Содержание

Задание

Обязательное задание. Написать программу на языке Python, которая: осуществляет парсинг и конвертацию исходного файла в бинарный объект (=десериализацию) и не использовует готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки файлов.

Дополнительное задание №1. Написать программу на языке Python, которая: осуществляет парсинг и конвертацию бинароного объекта, полученного в обязательном задании, в новый формат (=сериализацию) и не использовует готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки файлов.

Дополнительное задание №2. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов (десериализацию и сериализацию). Переписать исходный код и код из дополнительного задания №1, применив найденные библиотеки. Регулярные выражения также нельзя использовать. Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

Дополнительное задание №3. Переписать код из дополнительного задания №1, чтобы серализация происводилась в XML файл.

Дополнительное задание №4. Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле. Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

Основные этапы решения

Входной файл JSON.


```
{
  "Tuesday" : [
    {
      "Lesson" : "Высшая алгебра (лек)",
      "Time" : "11:30-13:00",
      "Weeks" : [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
      "Group" : "ВышАлг 11",
      "Room" : "1419"
    },
    {
      "Lesson" : "Высшая алгебра (прак)",
      "Time" : "13:30-15:00",
      "Weeks" : [2,3,4,5,6,7,8,10,11,12,13,14,15,16,17],
      "Group" : "ВышАлг 11.2",
      "Room" : "1327"
    },
    {
      "Lesson" : "Математический анализ (лек)",
      "Time" : "15:30-17:00",
      "Weeks" : [2,3,4,5,6,7,8,10,11,12,13,14,15,16,17],
      "Group" : "МатАнПрод 11",
      "Room" : "2433"
    },
    {
      "Lesson" : "Математический анализ (прак)",
      "Time" : "17:10-18:40",
      "Weeks" : [2,3,4,5,6,7,8,10,11,12,13,14,15,16,17],
      "Group" : "МатАнПрод 11.1",
      "Room" : "2306/1"
    }
  ],
  "Thursday" : [
    {
      "Lesson" : "Информатика (лаб)",
      "Time" : "11:30-13:00",
      "Weeks" : [2,4,6,8,10,12,14,16],
      "Group" : "ИНФОРМ (P3115) 1.10",
      "Room" : "1328"
    },
    {
      "Lesson" : "Информатика (лаб)",
```

```
      "Time" : "13:30-15:00",
      "Weeks" : [2,4,6,8,10,12,14,16],
      "Group" : "ИНФОРМ (Р3115) 1.10",
      "Room" : "1328"
    },
    {
      "Lesson" : "Основы профессиональной деятельности (лаб)",
      "Time" : "15:30-17:00",
      "Weeks" : [2,4,6,8,10,12,14,16],
      "Group" : "ОПД (Р3115) 1.10",
      "Room" : "2435/3"
    },
    {
      "Lesson" : "Основы профессиональной деятельности (лаб)",
      "Time" : "17:10-18:40",
      "Weeks" : [2,4,6,8,10,12,14,16],
      "Group" : "ОПД (Р3115) 1.10",
      "Room" : "2435/3"
    },
    {
      "Lesson" : "История российской науки и техники (прак)",
      "Time" : "11:30-13:00",
      "Weeks" : [1,3,5,7,9,11,13,15],
      "Group" : "ИНТ 1.3",
      "Room" : "2326"
    },
    {
      "Lesson" : "История российской науки и техники (лек)",
      "Time" : "13:30-15:00",
      "Weeks" : [1,3,5,7,9,11,13,15],
      "Group" : "ИНТ 1",
      "Room" : "1404"
    }
  ]
}
```

Десериализация файла JSON**.**

```
def json_to_bin(file_name: str) -> None:
    try:
        if file_name.split(".")[-1] != "json":
            print("Ошибка: тип файла должен быть json")
            exit()
        with open(file_name, "r", encoding="utf-8") as file:
            json_file = file.read()
        json_file = parse_value(json_file)
        bin_file = str()
        for i in json_file:
            character = bin(ord(i))[2:]
            bin_file += character + " "
        bin_file = str.encode(bin_file[:-1])
        with open(".".join(file_name.split(".")[:-1])+".bin", "wb") as file:
            file.write(bin_file)
    except Exception as e:
        print("Ошибка: ", e)
```

```
def parse_value(value: str) -> str:
    value = clear_spaces(value)
    new_value = str()
    try:
        value = clear_spaces(value)
        if value[0] == "{" and value[-1] == "}":
            new_value += "o"
            value = parse_object(value)
        elif value[0] == "[" and value[-1] == "]":
            new_value += "a"
            value = parse_array(value)
        elif value[0] == '"' and value[-1] == '"':
            new_value += "s"
            value = parse_string(value)
        elif all([i in "+-.0123456789eE" for i in value]):
            new_value += "n"
            value = parse_number(value)
        elif value == "true":
            value = "t"
        elif value == "false":
            value = "f"
        elif value == "null":
            value = "v"
        else:
            print("Ошибка при определении типа элемента: невозможно определить
тип: ", value)
            exit()
        new_value += value
        return new_value
    except Exception as e:
        print("Ошибка при определении типа элемента: ", e)


def parse_object(object: str) -> str:
    def error():
        print("Ошибка в определении объекта: последовательность " + object + " не
является объектом")
        exit()
    try:
        if len(object) <= 2:
            object = "0"
        else:
            object = object[1:-1]
            new_object = dict()
            while len(object) > 0:
                object = clear_spaces(object)
                key = str()
                value = str()
                if object[0] == '"':
                    for i in range(1, len(object)):
                        if object[i] == '"':
                            if object[i-1] == "\\":
                                continue
                            key = object[:i+1]
                            object = object[i+1:]
                            break
                        elif i + 2 == len(object): error()
                object = clear_spaces(object)
```

```
if len(object) == 0: error()
if object[0] != ":": error()
object = clear_spaces(object[1:])
if object.count(",") == 0:
    value = object
if len(object) == 1: error()
if object[0] not in '{["':
    value = object[:object.index(",")]
    object = object[object.index(",")+1:]
elif object[0] == "{":
    c = 0
    for i in range(len(object)):
        if object[i] == "{":
            c += 1
        elif object[i] == "}":
            c -= 1
        if c == 0:
            value = object[:i + 1]
            object = clear_spaces(object[i + 1:])
            if object[0] == ",":
                object = clear_spaces(object[1:])
            break
        elif i + 2 == len(object):
            value = object
            object = ""
            break
elif object[0] == "[":
    c = 0
    for i in range(len(object)-1):
        if object[i] == "[":
            c += 1
        elif object[i] == "]":
            c -= 1
        if c == 0:
            value = object[:i + 1]
            object = clear_spaces(object[i + 1:])
            if object[0] == ",":
                object = clear_spaces(object[1:])
            break
        elif i + 2 == len(object):
            value = object
            object = ""
            break
elif object[0] == '"':
    for i in range(1, len(object)):
        if object[i] == '"':
            if object[i-1] == "\\":
                continue
            value = object[:i+1]
            object = clear_spaces(object[i+1:])
            if object[0] == ",":
                object = clear_spaces(object[1:])
            break
        elif i + 2 == len(object):
            value = object
            object = ""
            break
else: error()
new_object.setdefault(key, value)
```

```
            object = str(len(new_object.keys()))
            for i in [[list(new_object.keys())[i], list(new_object.values())[i]]
for i in range(len(new_object.keys()))]:
                key = parse_value(i[0])
                value = parse_value(i[1])
                object += " " + key + " " + value
        return object
    except Exception as e:
        print("Ошибка при определении объекта: ", e)


def parse_array(array: str) -> str:
    def error():
        print("Ошибка в определении массива: последовательность " + array + " не
является массивом")
        exit()
    try:
        if len(array) <= 2:
            array = "0"
        else:
            new_array = list()
            array = array[1:-1]
            while len(array) > 0:
                array = clear_spaces(array)
                if array.count(",") == 0:
                    new_array.append(array)
                    break
                if len(array) == 1: error()
                if array[0] not in '{["':
                    new_array.append(array[:array.index(",")])
                    array = array[array.index(",")+1:]
                elif array[0] == "{":
                    c = 0
                    for i in range(len(array)):
                        if array[i] == "{":
                            c += 1
                        elif array[i] == "}":
                            c -= 1
                        if c == 0:
                            new_array.append(array[:i + 1])
                            array = clear_spaces(array[i + 1:])
                            if array[0] == ",":
                                array = clear_spaces(array[1:])
                            break
                        elif i + 2 == len(array):
                            new_array.append(array)
                            array = ""
                            break
                elif array[0] == "[":
                    c = 0
                    for i in range(len(array)):
                        if array[i] == "[":
                            c += 1
                        elif array[i] == "]":
                            c -= 1
                        if c == 0:
                            new_array.append(array[:i + 1])
                            array = clear_spaces(array[i + 1:])
                            if array[0] == ",":
```

```
                            array = clear_spaces(array[1:])
                        break
                    elif i + 2 == len(array):
                        new_array.append(array)
                        array = ""
                        break
            elif array[0] == '"':
                for i in range(1, len(array)):
                    if array[i] == '"':
                        if array[i-1] == "\\":
                            continue
                        new_array.append(array[:i+1])
                        array = array[i+1:]
                        break
                    elif i + 2 == len(array):
                        new_array.append(array)
                        array = ""
                        break
            else: error()
        array = str(len(new_array))
        for i in new_array:
            array += " " + parse_value(i)
    return array
except Exception as e:
    print("Ошибка при определении массива: ", e)


def parse_string(string: str) -> str:
    def error():
        print("Ошибка в определении строки: последовательность " + string + " не
является строкой")
        exit()
    try:
        for i in range(len(string)-1):
            if string[i] == "\\" and i != len(string) - 1:
                if string[i+1] not in '"\\/bfnrt':
                    continue
                elif string[i+1] == "u" and i + 6 < len(string):
                    for j in range(4):
                        if string[i+2+j].lower() not in "0123456789abcdef":
error()
                else: error()
        return string
    except Exception as e:
        print("Ошибка при определении строки: ", e)


def parse_number(number: str) -> str:
    def error():
        print("Ошибка в определении числа: последовательность " + number + " не
является числом")
        exit()
    checked_number = str()
    try:
        if number.lower()[-1] == "e": error()
        if "e" in number.lower():
            num1, num2 = number[:number.lower().find("e")],
number[number.lower().find("e") + 1:]
        else:
```

```
            num1, num2 = number, "not"
        if all([i in "-0123456789." for i in num1]):
            if num1.count(".") == 0 and num1.count("-") == 0:
                if len(num1) > 0:
                    checked_number += num1
                else: error()
            elif num1.count(".") == 1 and num1.count("-") == 1:
                if len(num1) > 2 and num1[0] == "-" and num1[-1] != ".":
                    checked_number += num1
                else: error()
            elif num1.count("-") == 1:
                if len(num1) > 1 and num1[0] == "-":
                    checked_number += num1
                else: error()
            elif num1.count(".") == 1:
                if len(num1) > 1 and num1[0] != "." and num1[-1] != ".":
                    checked_number += num1
                else: error()
            else: error()
        else: error()
        if num2 != "not":
            checked_number += "e"
            if all([i in "+-0123456789" for i in num2]):
                if (num2.count("+") + num2.count("-")) == 1 and len(num2) > 1:
                    if num2[0] == "+" or num2[0] == "-":
                        checked_number += num2
                    else: error()
                elif num2.count("+") + num2.count("-") == 0:
                    checked_number += num2
                else: error()
            else: error()
        return checked_number
    except Exception as e:
        print("Ошибка при определении числа: ", e)


def clear_spaces(line: str) -> str:
    while line[0] in "\r\t\n ":
        line = line[1:]
    while line[-1] in "\r\t\n ":
        line = line[:-1]
    return line


if __name__ == "__main__":
    json_to_bin("schedule.json")
```

Сериализация файла YAML.

```
def bin_to_yaml(file_name):
    try:
        if file_name.split(".")[-1] != "bin":
            print("Ошибка: тип файла должен быть bin")
            exit()
        with open(file_name, "rb") as file:
```

```
            bin_file = file.read()
        yaml_file = str()
        bin_file = bin_file.decode().split(" ")
        for i in bin_file:
            yaml_file += chr(int(i, 2))
        yaml_file = unparse_value(yaml_file)
        with open(".".join(file_name.split(".")[:-1])+".yaml", "w",
encoding="utf-8") as file:
            file.write(yaml_file)
    except Exception as e:
        print("Ошибка: ", e)


def unparse_value(value: str) -> str:
    try:
        new_value = str()
        while len(value) != 0:
            if value[0] == "o":
                object, value = unparse_object(value)
                new_value += object
            elif value[0] == "a":
                array, value = unparse_array(value)
                new_value += array
            elif value[0] == "s":
                for i in range(2, len(value)-1):
                    if value[i] == '"' and value[i-1] != "\\":
                        new_value += value[2:i-1]
                        if len(value[i:]) > 2:
                            value = value[i+2:]
                        else: value = ""
                        break
                    elif i + 1 == len(value)-1:
                        new_value += value[2:len(value)-1]
                        value = ""
            elif value[0] == "n":
                if value.count(" ") == 0:
                    new_value += value[1:] + "\n"
                    value = ""
                else:
                    new_value += value[1:value.index(" ")]
                    value = value[value.index(" ")+1:]
            elif value[0] == "t":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "true" + "\n"
            elif value[0] == "f":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "false" + "\n"
            elif value[0] == "v":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "null" + "\n"
            else:
                print("Ошибка в определении типа элемента " + value)
                exit()
        return new_value
    except Exception as e:
        print("Ошибка при определении типа элемента: ", e)
```

```
def unparse_object(object: str) -> tuple[str, str]:
    def error():
        print("Ошибка в определении объекта " + object)
        exit()
    try:
        result = dict()
        index, object = object[1:object.index(" ")], object[object.index(" ")+1:]
        for _ in range(int(index)):
            key = str()
            if object[0] == "s":
                for i in range(2, len(object)-1):
                    if object[i] == '"' and object[i-1] != "\\":
                        key = object[2:i]
                        object = object[i+2:]
                        break
            else: error()
            if object.count(" ") == 0:
                value = unparse_value(object)
                object = ""
            elif object[0] in "ntfv":
                value = object[1:object.index(" ")]
                object = object[object.index(" ")+1:]
                value = unparse_value(value)
            elif object[0] == "s":
                for i in range(2, len(object)-1):
                    if object[i] == '"' and object[i-1] != "\\":
                        value = object[2:i]
                        if len(object[i:]) > 2:
                            object = object[i+2:]
                        else:
                            object = ""
                        break
            elif object[0] == "a":
                value, object = unparse_array(object)
                value = "\n" + value
            elif object[0] == "o":
                value, object = unparse_object(object)
                value = "\n" + value
                flag = False
                new_value = str()
                for i in range(len(value)-1):
                    if flag:
                        if value[i] == '"' and value[i - 1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i - 1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "  "
                value = new_value
            else: error()
            result.setdefault(key, value)
        new_result = str()
        for i in [[list(result.keys())[i], list(result.values())[i]] for i in
range(len(result.keys()))]:
```

```
            key = i[0]
            value = i[1]
            new_result += key + ": " + value + "\n"
        result = new_result
        return result, object
    except Exception as e:
        print("Ошибка при определении объекта: ", e)


def unparse_array(array: str) -> tuple[str, str]:
    try:
        result = list()
        index, array = array[1:array.index(" ")], array[array.index(" ")+1:]
        for _ in range(int(index)):
            if array.count(" ") == 0:
                result.append(unparse_value(array))
                array = ""
            elif array[0] in "ntfv":
                value = array[:array.index(" ")]
                array = array[array.index(" ")+1:]
                result.append(unparse_value(value))
            elif array[0] == "s":
                for i in range(2, len(array)-1):
                    if array[i] == '"' and array[i-1] != "\\":
                        result.append(array[2:i])
                        if len(array[i:]) > 2:
                            array = array[i+2:]
                        else:
                            array = ""
                        break
            elif array[0] == "a":
                value, array = unparse_array(array)
                flag = False
                new_value = str()
                for i in range(len(value)):
                    if flag:
                        if value[i] == '"' and value[i - 1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i - 1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "  "
                value = new_value + "\n"
                result.append(value)
            elif array[0] == "o":
                value, array = unparse_object(array)
                flag = False
                new_value = str()
                for i in range(len(value)-1):
                    if flag:
                        if value[i] == '"' and value[i-1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i-1] != "\\":
                        flag = True
                        continue
```

```
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "  "
                value = new_value + "\n"
                result.append(value)
            else:
                print("Ошибка в определении массива " + str(result))
                exit()
        new_result = str()
        for i in result:
            new_result += "- " + i
        result = new_result
        return result[:-1], array
    except Exception as e:
        print("Ошибка при определении массива: ", e)


if __name__ == "__main__":
    bin_to_yaml("schedule.bin")
```

Выходной файл YAML.

```
Tuesday:
- Lesson: Высшая алгебра (лек)
  Time: 11:30-13:00
  Weeks:
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 9
  - 10
  - 11
  - 12
  - 13
  - 14
  - 15
  - 16
  - 17
  Group: ВышАлг 11
  Room: 1419
- Lesson: Высшая алгебра (прак)
  Time: 13:30-15:00
  Weeks:
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 10
  - 11
```

```
    - 12
    - 13
    - 14
    - 15
    - 16
    - 17
    Group: ВышАлг 11.2
    Room: 1327
  - Lesson: Математический анализ (лек)
    Time: 15:30-17:00
    Weeks:
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 10
    - 11
    - 12
    - 13
    - 14
    - 15
    - 16
    - 17
    Group: МатАнПрод 11
    Room: 2433
  - Lesson: Математический анализ (прак)
    Time: 17:10-18:40
    Weeks:
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 10
    - 11
    - 12
    - 13
    - 14
    - 15
    - 16
    - 17
    Group: МатАнПрод 11.1
    Room: 2306/1
  Thursday:
  - Lesson: Информатика (лаб)
    Time: 11:30-13:00
    Weeks:
    - 2
    - 4
    - 6
    - 8
    - 10
    - 12
    - 14
```

```
  - 16
  Group: ИНФОРМ (Р3115) 1.10
  Room: 1328
- Lesson: Информатика (лаб)
  Time: 13:30-15:00
  Weeks:
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  Group: ИНФОРМ (Р3115) 1.10
  Room: 1328
- Lesson: Основы профессиональной деятельности (лаб)
  Time: 15:30-17:00
  Weeks:
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  Group: ОПД (Р3115) 1.10
  Room: 2435/3
- Lesson: Основы профессиональной деятельности (лаб)
  Time: 17:10-18:40
  Weeks:
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  Group: ОПД (Р3115) 1.10
  Room: 2435/3
- Lesson: История российской науки и техники (прак)
  Time: 11:30-13:00
  Weeks:
  - 1
  - 3
  - 5
  - 7
  - 9
  - 11
  - 13
  - 15
  Group: ИНТ 1.3
  Room: 2326
- Lesson: История российской науки и техники (лек)
  Time: 13:30-15:00
  Weeks:
  - 1
```

```
  - 3
  - 5
  - 7
  - 9
  - 11
  - 13
  - 15
  Group: ИНТ 1
  Room: 1404
```

Использование библиотек.

```
import json
import yaml

def parse_json(file_name):
    try:
        if file_name.split(".")[-1] != "json":
            print("Ошибка: тип файла должен быть json")
            exit()
        with open(file_name, "r", encoding="utf-8") as file:
            json_file = file.read()
        bin_file = json.dumps(json_file).encode("utf-8")
        with open(".".join(file_name.split(".")[:-1])+"(lib).bin", "wb") as file:
            file.write(bin_file)
    except Exception as e:
        print("Ошибка: ", e)


def unparse_bin(file_name):
    try:
        if file_name.split(".")[-1] != "bin":
            print("Ошибка: тип файла должен быть bin")
            exit()
        with open(file_name, "rb") as file:
            bin_file = file.read()
        yaml_file = yaml.safe_load(yaml.safe_load(bin_file))
        yaml_file = yaml.dump(yaml_file, allow_unicode=True)
        with open(".".join(file_name.split(".")[:-1])+".yaml", "w",
encoding="utf-8") as file:
            file.write(yaml_file)
    except Exception as e:
        print("Ошибка :", e)


def json_to_yaml(file_name):
    try:
        parse_json(file_name)
        unparse_bin(".".join(file_name.split(".")[:-1])+"(lib).bin")
    except Exception as e:
        print("Ошибка: ", e)


if __name__ == "__main__":
    json_to_yaml("schedule.json")
```

Десериализация в XML.

```
def bin_to_xml(file_name):
    try:
        if file_name.split(".")[-1] != "bin":
            print("Ошибка: тип файла должен быть bin")
            exit()
        with open(file_name, "rb") as file:
            bin_file = file.read()
        xml_file = str()
        bin_file = bin_file.decode().split(" ")
        for i in bin_file:
            xml_file += chr(int(i, 2))
        value = unparse_value(xml_file)
        flag = False
        new_value = str()
        for i in range(len(value) - 1):
            if flag:
                if value[i] == '"' and value[i - 1] != "\\":
                    flag = False
                continue
            elif value[i] == '"' and value[i - 1] != "\\":
                flag = True
                continue
            new_value += value[i]
            if value[i] == "\n":
                new_value += "    "
        xml_file = "<root>\n    " + new_value + "\n</root>\n"
        with open(".".join(file_name.split(".")[:-1])+".xml", "w",
encoding="utf-8") as file:
            file.write(xml_file)
    except Exception as e:
        print("Ошибка: ", e)


def unparse_value(value: str) -> str:
    try:
        new_value = str()
        while len(value) != 0:
            if value[0] == "o":
                object, value = unparse_object(value)
                new_value += object
            elif value[0] == "a":
                array, value = unparse_array(value)
                new_value += array
            elif value[0] == "s":
                for i in range(2, len(value)-1):
                    if value[i] == '"' and value[i-1] != "\\":
                        new_value += value[2:i-1]
                        if len(value[i:]) > 2:
                            value = value[i+2:]
                        else: value = ""
                        break
                    elif i + 1 == len(value)-1:
                        new_value += value[2:len(value)-1]
                        value = ""
            elif value[0] == "n":
                if value.count(" ") == 0:
```

```
                    new_value += value[1:]
                    value = ""
                else:
                    new_value += value[1:value.index(" ")]
                    value = value[value.index(" ")+1:]
            elif value[0] == "t":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "true"
            elif value[0] == "f":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "false"
            elif value[0] == "v":
                if len(value) == 1: value = ""
                else: value = value[2:]
                new_value += "null"
            else:
                print("Ошибка в определении типа элемента " + value)
                exit()
        return new_value
    except Exception as e:
        print("Ошибка при определении типа элемента: ", e)


def unparse_object(object: str) -> tuple[str, str]:
    def error():
        print("Ошибка в определении объекта " + object)
        exit()
    try:
        result = dict()
        index, object = object[1:object.index(" ")], object[object.index(" ")+1:]
        for _ in range(int(index)):
            key = str()
            if object[0] == "s":
                for i in range(2, len(object)-1):
                    if object[i] == '"' and object[i-1] != "\\":
                        key = object[2:i]
                        object = object[i+2:]
                        break
            else: error()
            if object.count(" ") == 0:
                value = unparse_value(object)
                object = ""
            elif object[0] in "ntfv":
                value = object[1:object.index(" ")]
                object = object[object.index(" ")+1:]
                value = unparse_value(value)
            elif object[0] == "s":
                for i in range(2, len(object)-1):
                    if object[i] == '"' and object[i-1] != "\\":
                        value = object[2:i]
                        if len(object[i:]) > 2:
                            object = object[i+2:]
                        else:
                            object = ""
                        break
            elif object[0] == "a":
                value, object = unparse_array(object)
```

```
                value = "\n" + value + "\n"
                flag = False
                new_value = str()
                for i in range(len(value) - 1):
                    if flag:
                        if value[i] == '"' and value[i - 1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i - 1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "    "
                value = new_value + "\n"
            elif object[0] == "o":
                value, object = unparse_object(object)
                value = "\n" + value + "\n"
                flag = False
                new_value = str()
                for i in range(len(value)-1):
                    if flag:
                        if value[i] == '"' and value[i - 1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i - 1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "    "
                value = new_value[:-4]
            else: error()
            result.setdefault(key, value)
        new_result = str()
        for i in [[list(result.keys())[i], list(result.values())[i]] for i in
range(len(result.keys()))]:
            key = i[0]
            value = i[1]
            new_result += "<" + key + ">" + value + "</" + key + ">\n"
        result = new_result
        return result, object
    except Exception as e:
        print("Ошибка при определении объекта: ", e)


def unparse_array(array: str) -> tuple[str, str]:
    try:
        result = list()
        index, array = array[1:array.index(" ")], array[array.index(" ")+1:]
        for _ in range(int(index)):
            if array.count(" ") == 0:
                result.append(unparse_value(array))
                array = ""
            elif array[0] in "ntfv":
                value = array[:array.index(" ")]
                array = array[array.index(" ")+1:]
                result.append(unparse_value(value))
            elif array[0] == "s":
```

```
                for i in range(2, len(array)-1):
                    if array[i] == '"' and array[i-1] != "\\":
                        result.append("<element>" + array[2:i] + "</element>\n")
                        if len(array[i:]) > 2:
                            array = array[i+2:]
                        else:
                            array = ""
                        break
            elif array[0] == "a":
                value, array = unparse_array(array)
                flag = False
                new_value = str()
                for i in range(len(value)):
                    if flag:
                        if value[i] == '"' and value[i - 1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i - 1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "    "
                value = "\n    " + new_value + "\n"
                result.append(value)
            elif array[0] == "o":
                value, array = unparse_object(array)
                flag = False
                new_value = str()
                for i in range(len(value)-1):
                    if flag:
                        if value[i] == '"' and value[i-1] != "\\":
                            flag = False
                        continue
                    elif value[i] == '"' and value[i-1] != "\\":
                        flag = True
                        continue
                    new_value += value[i]
                    if value[i] == "\n":
                        new_value += "    "
                value = "\n    " + new_value + "\n"
                result.append(value)
            else:
                print("Ошибка в определении массива " + str(result))
                exit()
        new_result = str()
        for i in result:
            new_result += "<element>" + i + "</element>\n"
        result = new_result
        return result[:-1], array
    except Exception as e:
        print("Ошибка при определении массива: ", e)


if __name__ == "__main__":
    bin_to_xml("schedule.bin")
```

Выходной файл XML.

```
<root>
    <Tuesday>
        <element>
            <Lesson>Высшая алгебра (лек)</Lesson>
            <Time>11:30-13:00</Time>
            <Weeks>
                <element>2</element>
                <element>3</element>
                <element>4</element>
                <element>5</element>
                <element>6</element>
                <element>7</element>
                <element>8</element>
                <element>9</element>
                <element>10</element>
                <element>11</element>
                <element>12</element>
                <element>13</element>
                <element>14</element>
                <element>15</element>
                <element>16</element>
                <element>17</element>
            </Weeks>
            <Group>ВышАлг 11</Group>
            <Room>1419</Room>
        </element>
        <element>
            <Lesson>Высшая алгебра (прак)</Lesson>
            <Time>13:30-15:00</Time>
            <Weeks>
                <element>2</element>
                <element>3</element>
                <element>4</element>
                <element>5</element>
                <element>6</element>
                <element>7</element>
                <element>8</element>
                <element>10</element>
                <element>11</element>
                <element>12</element>
                <element>13</element>
                <element>14</element>
                <element>15</element>
                <element>16</element>
                <element>17</element>
            </Weeks>
            <Group>ВышАлг 11.2</Group>
            <Room>1327</Room>
        </element>
        <element>
            <Lesson>Математический анализ (лек)</Lesson>
            <Time>15:30-17:00</Time>
            <Weeks>
                <element>2</element>
                <element>3</element>
                <element>4</element>
                <element>5</element>
                <element>6</element>
```

```
            <element>7</element>
            <element>8</element>
            <element>10</element>
            <element>11</element>
            <element>12</element>
            <element>13</element>
            <element>14</element>
            <element>15</element>
            <element>16</element>
            <element>17</element>
        </Weeks>
        <Group>МатАнПрод 11</Group>
        <Room>2433</Room>
    </element>
    <element>
        <Lesson>Математический анализ (прак)</Lesson>
        <Time>17:10-18:40</Time>
        <Weeks>
            <element>2</element>
            <element>3</element>
            <element>4</element>
            <element>5</element>
            <element>6</element>
            <element>7</element>
            <element>8</element>
            <element>10</element>
            <element>11</element>
            <element>12</element>
            <element>13</element>
            <element>14</element>
            <element>15</element>
            <element>16</element>
            <element>17</element>
        </Weeks>
        <Group>МатАнПрод 11.1</Group>
        <Room>2306/1</Room>
    </element>
</Tuesday>
<Thursday>
    <element>
        <Lesson>Информатика (лаб)</Lesson>
        <Time>11:30-13:00</Time>
        <Weeks>
            <element>2</element>
            <element>4</element>
            <element>6</element>
            <element>8</element>
            <element>10</element>
            <element>12</element>
            <element>14</element>
            <element>16</element>
        </Weeks>
        <Group>ИНФОРМ (Р3115) 1.10</Group>
        <Room>1328</Room>
    </element>
    <element>
        <Lesson>Информатика (лаб)</Lesson>
        <Time>13:30-15:00</Time>
        <Weeks>
```

```
            <element>2</element>
            <element>4</element>
            <element>6</element>
            <element>8</element>
            <element>10</element>
            <element>12</element>
            <element>14</element>
            <element>16</element>
        </Weeks>
        <Group>ИНФОРМ (Р3115) 1.10</Group>
        <Room>1328</Room>
    </element>
    <element>
        <Lesson>Основы профессиональной деятельности (лаб)</Lesson>
        <Time>15:30-17:00</Time>
        <Weeks>
            <element>2</element>
            <element>4</element>
            <element>6</element>
            <element>8</element>
            <element>10</element>
            <element>12</element>
            <element>14</element>
            <element>16</element>
        </Weeks>
        <Group>ОПД (Р3115) 1.10</Group>
        <Room>2435/3</Room>
    </element>
    <element>
        <Lesson>Основы профессиональной деятельности (лаб)</Lesson>
        <Time>17:10-18:40</Time>
        <Weeks>
            <element>2</element>
            <element>4</element>
            <element>6</element>
            <element>8</element>
            <element>10</element>
            <element>12</element>
            <element>14</element>
            <element>16</element>
        </Weeks>
        <Group>ОПД (Р3115) 1.10</Group>
        <Room>2435/3</Room>
    </element>
    <element>
        <Lesson>История российской науки и техники (прак)</Lesson>
        <Time>11:30-13:00</Time>
        <Weeks>
            <element>1</element>
            <element>3</element>
            <element>5</element>
            <element>7</element>
            <element>9</element>
            <element>11</element>
            <element>13</element>
            <element>15</element>
        </Weeks>
        <Group>ИНТ 1.3</Group>
        <Room>2326</Room>
```

```
        </element>
        <element>
            <Lesson>История российской науки и техники (лек)</Lesson>
            <Time>13:30-15:00</Time>
            <Weeks>
                <element>1</element>
                <element>3</element>
                <element>5</element>
                <element>7</element>
                <element>9</element>
                <element>11</element>
                <element>13</element>
                <element>15</element>
            </Weeks>
            <Group>ИНТ 1</Group>
            <Room>1404</Room>
        </element>
    </Thursday>
</root>
```

Сравнение времени выполнения.

```
from time import time
from task1 import json_to_bin
from task2 import bin_to_yaml
from task3 import json_to_yaml
from task4 import bin_to_xml


try:
    timer = time()
    for i in range(100):
        json_to_bin("schedule.json")
        bin_to_yaml("schedule.bin")
    print("json -> yaml")
    print(time() - timer)

    timer = time()
    for i in range(100):
        json_to_yaml("schedule.json")
    print("json -> yaml (with libraries)")
    print(time() - timer)

    timer = time()
    for i in range(100):
        json_to_bin("schedule.json")
        bin_to_xml("schedule.bin")
    print("json -> xml")
    print(time() - timer)
except Exception as error:
    print(f"Ошибка: {error}")
```

Примерным выводом данной программы будут следующие занчения.

```
json -> yaml
0.3624873161315918
```

```
json -> yaml (with libraries)
1.1907222270965576
json -> xml
0.6629934310913086
```

Мы видим, что конвертация с помощью библиотек происходит примерно в 3 раза медленнее, чем самописным кодом. Это объясняется тем, что в библиотеках есть более сложные алгоритмы, специализированные для конкретных задач, что делает их более медленными на небольших файлах, но более быстрыми на расширенных файлах. В то время как самописный код сделан без использования каких-либо оптимизационных алгоритмов.

Вывод

Во время данной лабораторной работы были изучены различные языки разметки, разница между ними, а также способы сериализации и десериализации данных.