Содержание

Задание

Основное задание:

Написать программу на языке Python 3.x или любом другом, которая:

- Осуществляет парсинг и конвертацию исходного файла в бинарный объект;
- 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;
- Не использует готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки файлов.

Дополнительное задание №1:

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.

Дополнительное задание №2:

- Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов;
- Переписать исходный код и код из дополнительного задания №1, применив найденные библиотеки;
- Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

Дополнительное задание №3:

- Переписать код из дополнительного задания №1, чтобы сериализация производилась 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 выполнения

Основное задание

Необходимо написать программу на языке Python, которая выполняет десериализацию исходного файла в бинарный объект. Исходным объектом является составленный заранее файл с расписанием в формате json, определённом вариантом.

Исходный файл в формате json, составленный ранее:


```
{
  "Tuesday" : [
    {
      "Lesson" : "Высшая алгебра (лек)",
      "Time" : "11:30-13:00",
      "Weeks" : [1e,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"
    }
  ]
}
```


Написанная для его десериализации программа:

```
def json_error(type: int) -> None:  33 usages
    types = ["invalid string",
             "invalid order of elements",
             "invalid number",
             "invalid boolean",
             "invalid object",
             "invalid array",
             "invalid output"]
    print(f"not a valid JSON file, {types[type-1]}")
    exit()
```

```
def clear_spaces(line: str) -> str:  3 usages
    while line[0] in "\r\t\n ":
        line = line[1:]
    while line[-1] in "\r\t\n ":
        line = line[:-1]
    return line
```

```
def parse_json(file: str) -> dict:  1 usage
    file = clear_spaces(file)
    if file[0] == "{" and file[-1] == "}":
        file = parse_object(file)
    else: json_error(2)
    return file
```

```python
def parse_number(value: str) -> str:  2 usages
    if "e" in value.lower():
        num1, num2 = value[:value.lower().find("e")], value[value.lower().find("e") + 1:]
        for _ in num1:
            if _ not in "-0123456789.": json_error(4)
        if num1.count(".") > 1 \
                or num1.count("-") > 1 \
                or (num1.count("-") == 1 and num1[0] != "-"):
            json_error(3)
        for _ in num2:
            if _ not in "+-0123456789": json_error(5)
        if num2.count("+") > 1 \
                or num2.count("-") > 1:
            json_error(3)
    elif "." in value:
        for _ in value:
            if _ not in "-0123456789.": json_error(4)
        if value.count(".") > 1 \
                or value.count("-") > 1 \
                or (value.count("-") == 1 and value[0] != "-"):
            json_error(3)
    else:
        for _ in value:
            if _ not in "-0123456789": json_error(5)
        if value.count("-") > 1 \
                or (value.count("-") == 1 and value[0] != "-"):
            json_error(3)
    return value
```

```python
def get_pattern(value: dict | list) -> str:  3 usages
    result = str()
    if type(value) == dict:
        keys = list(value.keys())
        result += str(len(keys)) + "d"
        for i in keys:
            result += " " + str(len(i)) + " " + i
        values = list(value.values())
        for i in values:
            if type(i) == list or type(i) == dict:
                result += " " + get_pattern(i)
            elif type(i) == str:
                result += " " + i
            elif i in [True, False, None]:
                result += " " + str(i)
            else: json_error(7)
    elif type(value) == list:
        result += str(len(value)) + "l"
        for i in value:
            if type(i) == list or type(i) == dict:
                result += " " + get_pattern(i)
            elif type(i) == str:
                result += " " + i
            elif i in [True, False, None]:
                result += " " + str(i)
            else: json_error(7)
    return result
```

```python
def json_to_bin(json: str) -> str:  4 usages
    with open(json, "r", encoding="utf-8") as json_file:
        json_object = json_file.read()
    parsed_object = parse_json(json_object)
    parsed_object = get_pattern(parsed_object)
    bin_file = str()
    for i in parsed_object:
        character = bin(ord(i))[2:]
        while len(character) < 32:
            character = "0" + character
        bin_file += character
    return bin_file
```

```python
if __name__ == "__main__":
    try:
        bin_line = json_to_bin("schedule.json")
        with open("schedule.bin", "w") as result_file:
            result_file.write(bin_line)
    except Exception as error:
        print(f"Ошибка: {error}")
```

```python
def parse_array(array: str) -> list:  2 usages
    array = clear_spaces(array[1:-1])
    i = 0
    result = list()
    next_expected = "value"
    while i < len(array):
        if array[i] in "\r\t\n ": i += 1
        elif array[i] == "," and next_expected == "end":
            next_expected = "value"
            i += 1
        elif array[i] in "tfn":
            if next_expected == "value":
                if object[i:i+4] == "true":
                    result.append(True)
                    next_expected = "end"
                    i += 4
                elif object[i:i+4] == "null":
                    result.append(None)
                    next_expected = "end"
                    i += 4
                elif object[i:i+5] == "false":
                    result.append(False)
                    next_expected = "end"
                    i += 5
                else: json_error(4)
            else: json_error(2)
        elif array[i] == '"':
            j = i + 1
            while j < len(array):
                if array[j] == '"' and (j == 0 or array[j - 1] != '\\'): break
                elif j == len(array)-1: json_error(1)
                else: j += 1
            st = str(array[i+1:j])
            if next_expected == "value":
                result.append('"' + st + '"')
                next_expected = "end"
```

```python
            else:
                json_error(2)
            i = j + 1
        elif array[i] in "-0123456789":
            j = i
            while j < len(array) and array[j] in "+-.0123456789eE":
                j += 1
            if next_expected == "value":
                value = str(array[i:j])
                value = parse_number(value)
                result.append(value)
                next_expected = "end"
                i = j
            else:
                json_error(2)
        elif array[i] == "[":
            if next_expected == "value":
                value = list()
                j = i + 1
                count = 0
                while j < len(array):
                    if array[j] == "]":
                        if count == 0: break
                        else: count -= 1
                    elif array[j] == "[": count += 1
                    j += 1
                if j != len(array): value = parse_array(array[i:j+1])
                else: json_error(6)
                next_expected = "end"
                result.append(value)
                i = j + 1
            else: json_error(2)
        elif array[i] == "{":
            if next_expected == "value":
                value = list()
                j = i + 1
                count = 0
```

```python
                while j < len(array):
                    if array[j] == "}":
                        if count == 0: break
                        else: count -= 1
                    elif array[j] == "{": count += 1
                    j += 1
                if j != len(array): value = parse_object(array[i:j+1])
                else: json_error(5)
                next_expected = "end"
                result.append(value)
                i = j + 1
            else: json_error(2)
        else: json_error(2)
    if next_expected != "end": json_error(2)
    return result
```

```python
def parse_object(object: str) -> dict:  3 usages
    object = clear_spaces(object[1:-1])
    i = 0
    key = str()
    result = dict()
    next_expected = "key"
    while i < len(object):
        if object[i] in "\r\t\n ": i += 1
        elif object[i] == ":" and next_expected == "divider":
            next_expected = "value"
            i += 1
        elif object[i] == "," and next_expected == "end":
            next_expected = "key"
            i += 1
        elif object[i] in "tfn":
            if next_expected == "value":
                if object[i:i+4] == "true":
                    result.setdefault(key, True)
                    next_expected = "end"
                    i += 4
                elif object[i:i+4] == "null":
                    result.setdefault(key, None)
                    next_expected = "end"
                    i += 4
                elif object[i:i+5] == "false":
                    result.setdefault(key, False)
                    next_expected = "end"
                    i += 5
                else: json_error(4)
            else: json_error(2)
        elif object[i] == '"':
            j = i + 1
            while j < len(object):
                if object[j] == '"' and (j == 0 or object[j-1] != '\\'): break
                elif j == len(object) - 1: json_error(1)
                else: j += 1
            st = str(object[i+1:j])
            if next_expected == "key":
                next_expected = "divider"
                key = st
            elif next_expected == "value":
                value = '"' + st + '"'
                result.setdefault(key, value)
                next_expected = "end"
            else: json_error(2)
            i = j + 1
        elif object[i] in "-0123456789":
            j = i
            while j < len(object) and object[j] in "+-.0123456789eE":
                j += 1
            if next_expected == "value":
                value = str(object[i:j])
                value = parse_number(value)
                result.setdefault(key, value)
                next_expected = "end"
                i = j
            else: json_error(2)
        elif object[i] == "[":
            if next_expected == "value":
                value = list()
                j = i + 1
                count = 0
                while j < len(object):
                    if object[j] == "]":
                        if count == 0: break
                        else: count -= 1
                    elif object[j] == "[": count += 1
                    j += 1
                if j != len(object): value = parse_array(object[i:j+1])
                else: json_error(6)
                next_expected = "end"
                result.setdefault(key, value)
                i = j + 1
            else: json_error(2)
        elif object[i] == "{":
            if next_expected == "value":
                value = list()
                j = i + 1
                count = 0
                while j < len(object):
                    if object[j] == "}":
                        if count == 0: break
                        else: count -= 1
                    elif object[j] == "{": count += 1
                    j += 1
                if j != len(object): value = parse_object(object[i:j+1])
                else: json_error(5)
                next_expected = "end"
                result.setdefault(key, value)
                i = j + 1
            else: json_error(2)
        else: json_error(5)
    if next_expected != "end": json_error(2)
    return result
```

Дополнительное задание №1

Необходимо написать программу на языке Python, которая выполняет сериализацию полученного в основном задании бинарного объекта в файл формата yaml, определённый вариантом работы.

Исходный код программы:

```python
def parse_array(array: list) -> str:  2 usages
    values = str()
    for i in array:
        if type(i) == dict:
            some_value = parse_object(i).split("\n")
            value = str()
            for j in some_value:
                value += "  " + j + "\n"
            values += "-" + value[1:]
        elif type(i) == list:
            value = parse_array(i)
            values += "- " + value
        elif type(i) == str:
            if i[0] == '"':
                values += "- " + i[1:-1] + "\n"
            else:
                values += "- " + i + "\n"
        elif i in [True, False, None]:
            values += "- " + str(i) + "\n"
        else:
            print("Error in parse_object")
            exit()
    return values[:-1]
```

```python
def parse_object(object: dict) -> str:  3 usages
    result = str()
    keys = list(object.keys())
    values = list(object.values())
    for i in range(len(keys)):
        key, value = keys[i], values[i]
        result += key + ": "
        if type(value) == dict:
            some_value = parse_object(value).split("\n")
            value = str()
            for j in some_value:
                value += "    " + j + "\n"
            result += "\n" + value[:-1]
        elif type(value) == list:
            value = parse_array(value)
            result += "\n" + value
        elif type(value) == str:
            if value[0] == '"':
                result += value[1:-1]
            else:
                result += value
        elif value in [True, False, None]:
            result += str(value).lower()
        else:
            print("Error in parse_object")
            exit()
        result += "\n"
    return result[:-1]
```

```python
def bin_to_yaml(bin_line: str) -> str:  3 usages
    st = str()
    for i in range(len(bin_line) // 32):
        st += chr(int(bin_line[i * 32:(i + 1) * 32], 2))
    st = unparse_bin(st+" ")[0]
    result = "---\n"
    result += parse_object(st)
    return result
```

```python
if __name__ == "__main__":
    try:
        with open("schedule.bin", "r") as file:
            bin_file = file.read()
        result_file = bin_to_yaml(bin_file)
        with open("schedule.yml", "w", encoding="utf-8") as file:
            file.write(result_file)
    except Exception as error:
        print(f"Ошибка: {error}")
```

```python
def unparse_bin(line: str) -> list:  3 usages
    op, line = line[:line.index(" ")], line[line.index(" ")+1:]
    length, typer = int(op[:-1]), op[-1]
    if typer == "d":
        result = dict()
        keys = []
        values = []
        for i in range(length):
            count, line = int(line[:line.index(" ")]), line[line.index(" ")+1:]
            key, line = line[:count], line[count+1:]
            keys.append(key)
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = line[1:j]
                        line = line[j+2:]
                        break
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            values.append(value)
        for i in range(length):
            result.setdefault(keys[i], values[i])
    elif typer == "l":
        result = list()
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = line[1:j]
                        line = line[j+2:]
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            result.append(value)
    return [result, line]
```

Полученный в результате работы программы файл:

```
---
Tuesday:
- Lesson: Высшая алгебра (лек)
  Time: 11:30-13:00
  Weeks:
  - 1
  - 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 (P3115) 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 (P3115) 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 (P3115) 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
```

Дополнительно задание №2

Необходимо найти готовые библиотеки и написать с их использованием программу на языке Python, которая сможет десериализовать и сериализовать файл из исхдного в результирующий, определённый вариантом. В данном случае это библиотеки json и PyYAML. Файл в исходном формате был представлен в основном задании.

Исходный код программы:

```python
def get_pattern(value: dict | list) -> str:  3 usages
    result = str()
    if type(value) == dict:
        keys = list(value.keys())
        result += str(len(keys)) + "d"
        for i in keys:
            result += " " + str(len(i)) + " " + i
        values = list(value.values())
        for i in values:
            if type(i) == list or type(i) == dict:
                result += " " + get_pattern(i)
            elif type(i) in [str, float, int]:
                result += " " + '"' + str(i) + '"'
            elif i in [True, False, None]:
                result += " " + str(i)
            else:
                print("Error in get_pattern")
                exit()
    elif type(value) == list:
        result += str(len(value)) + "l"
        for i in value:
            if type(i) == list or type(i) == dict:
                result += " " + get_pattern(i)
            elif type(i) in [str, float, int]:
                result += " " + '"' + str(i) + '"'
            elif i in [True, False, None]:
                result += " " + str(i)
            else:
                print(type(i))
                print("Error in get_pattern")
                exit()
    return result
```

```python
def deserialize_json():  3 usages
    with open("schedule.json", "r", encoding="utf-8") as open_file:
        json_file = json.loads(open_file.read())
    json_file = get_pattern(json_file)
    bin_file = str()
    for i in json_file:
        character = bin(ord(i))[2:]
        while len(character) < 32:
            character = "0" + character
        bin_file += character
    with open("schedule(lib).bin", "w") as result_file:
        result_file.write(bin_file)
```

```python
def serialize_yaml():  3 usages
    with open("schedule(lib).bin", "r", encoding="utf-8") as open_file:
        bin_file = open_file.read()
    yaml_file = str()
    for i in range(len(bin_file) // 32):
        yaml_file += chr(int(bin_file[i * 32:(i + 1) * 32], 2))
    yaml_file = unparse_bin(yaml_file + " ")[0]
    yaml_file = yaml.safe_dump(yaml_file, allow_unicode=True)
    with open("schedule(lib).yml", "w", encoding="utf-8") as file:
        file.write(yaml_file)
```

```python
if __name__ == "__main__":
    try:
        deserialize_json()
        serialize_yaml()
    except Exception as error:
        print(f"Ошибка: {error}")
```

```python
def unparse_bin(line: str) -> list:  3 usages
    op, line = line[:line.index(" ")], line[line.index(" ")+1:]
    length, typer = int(op[:-1]), op[-1]
    if typer == "d":
        result = dict()
        keys = []
        values = []
        for i in range(length):
            count, line = int(line[:line.index(" ")]), line[line.index(" ")+1:]
            key, line = line[:count], line[count+1:]
            keys.append(key)
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = line[1:j]
                        line = line[j+2:]
                        break
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            values.append(value)
        for i in range(length):
            result.setdefault(keys[i], values[i])
    elif typer == "l":
        result = list()
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = line[1:j]
                        line = line[j+2:]
                        break
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            result.append(value)
    return [result, line]
```

Получаемый после выполнения программы файл:

```
Thursday:
- Group: ИНФОРМ (Р3115) 1.10
  Lesson: Информатика (лаб)
  Room: '1328'
  Time: 11:30-13:00
  Weeks:
  - '2'
  - '4'
  - '6'
  - '8'
  - '10'
  - '12'
  - '14'
  - '16'
- Group: ИНФОРМ (Р3115) 1.10
  Lesson: Информатика (лаб)
  Room: '1328'
```

```
  Time: 13:30-15:00
  Weeks:
  - '2'
  - '4'
  - '6'
  - '8'
  - '10'
  - '12'
  - '14'
  - '16'
- Group: ОПД (P3115) 1.10
  Lesson: Основы профессиональной деятельности (лаб)
  Room: 2435/3
  Time: 15:30-17:00
  Weeks:
  - '2'
  - '4'
  - '6'
  - '8'
  - '10'
  - '12'
  - '14'
  - '16'
- Group: ОПД (P3115) 1.10
  Lesson: Основы профессиональной деятельности (лаб)
  Room: 2435/3
  Time: 17:10-18:40
  Weeks:
  - '2'
  - '4'
  - '6'
  - '8'
  - '10'
  - '12'
  - '14'
  - '16'
- Group: ИНТ 1.3
  Lesson: История российской науки и техники (прак)
  Room: '2326'
  Time: 11:30-13:00
  Weeks:
  - '1'
  - '3'
  - '5'
  - '7'
  - '9'
  - '11'
  - '13'
  - '15'
- Group: ИНТ 1
  Lesson: История российской науки и техники (лек)
```

```
  Room: '1404'
  Time: 13:30-15:00
  Weeks:
  - '1'
  - '3'
  - '5'
  - '7'
  - '9'
  - '11'
  - '13'
  - '15'
Tuesday:
- Group: ВышАлг 11
  Lesson: Высшая алгебра (лек)
  Room: '1419'
  Time: 11:30-13:00
  Weeks:
  - '1'
  - '2'
  - '3'
  - '4'
  - '5'
  - '6'
  - '7'
  - '8'
  - '9'
  - '10'
  - '11'
  - '12'
  - '13'
  - '14'
  - '15'
  - '16'
  - '17'
- Group: ВышАлг 11.2
  Lesson: Высшая алгебра (прак)
  Room: '1327'
  Time: 13:30-15:00
  Weeks:
  - '2'
  - '3'
  - '4'
  - '5'
  - '6'
  - '7'
  - '8'
  - '10'
  - '11'
  - '12'
  - '13'
  - '14'
```

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

Как видно, размер программы, написанной с использованием библиотек, более чем в два раза короче, а также более читаем, чем код из основного задания. Также слегка отличается полученный этими программами файл: там, где использовались библиотеки, все элементы были расставленны по алфавиту (или по возрастанию).

Дополнительное задание №3

Необходимо переписать код из первого дополнительно задания так, чтобы после выполнения программы получался файл формата xml.

Исходный код программы:

```
def parse_array(array: list) -> str:  2 usages
    result = str()
    for i in range(len(array)):
        if i != 0: result += "\n"
        time_result = str()
        time_result += "<element>"
        value = array[i]
        if type(value) == dict:
            time_result += "\n"
            value = parse_object(value).split("\n")
            for i in range(len(value)):
                value[i] = "  " + value[i]
            value = "\n".join(value)
            time_result += value + "\n"
        elif type(value) == list:
            time_result += "\n"
            value = parse_array(value).split("\n")
            for i in range(len(value)):
                value[i] = "  " + value[i]
            value = "\n".join(value)
            time_result += value + "\n"
        elif type(value) == str:
            if value[0] == '"':
                time_result += value[1:-1]
            else:
                time_result += value
        elif value in [True, False, None]:
            time_result += str(value).lower()
        else:
            print("Error in parse_array")
            exit()
        time_result += "</element>"
        result += time_result
    return result
```

```
def parse_object(object: dict) -> str:  3 usages
    result = str()
    keys = list(object.keys())
    values = list(object.values())
    for i in range(len(keys)):
        if i != 0: result += "\n"
        time_result = str()
        key, value = keys[i], values[i]
        time_result += "<" + key + ">"
        if type(value) == dict:
            time_result += "\n"
            value = parse_object(value).split("\n")
            for i in range(len(value)):
                value[i] = "  " + value[i]
            value = "\n".join(value)
            time_result += value + "\n"
        elif type(value) == list:
            time_result += "\n"
            value = parse_array(value).split("\n")
            for i in range(len(value)):
                value[i] = "  " + value[i]
            value = "\n".join(value)
            time_result += value + "\n"
        elif type(value) == str:
            if value[0] == '"':
                time_result += value[1:-1]
            else:
                time_result += value
        elif value in [True, False, None]:
            time_result += str(value).lower()
        else:
            print("Error in parse_object")
            exit()
        time_result += "</" + key + ">"
        result += time_result
    return  result
```

```python
def bin_to_xml(bin_line: str) -> str:  3 usages
    result = str()
    st = str()
    for i in range(len(bin_line) // 32):
        st += chr(int(bin_line[i * 32:(i + 1) * 32], 2))
    st = unparse_bin(st  + " ")[0]
    result += parse_object(st)
    result = result.split("\n")
    for i in range(len(result)):
        result[i] = "  " + result[i]
    result = "<root>" + "\n" + "\n".join(result) + "\n" + "</root>"
    return result
```

```python
def unparse_bin(line: str) -> list:  3 usages
    op, line = line[:line.index(" ")], line[line.index(" ")+1:]
    length, typer = int(op[:-1]), op[-1]
    if typer == "d":
        result = dict()
        keys = []
        values = []
        for i in range(length):
            count, line = int(line[:line.index(" ")]), line[line.index(" ")+1:]
            key, line = line[:count], line[count+1:]
            keys.append(key)
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = '"' + line[1:j] + '"'
                        line = line[j+2:]
                        break
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            values.append(value)
        for i in range(length):
            result.setdefault(keys[i], values[i])
    elif typer == "l":
        result = list()
        for i in range(length):
            nexter = line[:line.index(" ")]
            if line[0] == '"':
```

```python
                for j in range(len(line)):
                    if line[j] == '"' and line[j-1] != "\\" and j != 0:
                        value = '"' + line[1:j] + '"'
                        line = line[j+2:]
            elif nexter[-1] in "dl":
                value, line = unparse_bin(line)
            elif all([x in "+-.0123456789eE" for x in nexter]):
                value = nexter
                line = line[line.index(" ")+1:]
            elif nexter in ["True", "False", "None"]:
                value = bool(nexter)
                line = line[line.index(" ") + 1:]
            else:
                print("Error in unparse_bin")
                exit()
            result.append(value)
    return [result, line]
```

```python
if __name__ == "__main__":
    try:
        with open("schedule.bin", "r") as file:
            bin_file = file.read()
        result_file = bin_to_xml(bin_file)
        with open("schedule.xml", "w", encoding="utf-8") as file:
            file.write(result_file)
    except Exception as error:
        print(f"Ошибка: {error}")
```

Полученный в результате выполнения программы файл:

```
<root>
  <Tuesday>
    <element>
      <Lesson>Высшая алгебра (лек)</Lesson>
```

```
  <Time>11:30-13:00</Time>
  <Weeks>
    <element>1</element>
    <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>
```

Дополнительно задание №4

Необходимо сравнить время стократного выполнения конвертации и парсинга программ, полученном в основном и дополнительном заданиях. Для этого напишем программу на языке Python, которая вызывает каждый из написанных ранее файлов в стократном цикле и считает время ее выполнения.

Исходный код программы:

```python
from time import time
from task1 import json_to_bin
from task2 import bin_to_yaml
from task3 import deserialize_json, serialize_yaml
from task4 import bin_to_xml


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

    timer = time()
    for i in range(100):
        deserialize_json()
        serialize_yaml()
    print("json -> yaml (with libraries)")
    print(time() - timer)

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

В результате выполнения данной программы мы получаем три значения времени, затраченного на выполнения программ. При каждом выполнении результаты слегка разнятся, но вот один из полученных выводов программы:

```
json -> yaml
0.4172251224517822
json -> yaml (with libraries)
3.7135400772094727
json -> xml
0.49333882331848145
```

На основании этих данных можем сделать ряд выводов. Во-первых, конвертация и парсинг файлов из формата json в формат yaml выполняется намного быстрее без использования библиотек. Во-вторых, хотя время перевода файла формата json в форматы yaml и xml почти одинакого, все же конвертация в xml выполняется немного дольше.

Заключение

Во время выполнения данной лабораторной работы были изучены основы работы с разными форматами файлов, их сериализация и десериализация, перевод файлов из одного формата в другой, а также закрепление всего этого путём написания программ для решения поставленных задач.