious Minimal Language)

Исходный код:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/task6.py>

Результат парсинга

:

<https://github.com/vodobryshkin/ITMO/blob/main/1%20%D0%9A%D1%83%D1%80%D1%81/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%E2%84%964/output/self_schedule.toml>

Инструменты для парсинга:

* Конечные автоматы для (ре) парсинга.
* Стандартная библиотека Python.
* Встроенный модуль re.

Полученные результаты полностью соответствуют синтаксичесим правилам TOML.

TOML (Tom's Obvious, Minimal Language) — это формат конфигурационных файлов, который разработан для обеспечения простоты и удобочитаемости. Особенности использования TOML:

1. Легкая читаемость.
2. Структурированность: Поддерживает вложенные структуры с помощью таблиц (tables) и подтаблиц..
3. Комментарии: Позволяет добавлять комментарии, которые начинаются с символа #. Это помогает документировать конфигурацию.
4. Отсутствие сложных структур: Не поддерживает сложные структуры данных, такие как массивы объектов, что делает его более простым.
5. Упорядоченность: Порядок ключей в TOML важен, что позволяет сохранять последовательность данных.

# Заключение

В ходе проделанной лабораторной работы, я познакомился с различными форматами файлов. Узнал о существовании формальных грамматик и БНФ. Написал парсеры, которые конвертируют JSON-файл в файлы форматов YAML и TOML.

# Список литературы

1. [Электронный ресурс]: Кафедра алгоритмических языков ВМК МГУ. – Режим доступа: <https://al.cs.msu.ru/files/formal.languages.translation.theory.pdf>
2. Грошев А.С. Г89 Информатика: Учебник для вузов / А.С. Грошев. – Архангельск, Арханг. гос. техн. ун-т, 2010. - 470с. - Режим доступа: <http://arm.sies.uz/wp-content/uploads/2020/11/16-%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0-2010-%D0%B4%D0%B0%D1%80%D1%81%D0%BB%D0%B8%D0%BA-%D0%93%D1%80%D0%BE%D1%88%D0%B5%D0%B2-%D0%90.%D0%A1.pdf>