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 5 (XML -> YAML, пн, чт)

***Выполнил:***Студент группы P3107  
 Пшеничников Артём Дмитриевич

***Проверил:***Балакшин Павел Валерьевич

кандидат технических наук, доцент факультета ПИиКТ

Оглавление

[Задание 3](#_Toc181533987)

[Основные этапы вычисления 4](#_Toc181533988)

[1. Обязательное задание 4](#_Toc181533989)

[2. Дополнительное задание 1 7](#_Toc181533990)

[3. Дополнительное задание 2 8](#_Toc181533991)

[4. Дополнительное задание 3 10](#_Toc181533992)

[5. Дополнительное задание 4 12](#_Toc181533993)

[6. Дополнительное задание 5 13](#_Toc181533994)

[Заключение 18](#_Toc181533995)

[Список литературы 19](#_Toc181533996)

# Задание

1. 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.
2. 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.
3. 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. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
4. Дополнительное задание № 3 (позволяет набрать +25 процентов от максимального числа баллов БаРС за данную лабораторную).
   1. 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.
   2. Проверку осуществить как минимум для расписания с двумя учебными днями по два занятия в каждом.
   3. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
5. 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.
6. 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

# Основные этапы вычисления

## 1. Обязательное задание

Написать программу на языке осуществляла парсинг и конвертацию исходного файла в новый путём простой замены метасимволов исходного формата на метасимволы результирующего формата.

Входные данные (аналогичны входным данным для всех подзаданий):

<?xml version="1.0"?>  
<Timetable>  
 <User>Пшеничников Артём Дмитриевич</User>  
 <UserISU>467205</UserISU>  
 <Week>  
 <Monday>  
 <Lesson Time="08:20">  
 <Lector>Цю Тяньшэн</Lector>  
 <Theme>Основы профессиональной деятельности</Theme>  
 <Type>Лабораторная</Type>  
 <Classroom>2112</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 <Lesson Time="10:00">  
 <Lector>Цю Тяньшэн</Lector>  
 <Theme>Основы профессиональной деятельности</Theme>  
 <Type>Лабораторная</Type>  
 <Classroom>2112</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 <Lesson Time="15:20">  
 <Lector>Карпов Дмитрий Валерьевич</Lector>  
 <Theme>Линейная алгебра (продвинутый уровень)</Theme>  
 <Type>Лекция</Type>  
 <Classroom>1404</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 <Lesson Time="17:00">  
 <Lector>Покидова Марина Владимировна</Lector>  
 <Theme>Линейная алгебра (продвинутый уровень)</Theme>  
 <Type>Практика</Type>  
 <Classroom>2426</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 <Lesson Time="19:00">  
 <Lector>Рекин Владимир Захарович</Lector>  
 <Theme>Шашки</Theme>  
 <Type>Спорт</Type>  
 <Classroom>Бильярдный клуб (1784)</Classroom>  
 <Address>Вяземский пер., д.5-7, лит.А</Address>  
 </Lesson>  
 </Monday>  
 <Thursday>  
 <Lesson Time="11:40">  
 <Lector>Щеглова Иванна Павловна</Lector>  
 <Theme>История русской культуры в контексте мировой культуры</Theme>  
 <Type>Практика</Type>  
 <Classroom>2412</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 <Lesson Time="13:30">  
 <Lector>Жиркова Галина Петровна</Lector>  
 <Theme>История русской культуры в контексте мировой культуры</Theme>  
 <Type>Лекция</Type>  
 <Classroom>1405</Classroom>  
 <Address>Кронверкский пр., д.49, лит.А</Address>  
 </Lesson>  
 </Thursday>  
 </Week>  
</Timetable>

Выходные данные (аналогичны входным данным для всех подзаданий, кроме дополнительного задания 1):

Timetable:  
 User: 'Пшеничников Артём Дмитриевич'  
 UserISU: '467205'  
 Week:  
 Monday:  
 Lesson:  
 - Lector: 'Цю Тяньшэн'  
 Theme: 'Основы профессиональной деятельности'  
 Type: 'Лабораторная'  
 Classroom: '2112'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '08:20'  
 - Lector: 'Цю Тяньшэн'  
 Theme: 'Основы профессиональной деятельности'  
 Type: 'Лабораторная'  
 Classroom: '2112'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '10:00'  
 - Lector: 'Карпов Дмитрий Валерьевич'  
 Theme: 'Линейная алгебра (продвинутый уровень)'  
 Type: 'Лекция'  
 Classroom: '1404'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '15:20'  
 - Lector: 'Покидова Марина Владимировна'  
 Theme: 'Линейная алгебра (продвинутый уровень)'  
 Type: 'Практика'  
 Classroom: '2426'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '17:00'  
 - Lector: 'Рекин Владимир Захарович'  
 Theme: 'Шашки'  
 Type: 'Спорт'  
 Classroom: 'Бильярдный клуб (1784)'  
 Address: 'Вяземский пер., д.5-7, лит.А'  
 \_Time: '19:00'  
 Thursday:  
 Lesson:  
 - Lector: 'Щеглова Иванна Павловна'  
 Theme: 'История русской культуры в контексте мировой культуры'  
 Type: 'Практика'  
 Classroom: '2412'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '11:40'  
 - Lector: 'Жиркова Галина Петровна'  
 Theme: 'История русской культуры в контексте мировой культуры'  
 Type: 'Лекция'  
 Classroom: '1405'  
 Address: 'Кронверкский пр., д.49, лит.А'  
 \_Time: '13:30'

Листинг программы:

import pprint  
  
  
class XmlToYaml:  
 def main(self, debug=False):  
 f = open("input.xml", encoding="utf-8")  
 data = f.read()  
 f.close()  
 out = dict()  
 stack = []  
 curr\_filed = ""  
 in\_reading\_header = False  
 in\_reading\_field = False  
 counter = -1  
 while counter < len(data) - 1:  
 counter += 1  
 curr\_symbol = data[counter]  
 if in\_reading\_header == False and in\_reading\_field == False:  
 if curr\_symbol == "<":  
 in\_reading\_header = True  
 stack.append("")  
 else:  
 curr\_filed += curr\_symbol  
 continue  
 if in\_reading\_header:  
 if curr\_symbol == ">":  
 in\_reading\_header = False  
 if len(stack) > 1 and len(stack[-1]) > 1 and stack[-1][0] == "/" and stack[-2].split(" ")[0] == stack[-1][1:]:  
 stack.pop(-1)  
 if curr\_filed.replace("\n", " ").count(" ") != len(curr\_filed):  
 self.add\_filed(out, stack.copy(), curr\_filed)  
 stack.pop(-1)  
 curr\_filed = ""  
 else:  
 self.add\_dict(out, stack.copy())  
 curr\_filed = ""  
 else:  
 stack[-1] = stack[-1] + curr\_symbol  
 out = out[list(out.keys())[0]]  
 if debug: pprint.pprint(out, width=200)  
  
 file = open("outputTask.yaml", "w", encoding="utf-8")  
 curr\_spase = 0  
 self.write\_to\_yaml(file, out, curr\_spase, False)  
 file.close()  
 return "Изменения успешно записаны в файл outputTask.yaml"  
  
 def write\_to\_yaml(self, file, out, curr\_spase, arrow\_before\_first):  
 last\_header = ""  
 for q in out.keys():  
 st = " " \* curr\_spase + q + ":"  
 lst\_attr = []  
 if " " in q:  
 st = " " \* curr\_spase + q.split(" ")[0] + ":"  
 lst\_attr = q.split(" ")[1:]  
 if type(out[q]) != dict:  
 st += " '" + out[q] + "'\n"  
 if arrow\_before\_first:  
 arrow\_before\_first = False  
 st = st[:curr\_spase - 2] + "-" + st[curr\_spase - 1:]  
 file.write(st)  
 else:  
 st += "\n"  
 if last\_header != st:  
 file.write(st)  
 last\_header = st  
 for w in lst\_attr:  
 out[q]["\_" + w.split("=")[0]] = w.split("=")[1][1:-1]  
 self.write\_to\_yaml(file, out[q], curr\_spase + 2, True if len(out.keys()) > 1 else False)  
  
 def add\_dict(self, out, stack):  
 if len(stack) > 1:  
 self.add\_dict(out[stack.pop(0)], stack)  
 else:  
 out[stack[0]] = dict()  
  
 def add\_filed(self, out, stack, filed):  
 if len(stack) > 1:  
 self.add\_filed(out[stack.pop(0)], stack, filed)  
 else:  
 out[stack[0]] = filed  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(XmlToYaml().main(True))

## 2. Дополнительное задание 1

1. Были найдены библиотеки yaml и xmltodoct
2. Исходный код:

import xmltodict  
import yaml  
  
  
class XmlToYamlLib:  
 def main(self):  
 f = open("input.xml", encoding="utf-8")  
 data = f.read()  
 f.close()  
 data\_dict = xmltodict.parse(data, encoding="utf-8")  
 f = open("outputDop1.yaml", "w", encoding="utf-8")   
 f.write(yaml.dump(data\_dict, default\_flow\_style=False, allow\_unicode=True))  
 f.close()  
 return "Изменения успешно записаны в файл outputDop1.yaml"  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(XmlToYamlLib().main())

с) Выходные данные:

Timetable:  
 User: Пшеничников Артём Дмитриевич  
 UserISU: '467205'  
 Week:  
 Monday:  
 Lesson:  
 - '@Time': 08:20  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '2112'  
 Lector: Цю Тяньшэн  
 Theme: Основы профессиональной деятельности  
 Type: Лабораторная  
 - '@Time': '10:00'  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '2112'  
 Lector: Цю Тяньшэн  
 Theme: Основы профессиональной деятельности  
 Type: Лабораторная  
 - '@Time': '15:20'  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '1404'  
 Lector: Карпов Дмитрий Валерьевич  
 Theme: Линейная алгебра (продвинутый уровень)  
 Type: Лекция  
 - '@Time': '17:00'  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '2426'  
 Lector: Покидова Марина Владимировна  
 Theme: Линейная алгебра (продвинутый уровень)  
 Type: Практика  
 - '@Time': '19:00'  
 Address: Вяземский пер., д.5-7, лит.А  
 Classroom: Бильярдный клуб (1784)  
 Lector: Рекин Владимир Захарович  
 Theme: Шашки  
 Type: Спорт  
 Thursday:  
 Lesson:  
 - '@Time': '11:40'  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '2412'  
 Lector: Щеглова Иванна Павловна  
 Theme: История русской культуры в контексте мировой культуры  
 Type: Практика  
 - '@Time': '13:30'  
 Address: Кронверкский пр., д.49, лит.А  
 Classroom: '1405'  
 Lector: Жиркова Галина Петровна  
 Theme: История русской культуры в контексте мировой культуры  
 Type: Лекция

Как можно видеть, парсинг в библиотеке реализован так, что закавычиваются не все текстовые поля, а только некоторые, содержащие числовые значения. Также атрибуты полей выносятся на первое место в блоке и помечаются символом @, впрочем, встречаются различные варианты того, как отмечаются атрибуты.

## 3. Дополнительное задание 2

a) Исходный код:

import pprint  
import re  
  
  
class XmlToYamlRegEx:  
 def main(self, debug=False):  
 f = open("input.xml", encoding="utf-8")  
 data = f.read()  
 f.close()  
 headers = []  
 for q in re.finditer(r"<[^>]+>", data):  
 headers.append([q[0], q.start(), ""])  
 for q in range(len(headers)):  
 if headers[q][0][:2] == "</" and headers[q][0][2:] == headers[q - 1][0][1:]:  
 headers[q - 1][2] = data[headers[q - 1][1] + len(headers[q - 1][0]):headers[q][1]]  
 headers.pop(0)  
 res = dict()  
 self.rec\_add(res, headers, 0)  
 if debug: pprint.pprint(res, width=200)  
  
 file = open("outputDop2.yaml", "w", encoding="utf-8")  
 curr\_spase = 0  
 self.write\_to\_yaml(file, res, curr\_spase, False)  
 file.close()  
 return "Изменения успешно записаны в файл outputDop2.yaml"  
  
 def rec\_add(self, res: dict, headers: list[list[str | int]], start\_on: int) -> None:  
 if not headers: return  
 if start\_on > len(headers) - 1: return  
 curr = headers[start\_on]  
 curr\_name = curr[0][1:-1]  
 if curr[2]:  
 res[curr\_name] = curr[2]  
 else:  
 res[curr\_name] = dict()  
 end\_ind = len(headers)  
 for q in range(start\_on, len(headers)):  
 if headers[q][0][2:-1] == curr[0][1:-1].split(" ")[0]:  
 end\_ind = q  
 break  
 self.rec\_add(res[curr\_name], headers[start\_on + 1:end\_ind], 0)  
 self.rec\_add(res, headers, end\_ind + 1)  
  
 def write\_to\_yaml(self, file, out, curr\_spase, arrow\_before\_first):  
 last\_header = ""  
 for q in out.keys():  
 st = " " \* curr\_spase + q + ":"  
 lst\_attr = []  
 if " " in q:  
 st = " " \* curr\_spase + q.split(" ")[0] + ":"  
 lst\_attr = q.split(" ")[1:]  
 if type(out[q]) != dict:  
 st += " '" + out[q] + "'\n"  
 if arrow\_before\_first:  
 arrow\_before\_first = False  
 st = st[:curr\_spase - 2] + "-" + st[curr\_spase - 1:]  
 file.write(st)  
 else:  
 st += "\n"  
 if last\_header != st:  
 file.write(st)  
 last\_header = st  
 for w in lst\_attr:  
 out[q]["\_" + w.split("=")[0]] = w.split("=")[1][1:-1]  
 self.write\_to\_yaml(file, out[q], curr\_spase + 2, True if len(out.keys()) > 1 else False)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(XmlToYamlRegEx().main(True))

1. Результаты работы программ идентичны

## 4. Дополнительное задание 3

а) Переписать исходный код таким образом, чтобы для

решения задачи использовались формальные грамматики.

Полученный исходный код:

import pprint  
import re  
  
  
class XmlToYamlFull:  
 def main(self, debug=False):  
 file = open("input.xml", encoding="utf-8")  
 data = file.read()  
 file.close()  
 headers = []  
 res = dict()  
 self.parse\_headers(headers, data) # парсим теги  
 self.rec\_add(res, headers, 0) # рекурсивно формируем словарь  
 self.group\_equal\_tags(res, []) # группируем одинаковые теги и выносим их атрибуты  
 if debug: pprint.pprint(res, width=200)  
  
 file = open("outputDop3.yaml", "w", encoding="utf-8")  
 curr\_spase = 0  
 self.write\_to\_yaml(file, res, curr\_spase, False) # пишем в yaml  
 file.close()  
 return "Изменения успешно записаны в файл outputDop3.yaml"  
  
 @staticmethod  
 def parse\_headers(headers, data):  
 for q in re.finditer(r"<[^>]+>", data): # ищем все теги по регулярке  
 headers.append([q[0], q.start(), ""])  
 if re.fullmatch(r"<!--[^>]+-->", headers[-1][0]): headers.pop(-1) # удаляем <!-- комментарии -->  
 if headers[-1][0][-2:] == "/>":  
 headers[-1][0] = headers[-1][0][:-2] + ">" # случай, когда тег пустой, делаем ещё один фиктивный  
 headers.append(["</" + headers[-1][0].split(" ")[0][1:] + ">", headers[-1][1], headers[-1][2]])  
 for q in range(len(headers)): # пишем текстовые данные там, где они есть  
 if headers[q][0][:2] == "</" and headers[q][0][2:] == headers[q - 1][0][1:]:  
 headers[q - 1][2] = data[headers[q - 1][1] + len(headers[q - 1][0]):headers[q][1]]  
 headers.pop(0)  
  
 def rec\_add(self, res: dict, headers: list[list[str | int]], start\_on: int) -> None:  
 if not headers: return  
 if start\_on > len(headers) - 1: return # завершение рекурсии  
 curr = headers[start\_on]  
 curr\_name = curr[0][1:-1]  
 if curr[2]:  
 if curr\_name in res.keys():  
 if type(res[curr\_name]) == list: res[curr\_name].append(self.replace\_entities(curr[2])) # заменяем сущ.  
 else: res[curr\_name] = [res[curr\_name], self.replace\_entities(curr[2])] # и пишем, если строка не пустая  
 else: res[curr\_name] = self.replace\_entities(curr[2])  
 else: res[curr\_name] = dict() # создаём словарь если пустая строка  
 end\_ind = len(headers)  
 for q in range(start\_on, len(headers)):  
 if headers[q][0][2:-1] == curr[0][1:-1].split(" ")[0]: # если два соседних тега, записываем  
 end\_ind = q # стартовую позицию и передаём дальше  
 break  
 self.rec\_add(res[curr\_name], headers[start\_on + 1:end\_ind], 0) # рекурсия вглубь  
 self.rec\_add(res, headers, end\_ind + 1) # и вбок  
  
 @staticmethod  
 def replace\_entities(s):  
 for q in [["&lt;", '<'], ["&gt;", '>'], ["&amp;", '&'], ["&apos;", "'"], ["&quot;", '"']]:  
 s = s.replace(q[0], q[1]) # замена сущностей  
 return s  
  
 def group\_equal\_tags(self, res: dict, add\_fields: list[list]):  
 if type(res) != dict: return  
 if add\_fields: # добавляем поля с предыдущего шага рекурсии  
 for q in add\_fields:  
 res[q[0]] = q[1]  
 for q in list(res.keys()): # идём по ключам  
 link = res[q] # ссылочка на словарь  
 if " " in q: # если есть атрибуты  
 nm, prop = q.split(" ")[0], [w[0][1:] for w in re.finditer(r'\s[^\n<"]+="[^\n<"]\*"', q)]  
 if nm in res.keys(): res[nm].append(res[q]) # добавляем в список  
 else: res[nm] = [res[q]]  
 del res[q] # удалили  
 link = res[nm][-1] # перезаписали ссылку  
 add\_fields = [["\_" + w[:w.index("=")], w[w.index("=") + 1:][1:-1]] for w in prop] # добавление полей  
 else: # для след. шага  
 add\_fields = []  
 self.group\_equal\_tags(link, add\_fields)  
  
 def write\_to\_yaml(self, file, out, curr\_spase, arrow\_before\_first): # рекурсивная запись  
 for q in out.keys(): # идём по ключам  
 st = " " \* curr\_spase + q + ":" # текущий тег  
 if type(out[q]) == str: # если попалась строка => просто текстовое поле  
 st += " '" + out[q] + "'\n" # пишем его  
 if arrow\_before\_first:  
 arrow\_before\_first = False # если с прошлого шага пришла необходимость стрелочки - добавляем  
 st = st[:curr\_spase - 2] + "-" + st[curr\_spase - 1:]  
 file.write(st)  
 if type(out[q]) == dict: # если попался словарь => печатаем тег, переводим строку и идём глубже  
 st += "\n"  
 file.write(st) # файл|словарь|добавили пробел | стрелка если несколько элементов  
 self.write\_to\_yaml(file, out[q], curr\_spase + 2, True if len(out.keys()) > 1 else False)  
 if type(out[q]) == list: # если попался список => печатаем тег, переводим строку и идём по элементам,  
 st += "\n" # каждый раз взводя флаг добавления стрелочки  
 file.write(st)  
 for w in out[q]:  
 if type(w) == str: file.write(" " \* (curr\_spase + 2) + "- " + w + "\n")  
 else: self.write\_to\_yaml(file, w, curr\_spase + 2, True)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(XmlToYamlFull().main(True))

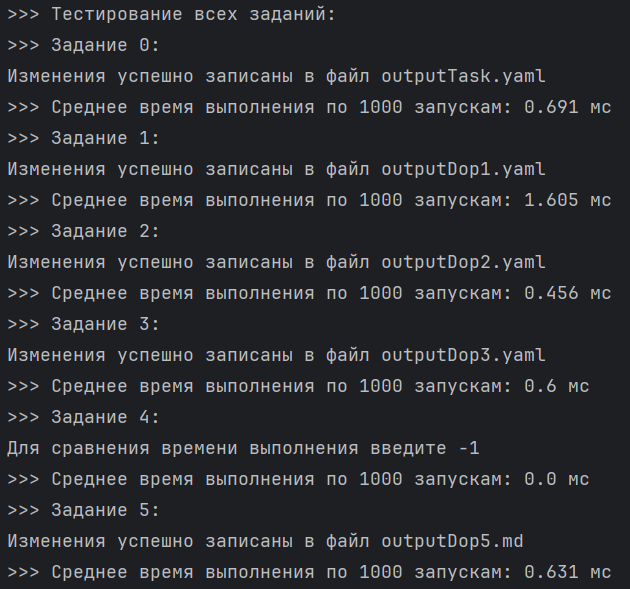
b) Проверка осуществлялась на входных данных, приведённых в описании основного задания, и на входных данных, приведённых ниже:

<?xml version="1.0"?>  
<Контрагенты>  
 <Контрагент Наименование="Контрагент 1" Код="000000001">  
 <Телефон>333-33-33</Телефон>  
 </Контрагент>  
 <Контрагент Наименование="Контрагент 2" Код="000000002"/>  
 <Контрагент Наименование="Контрагент 3" Код="000000003">  
 <Телефон>111-11-11</Телефон>  
 <Телефон>222-22-22</Телефон>  
 </Контрагент>  
</Контрагенты>

с) В обоих случаях программа показала вывод, идентичный готовым парсерам их xml в yaml.

## 5. Дополнительное задание 4

1. Результаты сравнения представлены на Рис. 1
2. Описание времени выполнения подзадач
   1. Наиболее долго выполнялась 2 программа, где для конвертации использовались готовые библиотеки. Это может быть связано с тем, что в библиотеках реализован гораздо больший функционал, который может быть полезен в других прикладных задачах, но в задаче, требующей исключительно конвертации одного формата в другой совершается гораздо больше действий, чем для этого необходимо, следовательно, увеличивается во время выполнения программы.
   2. Следующий по времени выполнения была основная задача лабораторной работы.
   3. Реализация парсинга с использованием регулярных выражений завершила свою работу уже на 200 наносекунд быстрее, так как в Python регулярные выражения реализованы на языке С++, который значительно быстрее.
   4. Решение, требующее использования формальных грамматик работало почти на том же уровне как и решение, требующие использования регулярных выражений, так как для учтения всех особенностей языка нужно большее количество проверок.
   5. Последняя подзадача работает за такое же время, как и конвертация с использованием формальных грамматик, так как в языке Markdown нет сложных синтаксических конструкций, значительно отличающихся от конструкции в языке YAML.

 Рис. 1

## 6. Дополнительное задание 5

a) В качестве языка для данной подзадачи был выбран Markdown, как один из самых популярных легко читаемых языков разметки документов.

Исходный код:

import pprint  
import re  
  
  
class XmlToMarkdown:  
 def main(self, debug=False):  
 file = open("input.xml", encoding="utf-8")  
 data = file.read()  
 file.close()  
 headers = []  
 res = dict()  
 self.parse\_headers(headers, data) # парсим теги  
 self.rec\_add(res, headers, 0) # рекурсивно формируем словарь  
 self.group\_equal\_tags(res, []) # группируем одинаковые теги и выносим их атрибуты  
 if debug: pprint.pprint(res, width=200)  
  
 file = open("outputDop5.md", "w", encoding="utf-8")  
 curr\_spase = 0  
 self.write\_to\_md(file, res, curr\_spase, False) # пишем в yaml  
 file.close()  
 return "Изменения успешно записаны в файл outputDop5.md"  
  
 @staticmethod  
 def parse\_headers(headers, data):  
 for q in re.finditer(r"<[^>]+>", data): # ищем все теги по регулярке  
 headers.append([q[0], q.start(), ""])  
 if re.fullmatch(r"<!--[^>]+-->", headers[-1][0]): headers.pop(-1) # удаляем <!-- комментарии -->  
 if headers[-1][0][-2:] == "/>":  
 headers[-1][0] = headers[-1][0][:-2] + ">" # случай, когда тег пустой, делаем ещё один фиктивный  
 headers.append(["</" + headers[-1][0].split(" ")[0][1:] + ">", headers[-1][1], headers[-1][2]])  
 for q in range(len(headers)): # пишем текстовые данные там, где они есть  
 if headers[q][0][:2] == "</" and headers[q][0][2:] == headers[q - 1][0][1:]:  
 headers[q - 1][2] = data[headers[q - 1][1] + len(headers[q - 1][0]):headers[q][1]]  
 headers.pop(0)  
  
 def rec\_add(self, res: dict, headers: list[list[str | int]], start\_on: int) -> None:  
 if not headers: return  
 if start\_on > len(headers) - 1: return # завершение рекурсии  
 curr = headers[start\_on]  
 curr\_name = curr[0][1:-1]  
 if curr[2]:  
 if curr\_name in res.keys():  
 if type(res[curr\_name]) == list: res[curr\_name].append(self.replace\_entities(curr[2])) # заменяем сущ.  
 else: res[curr\_name] = [res[curr\_name], self.replace\_entities(curr[2])] # и пишем, если строка не пустая  
 else: res[curr\_name] = self.replace\_entities(curr[2])  
 else: res[curr\_name] = dict() # создаём словарь если пустая строка  
 end\_ind = len(headers)  
 for q in range(start\_on, len(headers)):  
 if headers[q][0][2:-1] == curr[0][1:-1].split(" ")[0]: # если два соседних тега, записываем  
 end\_ind = q # стартовую позицию и передаём дальше  
 break  
 self.rec\_add(res[curr\_name], headers[start\_on + 1:end\_ind], 0) # рекурсия вглубь  
 self.rec\_add(res, headers, end\_ind + 1) # и вбок  
  
 @staticmethod  
 def replace\_entities(s):  
 for q in [["&lt;", '<'], ["&gt;", '>'], ["&amp;", '&'], ["&apos;", "'"], ["&quot;", '"']]:  
 s = s.replace(q[0], q[1]) # замена сущностей  
 return s  
  
 def group\_equal\_tags(self, res: dict, add\_fields: list[list]):  
 if type(res) != dict: return  
 if add\_fields: # добавляем поля с предыдущего шага рекурсии  
 for q in add\_fields:  
 res[q[0]] = q[1]  
 for q in list(res.keys()): # идём по ключам  
 link = res[q] # ссылочка на словарь  
 if " " in q: # если есть атрибуты  
 nm, prop = q.split(" ")[0], [w[0][1:] for w in re.finditer(r'\s[^\n<"]+="[^\n<"]\*"', q)]  
 if nm in res.keys(): res[nm].append(res[q]) # добавляем в список  
 else: res[nm] = [res[q]]  
 del res[q] # удалили  
 link = res[nm][-1] # перезаписали ссылку  
 add\_fields = [["\_" + w[:w.index("=")], w[w.index("=") + 1:][1:-1]] for w in prop] # добавление полей  
 else: # для след. шага  
 add\_fields = []  
 self.group\_equal\_tags(link, add\_fields)  
  
 def write\_to\_md(self, file, out, curr\_spase, arrow\_before\_first): # рекурсивная запись  
 for q in out.keys(): # идём по ключам  
 st = "\t" \* curr\_spase + "- " + q + ":" # текущий тег  
 if type(out[q]) == str: # если попалась строка => просто текстовое поле  
 if arrow\_before\_first:  
 arrow\_before\_first = False # если с прошлого шага пришла необходимость стрелочки - добавляем  
 st = st[:curr\_spase + 3 - 1] + "\*\*" + st[curr\_spase + 3 - 1:] + "\*\*"  
 st += " " + out[q] + "\n" # пишем его  
 file.write(st)  
 if type(out[q]) == dict: # если попался словарь => печатаем тег, переводим строку и идём глубже  
 st += "\n"  
 file.write(st) # файл|словарь|добавили пробел | стрелка если несколько элементов  
 self.write\_to\_md(file, out[q], curr\_spase + 1, True if len(out.keys()) > 1 else False)  
 if type(out[q]) == list: # если попался список => печатаем тег, переводим строку и идём по элементам,  
 st += "\n" # каждый раз взводя флаг добавления стрелочки  
 file.write(st)  
 for w in out[q]:  
 if type(w) == str: file.write(" " \* (curr\_spase + 1) + "- " + w + "\n")  
 else: self.write\_to\_md(file, w, curr\_spase + 1, True)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(XmlToMarkdown().main(True))

Выходные данные:

- Timetable:  
 - User: Пшеничников Артём Дмитриевич  
 - UserISU: 467205  
 - Week:  
 - Monday:  
 - Lesson:  
 - \*\*Lector:\*\* Цю Тяньшэн  
 - Theme: Основы профессиональной деятельности  
 - Type: Лабораторная  
 - Classroom: 2112  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 08:20  
 - \*\*Lector:\*\* Цю Тяньшэн  
 - Theme: Основы профессиональной деятельности  
 - Type: Лабораторная  
 - Classroom: 2112  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 10:00  
 - \*\*Lector:\*\* Карпов Дмитрий Валерьевич  
 - Theme: Линейная алгебра (продвинутый уровень)  
 - Type: Лекция  
 - Classroom: 1404  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 15:20  
 - \*\*Lector:\*\* Покидова Марина Владимировна  
 - Theme: Линейная алгебра (продвинутый уровень)  
 - Type: Практика  
 - Classroom: 2426  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 17:00  
 - \*\*Lector:\*\* Рекин Владимир Захарович  
 - Theme: Шашки  
 - Type: Спорт  
 - Classroom: Бильярдный клуб (1784)  
 - Address: Вяземский пер., д.5-7, лит.А  
 - \_Time: 19:00  
 - Thursday:  
 - Lesson:  
 - \*\*Lector:\*\* Щеглова Иванна Павловна  
 - Theme: История русской культуры в контексте мировой культуры  
 - Type: Практика  
 - Classroom: 2412  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 11:40  
 - \*\*Lector:\*\* Жиркова Галина Петровна  
 - Theme: История русской культуры в контексте мировой культуры  
 - Type: Лекция  
 - Classroom: 1405  
 - Address: Кронверкский пр., д.49, лит.А  
 - \_Time: 13:30

Скомпилированный файл:

* Timetable:
  + User: Пшеничников Артём Дмитриевич
  + UserISU: 467205
  + Week:
    - Monday:
      * Lesson:
        + **Lector:** Цю Тяньшэн
        + Theme: Основы профессиональной деятельности
        + Type: Лабораторная
        + Classroom: 2112
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 08:20
        + **Lector:** Цю Тяньшэн
        + Theme: Основы профессиональной деятельности
        + Type: Лабораторная
        + Classroom: 2112
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 10:00
        + **Lector:** Карпов Дмитрий Валерьевич
        + Theme: Линейная алгебра (продвинутый уровень)
        + Type: Лекция
        + Classroom: 1404
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 15:20
        + **Lector:** Покидова Марина Владимировна
        + Theme: Линейная алгебра (продвинутый уровень)
        + Type: Практика
        + Classroom: 2426
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 17:00
        + **Lector:** Рекин Владимир Захарович
        + Theme: Шашки
        + Type: Спорт
        + Classroom: Бильярдный клуб (1784)
        + Address: Вяземский пер., д.5-7, лит.А
        + \_Time: 19:00
    - Thursday:
      * Lesson:
        + **Lector:** Щеглова Иванна Павловна
        + Theme: История русской культуры в контексте мировой культуры
        + Type: Практика
        + Classroom: 2412
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 11:40
        + **Lector:** Жиркова Галина Петровна
        + Theme: История русской культуры в контексте мировой культуры
        + Type: Лекция
        + Classroom: 1405
        + Address: Кронверкский пр., д.49, лит.А
        + \_Time: 13:30

b) Готовых конверторов такого типа нет, так как сложно перенести всю информацию в такой довольно простой формат предствления данных, но тем не менее, markdown позволяет представлять сложные древовидные структуры, ссылки внутри файла и также различные уровни заголовков.

# Заключение

В ходе проделанной лабораторной работы я узнал о различных языках гипертекстовой разметки текста, познакомился с их синтаксисом, а также написал свой парсер из языка xml в язык yaml с различными реализациями. В итоге был получен парсер, учитывающий все формальные грамматики, исходного языка и работающий на уровне с готовыми решениями. В дальнейшем данный парсер может быть доработан, для конвертации формата xml в любой из форматов, представимых в json.

# Список литературы

1. Сервис тестирования регулярных выражений / [Электронный ресурс]. – Режим доступа: <https://regex101.com/>
2. Регулярные выражения в Python от простого к сложному / [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/349860>
3. Онлайн парсер различных форматов разметки теста / [Электронный ресурс]. – Режим доступа: <https://jsonformatter.org>
4. Официальная документация по библиотеке re / [Электронный ресурс]. – Режим доступа: <https://docs.python.org/3/library/re.html>