Практическая работа № 4.  
Работа с json файлами

**Цель практической работы**

Закрепление теоретических знаний по работе с данными в формате JSON.

**Постановка задачи**

**Сериализация** — это процесс **преобразования объекта** в поток байтов (в нашем случае **в строку**) для сохранения или передачи в память, базу данных или файл. Эта операция предназначена для того, чтобы **сохранить состояния объекта** для последующего воссоздания при необходимости. Обратный процесс называется **десериализацией**.

Например, мы хотим создать объект типа Employee и сохранить его в файл, чтобы потом можно было загрузить с заполненными данными.

public class Employee

{

public string FirstName { get; set; } = "";

public string LastName { get; set; } = "";

public double Salary { get; set; }

public Employee(

string firstName,

string lastName,

double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

Для начала нужно скачать с помощью NuGet пакетов библиотеку **Newtonsoft.JSON** (рисунок 1).

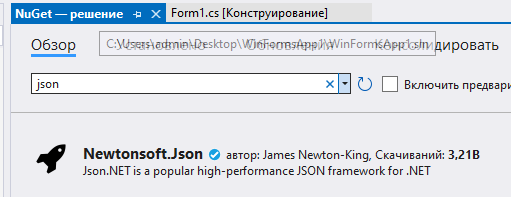


Рисунок 1 – Установка библиотеки Newtonsoft.Json

Создаем объект сотрудника, и сериализуем в строку формата JSON, далее сохраняем в файл стандартным способом:

var employee = new Employee("John", "Black", 3000);

var jsonString = JsonConvert.SerializeObject(employee);

File.WriteAllText("file.json", jsonString);

В данном случае **сериализуются все публичные поля и свойства!**

Полученный файл представлен на рисунке 2. Данный файл можно открыть любым текстовым редактором, также удобно использовать **VisualStudio** или **Notepad++**.

Данные по сотруднику также можно было сохранить в виде текста вручную: сформировать самим строку определенного формата и сохранить ее, однако тогда нужно было бы продумать обратную операцию получения объекта из строки. Очевидно, что гораздо удобнее пользоваться готовым решением в виде JSON.

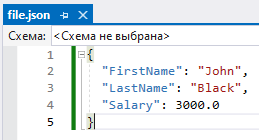


Рисунок 2 – Содержимое JSON файла

**Для десериализации** (получения объекта из JSON строки) нужно написать:

var employee1 = JsonConvert.DeserializeObject<Employee>(jsonString);

Далее рассмотрим, **что делать, если нужно сериализовать приватное поле или свойство, или наоборот если что-то сериализовать не нужно.**

Атрибуты в .NET представляют специальные инструменты, которые позволяют встраивать в сборку дополнительные **метаданные**.

**Атрибуты указываются в квадратных скобках**, над классом, полем или свойством.

Чтобы явно указать, **что объект можно сериализовать** используется атрибут [Serializable] (для рассматриваемой библиотеки он не обязателен).

Если мы хотим задавать свои имена у данных **в файле JSON** нужно использовать.

[DataContract] перед именем класса, а сами поля и свойства пометить [DataMember].

Чтобы отметить нужный конструктор для десериализации, нужно использовать атрибут [JsonConstructor], при этом имена переменных в конструкторе должны совпадать с именами соответствующих полей и свойств (без учета регистра).

[DataContract]

public class Employee

{

[DataMember(Name = "Имя")]

public string FirstName { get; set; } = "";

[DataMember(Name = "Фамилия")]

public string LastName { get; set; } = "";

[DataMember(Name = "Зарплата")]

public double Salary { get; set; }

[JsonConstructor]

public Employee(string firstName, string lastName, double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

На рисунке 3 представлен результат сериализации объекта с заданными именами полей.

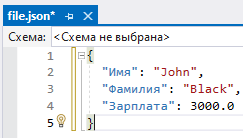


Рисунок 3 – Содержимое JSON файла  
с заданными именами полей

Чтобы указать, что поле не нужно сериализовать необходимо отметить его атрибутом [JsonIgnore], результат представлен на рисунке 4.

[DataContract]

public class Employee

{

[JsonIgnore]

public string FirstName { get; set; } = "";

[DataMember(Name = "Фамилия")]

public string LastName { get; set; } = "";

[DataMember(Name = "Зарплата")]

public double Salary { get; set; }

[JsonConstructor]

public Employee(string firstName, string lastName, double salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

}

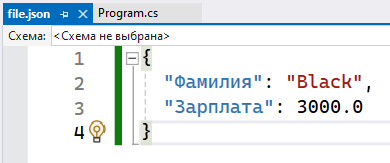


Рисунок 4 – Содержимое JSON файла  
с исключением поля «FirstName»

Также можно сериализовать вложенные классы. Пример сериализации объекта класса, содержащего внутри себя другие классы:

[DataContract]

public class Company

{

[DataMember(Name = "Директор")]

public Employee Director { get; set; }

[DataMember(Name = "Сотрудники")]

public List<Employee> Employees { get; set; } = new();

[JsonConstructor]

public Company(Employee director, List<Employee> employees)

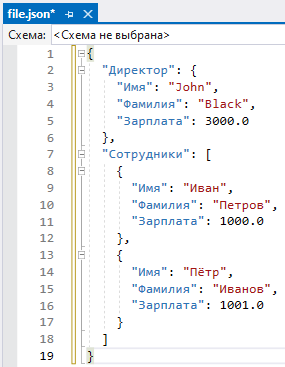
{

Director = director;

Employees = employees;

}

}



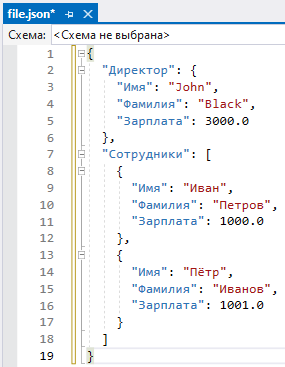


Рисунок 5 – Содержимое JSON файла с вложенными классами

Если объекты имеют сложную и запутанную структуру, являются наследниками класса, требуют вызова каких-либо метода для инициализации, то автоматическая сериализация будет затруднена или невозможна. Рассмотрим пример «ручной» сериализации.

var employee = new Employee("John", "Black", 3000);

JObject j = new JObject();

j["Имя"] = employee.FirstName;

j["Фамилия"] = employee.LastName;

j["Зарплата"] = employee.Salary;

var status = employee.Salary switch

{

< 100 => "Бедняк",

< 1000 => "Средний класс",

< 5000 => "Богач",

\_ => "Сказочный богач"

};

j["Статус"] = status;

File.WriteAllText("file.json", j.ToString());

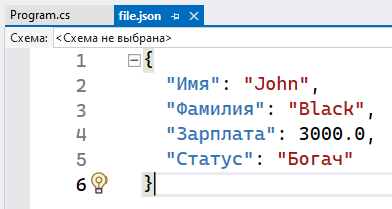


Рисунок 6 – Содержимое JSON файла после ручного заполнения

Видно, что работа с JSON напоминает работу со словарем. напишем код для чтения данного файла.

var j\_str = File.ReadAllText("file.json");

var json = JObject.Parse(j\_str);

var employee1 = new Employee(

json["Имя"].Value<string>(),

json["Фамилия"].Value<string>(),

json["Зарплата"].Value<int>());

**Задание на практическую работу**

1. Создать Windows Forms приложение, реализовать графический интерфейс для создания, сохранения и загрузки **объекта** согласно варианту.
2. При загрузке объекта его параметры записываются в интерфейс формы.
3. Для сохранения и загрузки использовать **JSON**.
4. Защита работы включает демонстрацию работы программы на подготовленном примере.

**Варианты заданий на практическую работу**

**1. Продуктовый магазин**

* Name – Название магазина.
* Address – Адрес.
* NumberOfEmployees – Количество сотрудников.
* AvailableProducts – Список товаров.
* AverageRating – Средний рейтинг (1-5).

**2. Банковский счет**

* AccountNumber – Номер счета.
* OwnerName – Владелец.
* Balance – Баланс.
* Currency – Валюта (RUB, USD и т. д.).
* IsBlocked – Блокировка.

**3. Студент**

* FullName – ФИО.
* StudentId – Номер студ. билета.
* Group – Группа.
* AverageGrade – Средний балл.
* IsOnScholarship – Стипендия.

**4. Завод**

* Name – Название.
* Industry – Отрасль.
* EmployeeCount – Число работников.
* AnnualRevenue – Годовой доход.
* IsOperational – Работает/не работает.

**5. Смартфон**

* Brand – Бренд.
* Model – Модель.
* StorageGB – Память (ГБ).
* ScreenSizeInches – Диагональ экрана.
* Has5G – Поддержка 5G.

**6. Компьютер**

* CPU – Процессор.
* RAMGB – ОЗУ (ГБ).
* GPU – Видеокарта.
* StorageSSDGB – SSD (ГБ).
* OperatingSystem – ОС.

**7. Продукт (в магазине)**

* Name – Название.
* Price – Цена.
* ExpirationDate – Срок годности.
* Category – Категория.
* InStock – Количество в наличии.

**8. Грузовой автомобиль**

* Brand – Марка.
* LoadCapacityKg – Грузоподъемность (кг).
* Mileage – Пробег (км).
* IsRefrigerated – Рефрижератор.
* LicensePlate – Номер.

**9. Книжный шкаф**

* Material – Материал.
* NumberOfShelves – Количество полок.
* HeightCm – Высота (см).
* WidthCm – Ширина (см).
* IsOpen – Открытый/закрытый.

**10. Портфель**

* Brand – Бренд.
* Color – Цвет.
* NumberOfCompartments – Отделения.
* WeightKg – Вес (кг).
* IsWaterproof – Водонепроницаемость.

**11. Банк**

* Name – Название.
* Headquarters – Штаб-квартира.
* NumberOfBranches – Филиалы.
* TotalAssets – Активы.
* IsInternational – Международный.

**12. Жилой дом**

* Address – Адрес.
* Floors – Этажность.
* ApartmentsCount – Квартиры.
* YearBuilt – Год постройки.
* HasElevator – Наличие лифта.

**13. Ракета**

* Model – Модель.
* MaxSpeedKmH – Макс. скорость (км/ч).
* PayloadCapacityKg – Грузоподъемность (кг).
* FuelType – Тип топлива.
* IsReusable – Многоразовая.

**14. Офицер**

* FullName – ФИО.
* Rank – Звание.
* ServiceNumber – Номер жетона.
* YearsOfService – Стаж.
* IsOnDuty – На службе.

**15. Порт (для кораблей)**

* Name – Название.
* Location – Местоположение.
* DocksCount – Количество доков.
* MaxShipSize – Макс. размер судна.
* IsMilitary – Военный/гражданский.

**16. Животное**

* Species – Вид животного
* Age – Возраст (лет)
* WeightKg – Вес (кг)
* IsDomestic – Домашнее/дикое
* Habitat – Среда обитания

**17. Страна**

* Name – Название страны
* Capital – Столица
* Population – Численность населения
* OfficialLanguage – Официальный язык
* AreaSqKm – Площадь (км²)

**18. Самолет**

* Model – Модель
* MaxSpeedKmH – Макс. скорость (км/ч)
* PassengerCapacity – Вместимость пассажиров
* FuelType – Тип топлива
* IsCommercial – Коммерческий/военный

**19. Материал**

* Name – Название материала
* Density – Плотность (г/см³)
* MeltingPoint – Температура плавления (°C)
* IsConductive – Электропроводность
* Color – Цвет

**20. Планета**

* Name – Название
* DiameterKm – Диаметр (км)
* DistanceFromSun – Расстояние от Солнца (млн км)
* HasAtmosphere – Наличие атмосферы
* NumberOfMoons – Количество спутников

**21. Адресная книга**

* OwnerName – Владелец
* NumberOfContacts – Количество контактов
* IsDigital – Электронная/бумажная
* LastUpdated – Дата обновления
* IsPrivate – Приватность

**22. Аквариум**

* VolumeLiters – Объем (литры)
* FishCount – Количество рыб
* HasFilter – Наличие фильтра
* WaterType – Тип воды (пресная/морская)
* IsLighted – Подсветка

**23. Департамент**

* Name – Название
* HeadName – Руководитель
* EmployeeCount – Число сотрудников
* Budget – Годовой бюджет
* IsActive – Активен/неактивен

**24. Игра**

* Title – Название
* Genre – Жанр
* ReleaseYear – Год выпуска
* IsMultiplayer – Многопользовательская
* Rating – Рейтинг (1-10)

**25. Артиллерийское орудие**

* Model – Модель
* CaliberMm – Калибр (мм)
* RangeKm – Дальность стрельбы (км)
* YearProduced – Год выпуска
* IsMobile – Мобильное/стационарное

**26. Сундук**

* Material – Материал
* LockType – Тип замка
* VolumeLiters – Объем (литры)
* IsAntique – Антикварный
* SecretCompartment – Секретное отделение

**27. Армия**

* Country – Страна
* ActivePersonnel – Численность
* Budget – Годовой бюджет
* MainBase – Главная база
* IsNuclearCapable – Ядерный потенциал

**28. Огород**

* AreaSqM – Площадь (м²)
* PlantTypes – Виды растений
* HasIrrigation – Поливная система
* FenceType – Тип ограждения
* IsOrganic – Органическое земледелие

**29. Танк**

* Model – Модель
* ArmorThicknessMm – Толщина брони (мм)
* MainGunCaliber – Калибр орудия (мм)
* CrewSize – Экипаж
* IsAmphibious – Плавучесть

**30. Корабль**

* Name – Название
* Type – Тип (пассажирский/грузовой)
* DisplacementTons – Водоизмещение (тонн)
* MaxSpeedKnots – Макс. скорость (узлов)
* IsMilitary – Военный/гражданский

**Пример разработки приложения**

**Корабль**

* Name – Название
* Type – Тип (пассажирский/грузовой)
* DisplacementTons – Водоизмещение (тонн)
* MaxSpeedKnots – Макс. скорость (узлов)
* IsMilitary – Военный/гражданский

**Шаг 1: Создание проекта Windows Forms**

1. Откройте Visual Studio.
2. Создайте новый проект: **Windows Forms App (.NET Framework)**.
3. Назовите проект, например, ShipApp.

**Шаг 2: Добавление класса Ship**

1. Правой кнопкой по проекту → **Add** → **Class**.
2. Назовите класс Ship.cs.
3. Реализуйте класс:

using System.Runtime.Serialization;

namespace WinFormsApp1;

public class Ship

{

public Ship(

string name,

string type,

double displacementTons,

double maxSpeedKnots,

bool isMilitary)

{

Name = name;

Type = type;

DisplacementTons = displacementTons;

MaxSpeedKnots = maxSpeedKnots;

IsMilitary = isMilitary;

}

[DataMember(Name = "Название")]

public string Name { get; set; }

[DataMember(Name = "Тип")]

public string Type { get; set; } // "Пассажирский" или "Грузовой"

[DataMember(Name = "Водоизмещение, тонн")]

public double DisplacementTons { get; set; }

[DataMember(Name = "Макс. скорость, узлов")]

public double MaxSpeedKnots { get; set; }

[DataMember(Name = "Военный?")]

public bool IsMilitary { get; set; }

}

**Шаг 3: Добавление Newtonsoft.Json для работы с JSON**

1. Правой кнопкой по проекту → **Manage NuGet Packages**.
2. Найдите и установите Newtonsoft.Json.

**Шаг 4: Дизайн формы (Form1)**

1. Откройте Form1 в дизайнере.
2. Добавьте элементы управления:
   * **TextBox** для Name (название).
   * **ComboBox** для Type (тип корабля).
   * **NumericUpDown** для DisplacementTons и MaxSpeedKnots.
   * **CheckBox** для IsMilitary (военный/гражданский).
   * **Button** для сохранения (btnSave).
   * **Button** для загрузки (btnLoad).
   * **Label** для вывода информации (lblInfo).

Примерный дизайн представлен на рисунке 7.

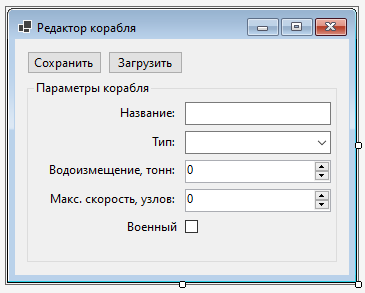


Рисунок 7 – Дизайн формы

**Шаг 5: Код формы (Form1.cs)**

using Newtonsoft.Json;

namespace WinFormsApp1;

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

// Заполняем ComboBox типами кораблей

comboBoxType.Items.AddRange(new[] { "Пассажирский", "Грузовой" });

comboBoxType.SelectedIndex = 0;

}

private void btnSave\_Click(object sender, EventArgs e)

{

var ship = new Ship(

textBoxName.Text,

comboBoxType.SelectedItem.ToString(),

(double)numericUpDownDisplacement.Value,

(double)numericUpDownMaxSpeed.Value,

checkBoxIsMilitary.Checked);

string json = JsonConvert.SerializeObject(ship, Formatting.Indented);

var sfd = new SaveFileDialog

{

DefaultExt = ".json",

Filter = "JSON | \*.json"

};

if (sfd.ShowDialog() != DialogResult.OK)

return;

File.WriteAllText(sfd.FileName, json);

MessageBox.Show($"Данные сохранены в {sfd.FileName}");

}

private void btnLoad\_Click(object sender, EventArgs e)

{

var ofd = new OpenFileDialog

{

DefaultExt = ".json",

Filter = "JSON | \*.json"

};

if (ofd.ShowDialog() != DialogResult.OK)

return;

if (!File.Exists(ofd.FileName))

MessageBox.Show($"Файл {ofd.FileName} не найден!");

string json = File.ReadAllText(ofd.FileName);

Ship ship = JsonConvert.DeserializeObject<Ship>(json);

textBoxName.Text = ship.Name;

comboBoxType.SelectedItem = ship.Type;

numericUpDownDisplacement.Value = (decimal)ship.DisplacementTons;

numericUpDownMaxSpeed.Value = (decimal)ship.MaxSpeedKnots;

checkBoxIsMilitary.Checked = ship.IsMilitary;

}

}

**Шаг 6: Инициализация компонентов (Form1.Designer.cs)**

Убедитесь, что в Form1.Designer.cs корректно объявлены все элементы управления (они создаются автоматически при перетаскивании из Toolbox).

**Шаг 7: Тестирование приложения**

1. Запустите приложение (**F5**).
2. Введите данные:
   * Название: "Титаник".
   * Тип: "Пассажирский".
   * Водоизмещение: 52310.
   * Макс. скорость: 23.
   * Военный: Нет.
3. Нажмите **Сохранить** → выберите куда сохранить файл.
4. Нажмите **Загрузить** → выберите откуда загрузить файл, данные появятся в форме.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»