|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Технологии индустриального программирования |
| ИНСТИТУТ | ИПТИП |
| КАФЕДРА | Индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Методические указания к практическим занятиям по дисциплине |
| ПРЕПОДАВАТЕЛЬ | Александров Алексей Сергеевич |
| СЕМЕСТР | 3 семестр, 2024/2025 уч. год |

# Практическое занятие №9

## Форматирование данных

JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript, но при этом формат независим от JS и может использоваться в любом языке программирования.

JSON-текст представляет собой набор пар ключ: значение.

В качестве значений в JSON могут быть использованы:

* Запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки {}. Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* Массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки []. Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
* Число (целое или вещественное).
* Литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
* Строка – текст любой длины.

Пример JSON:

{

"name": "Иван",

"group": "ЭФБО-01-22",

"exams": {

"русский язык": 90,

"математика": 95,

"информатика": 93

},

"phoneNumbers": [

"88121231234",

"89161234567"

]

}

В Qt для работы с JSON используются следующие классы:

|  |  |
| --- | --- |
| **Класс** | **Действие** |
| [QJsonDocument](https://doc.qt.io/qt-6/qjsondocument.html) | Инкапсулирует чтение и запись документов JSON |
| [QJsonObject](https://doc.qt.io/qt-6/qjsonobject.html) | Инкапсулирует объект JSON |
| [QJsonArray](https://doc.qt.io/qt-6/qjsonarray.html) | Инкапсулирует массив JSON |
| [QJsonValue](https://doc.qt.io/qt-6/qjsonvalue.html) | Инкапсулирует значение в JSON |

Для создания объекта JSON необходимо использовать класс QJsonObject:

QJsonObject jsonObject;

Наполнение объекта значениями производится с помощью оператора []:

jsonObject["name"] = "Иван";

jsonObject["group"] = "ЭФБО-01-22";

Вложенный объект JSON создаётся таким же образом:

QJsonObject examsObject;

examsObject["русский язык"] = 90;

examsObject["математика"] = 95;

examsObject["информатика"] = 93;

После чего с помощью оператора [] устанавливается в основной объект:

jsonObject["exams"] = examsObject;

Создание массива производится с помощью класса QJsonArray, данные добавляются с помощью метода append(), как в обычном контейнере:

QJsonArray phoneNumbersArray;

phoneNumbersArray.append("88121231234");

phoneNumbersArray.append("89161234567");

jsonObject["phoneNumbers"] = phoneNumbersArray;

После добавления всех необходимых данных, из объекта JSON создаётся документ, с помощью класса QJsonDocument:

QJsonDocument jsonDoc(jsonObject);

Данный документ можно преобразовать в строку, с помощью метода toJson() и записать в файл:

// записываем JSON в файл

QFile file("data.json");

if (!file.open(QIODevice::WriteOnly))

{

QMessageBox::warning(this, "Ошибка", "Не удалось открыть файл!");

return;

}

file.write(jsonDoc.toJson());

file.close();

QMessageBox::information(this, "Успех", "Файл успешно записан!");

Полный код записи JSON файла выглядит следующим образом:



Код записи JSON файла.

Чтение записанного файла в формате JSON производится аналогичным образом.

Сначала необходимо открыть файл JSON и прочитать весь текст:

// открываем файл

QFile file("data.json");

if (!file.open(QIODevice::ReadOnly | QIODevice::Text))

{

QMessageBox::warning(this, "Ошибка", "Файл не открыт!");

return;

}

QString jsonString = file.readAll(); // достаём JSON

file.close();

Сейчас весь текст JSON находится в переменной jsonString. Используя статический метод QJsonDocument::fromJson() класса QJsonDocument необходимо разобрать документ на элементы:

QJsonDocument jsonDoc = QJsonDocument::fromJson(jsonString.toUtf8());

if (jsonDoc.isEmpty())

{

QMessageBox::critical(this, "Ошибка", "Файл JSON некорректный!");

return;

}

Далее, из документа необходимо извлечь главный (корневой) объект:

QJsonObject jsonObject = jsonDoc.object();

Из которого можно достать значения:

QString name = jsonObject["name"].toString();

QString group = jsonObject["group"].toString();

Аналогичным образом извлекаются вложенные объекты:

QJsonObject examsObject = jsonObject["exams"].toObject();

int russianExam = examsObject["русский язык"].toInt();

int mathExam = examsObject["математика"].toInt();

int informaticsExam = examsObject["информатика"].toInt();

Для извлечения массива необходимо выбрать значение, содержащее массив и использовать метод toArray():

QJsonArray phoneNumbersArray = jsonObject["phoneNumbers"].toArray();

QStringList phoneNumbers;

for (const QJsonValue& value : phoneNumbersArray)

{

phoneNumbers.append(value.toString());

}

Полный код чтения JSON файла выглядит следующим образом:



Код чтения файла JSON.

Таким образом после извлечения всех данных, значения переменных name, group, russianExam, mathExam, informaticsExam и списка phoneNumbers можно использовать для дальнейшей их обработки в программе.