Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

Кошовий В. В., Трунов О. М.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «ДІАГНОСТИКА СПОНДИЛОАРТРОЗУ ПОПЕРЕКОВОГО ВІДДІЛУ ХРЕБТА»

Навчальний посібник
для підготовки бакалаврів у галузі знань
12 «Інформаційні технології» та магістрів за спеціальністю
174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Миколаїв – 2023

УДК 004.42:616.711
К76

Рекомендовано до друку вченою радою Чорноморського національного університету імені Петра Могили (протокол № 10 від 22 грудня 2022 р .)

Рецензенти:

Дихта Л. М. – д-р техн. наук, професор, професор кафедри автоматизації та комп'ютерно-інтегрованих технологій Чорноморського національного університету імені Петра Могили.

Лисенков Е. А. – д-р фіз.-мат. наук, доцент, доцент кафедри інтелектуальних інформаційних систем Чорноморського національного університету імені Петра Могили.

Іванна Дронюк – канд. фіз.-мат. наук, ад'юнкт-професор факультету природничих та технічних наук Університету ім. Яна Длугоша в Ченсто-хові.

К76 **Кошовий В. В.** Розробка програмного забезпечення «Діагностика спондилоартрозу поперекового відділу хребта» : навч. посіб. для підготовки бакалаврів у галузі знань 12 «Інформаційні технології» та магістрів за спец. 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» / В. В. Кошовий, О. М. Трунов. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 36 с.

ISBN 978-966-336-445-2

Навчальний посібник присвячено проблемам розробки програмного забезпечення для діагностики спондилоартрозу поперекового відділу хребта.

В посібнику розглянуто використання мови C# для створення програмного забезпечення, за допомогою якого спрощується діагностування спондилоартрозу поперекового відділу хребта (далі діагностика СПВХ). Посібник реалізує ідеї теми «Розроблення мобільних малогабаритних та стаціонарних бездротових приладів ранньої діагностики, профілактики, лікування та посттравматичних відновлень військово-цивільного застосування», номер державної реєстрації: 0119U100422, та відкриває можливості для командної роботи науковців, аспірантів, студентів факультету комп'ютерних наук та медичного інституту з креативного розвитку.

УДК 004.42:616.711

ISBN 978-966-336-445-2

ЗМІСТ

ВСТУП

Сьогодні мова програмування C# – одна з найпотужніших мов в IT-галузі, що швидко розвиваються. На даний момент на C# пишуться найрізноманітніші програми: від невеликих десктопних програм до великих інформаційно-довідкових систем, які обслуговують щодня мільйони користувачів.

Коли говорять C#, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). Мова C# була створена спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Фреймворк .NET представляє собою потужну платформу для створення додатків. Можна виділити наступні її основні риси:

- **Підтримка декількох мов**. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов. При компіляції код на будь-якій з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) – свого роду асемблер платформи .NET.
- **Кросплатформеність**. .NET підтримується на більшості сучасних ОС: Windows, MacOS, Linux тощо.
- **Потужна бібліотека класів**. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів.
- **Різноманітність технологій**. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF і UWP, для створення більш простих графічних додатків – Windows Forms.

В посібнику розглянуто використання мови C# для створення програмного забезпечення, за допомогою якого спрощується діагностування спондилоартрозу поперекового відділу хребта (далі діагностика СПВХ). Посібник реалізує ідеї теми «Розроблення мобільних малогабаритних та стаціонарних бездротових приладів ранньої діагностики, профілактики, лікування та посттравматичних відновлень військово-цивільного застосування», номер державної реєстрації: 0119U100422, та відкриває можливості для командної роботи науковців, аспірантів, студентів факультету комп'ютерних наук та медичного інституту з креативного розвитку.

1. ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «ДІАГНОСТИКА СПОНДИЛОАРТРОЗУ ПОПЕРЕКОВОГО ВІДДІЛУ ХРЕБТА»

1.1 Структура програмного забезпечення

Програмне забезпечення для діагностики спондилоартрозу поперекового відділу хребта повинно складається з наступних підсистем:

- Підсистема додавання даних до сховища даних. Спеціально розроблений інтерфейс дозволяє додавати до сховища дані, що використовуються для діагностування СПВХ.
- Підсистема завантаження даних зі сховища даних для діагностики СПВХ. Актуальні дані завантажуються зі сховища даних для подальшого використання в діагностиці СПВХ.
- Підсистема проведення діагностики та визначення діагнозу щодо СПВХ. Призначена для проведення діагностики СПВХ за допомогою даних, завантажених зі сховища даних за визначеним алгоритмом, та визначення діагнозу на основі результатів обстеження.
- Підсистема довідкової інформації.
- Сховище даних. В якості сховища даних може виступати файл, база даних тощо.

Структуру програмного забезпечення та зв’язки між підсистемами зображена на рис. 1.1.

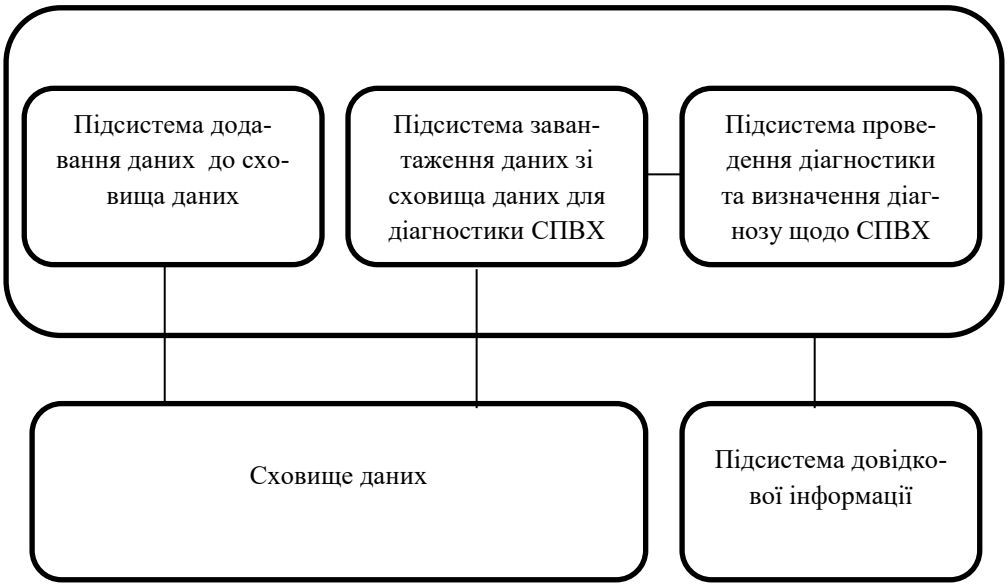

Рис. 1.1. Структура програмного забезпечення «Діагностика спондилоартрозу поперекового відділу хребта»

Розділення на підсистеми зумовлено виділенням основних функцій, що виконує програмне забезпечення. Це також дозволяє вносити модифікації в окремі модулі, не змінюючи інший код.

1.2 Інтерфейс програмного забезпечення

При завантаженні програми відкривається головна форма з рядком меню.

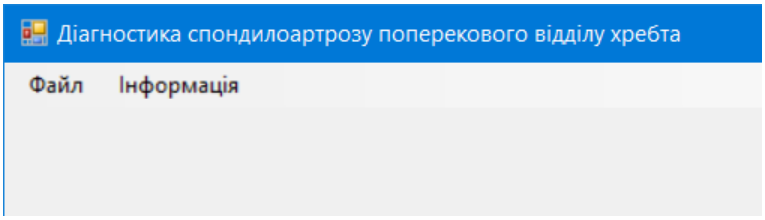

Рис. 1.2. Головне меню програми

Меню Файл містить команди для завантаження даних у сховище даних, зчитування даних зі сховища даних, проведення тестування СПВХ та виходу із системи.

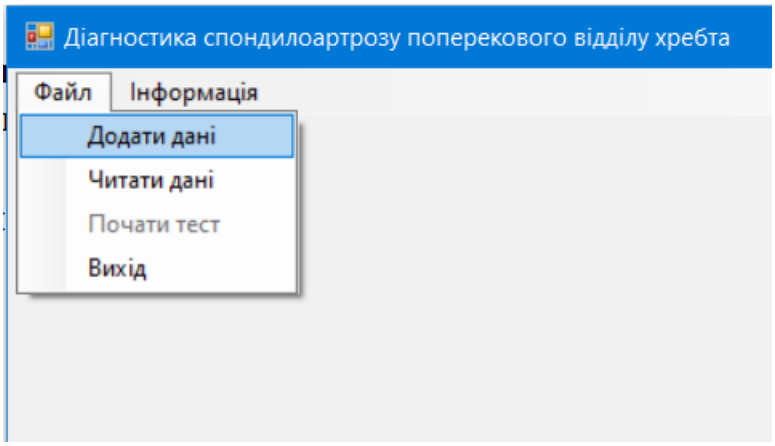

Рис. 1.3. Меню «Файл»

Для додавання даних до сховища даних використовується підсистема додавання даних. Читання даних зі сховища виконує підсистема завантаження даних зі сховища даних для діагностики СПВХ. Поки дані не завантажені, команда «Почати тест» неактивна. Після читання даних зі сховища даних відбувається активація команди «Почати тест».

Зауваження 1. Якщо після завантаження даних командою «Читати дані», виконати команду «Додати дані», тобто оновити інформацію в сховищі даних, команда «Почати тест» знову стає неактивна. Тому потрібно знову завантажити вже оновлені дані зі сховища даних.

Для виходу з системи використовується команда «Вихід».

Меню «Інформація» містить інформацію про призначення та авторів програми.

2. ФУНКЦІЇ ПІДСИСТЕМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Підсистема додавання даних до сховища даних

При виконанні команди «Файл»–>«Додати дані» відкривається модальне вікно для додавання додаткової інформації до сховища даних.

В полі «Нове питання» потрібно ввести текст нового запитання. В полях «Варіант відповіді» вводяться можливі варіанти відповіді на задане питання. В полях «Бали за відповідь» вказується кількість балів, що нараховується при виборі відповідного варіанту відповіді. При некоректному введенні даних з'являється повідомлення «Помилка введення даних».

Після введення інформації потрібно натиснути кнопку «Додати питання». Питання додається до сховища даних, а поля форми очищуються для введення нової інформації.

Для закінчення роботи – натиснути кнопку «Закрити форму».

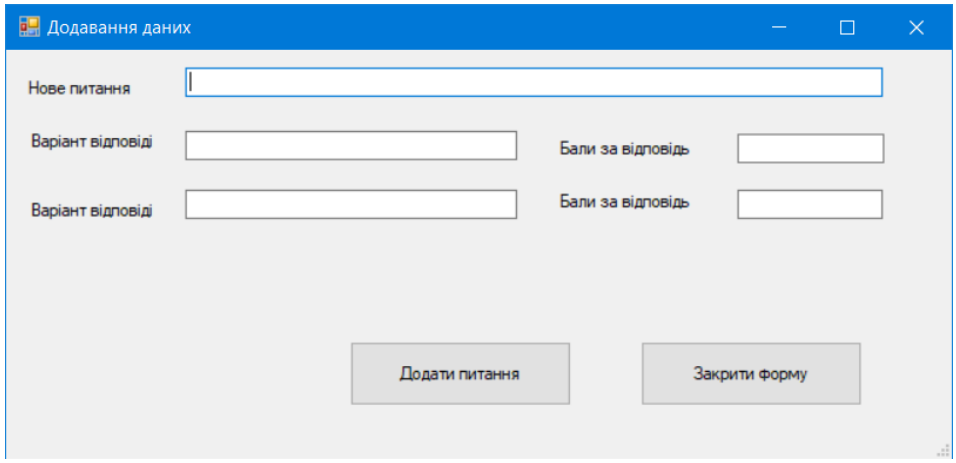

Рис. 2.1. Додавання даних до сховища даних

2.2 Підсистема завантаження даних зі сховища даних

При виконанні команди «Файл»–>«Читати дані» здійснюється перевірка на існування та можливість читання інформації зі сховища даних. При позитивному результаті на форму виводиться повідомлення про кількість питань для тестування та активується команда «Файл»–>«Почати тест».

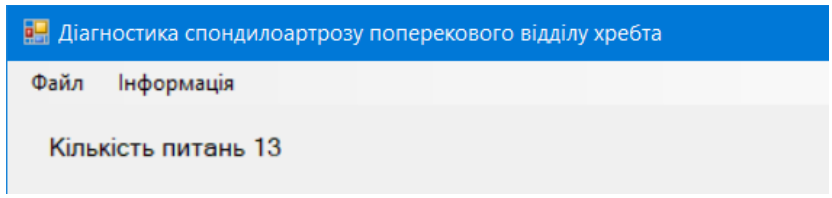

Рис. 2.2. Зчитування даних зі сховища даних

При негативному результаті перевірки на існування доступу до сховища даних, виводиться вікно повідомлення про помилку і неможливість зчитати дані. Команда «Файл»–>«Почати тест» залишається неактивною.

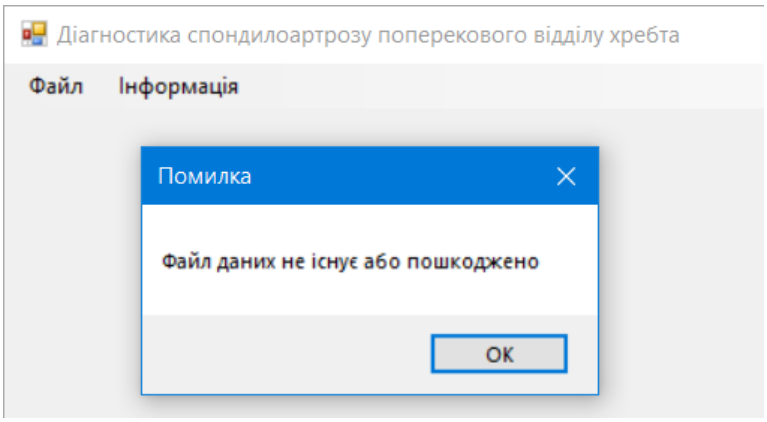

Рис. 2.3. Повідомлення про помилку при зчитуванні даних зі сховища даних

2.3 Підсистема проведення діагностики

Для проведення тестування потрібно вибрати команду «Файл»->«Почати тест» (якщо команда неактивна див. п.п.2.2). Після цього на екрані програми з'явиться питання та варіанти відповідей. Один з варіантів вибрано за замовчуванням, тому користувач не зможе не вибрати жодного варіанту. Коли користувач визначився з вибором, йому потрібно натиснути кнопку «Наступне питання» тощо. Продовжувати доти, поки не відповість на всі питання. В будь-який момент він може перервати виконання тесту та вийти з програми. Почати новий тест, доки не закінчено поточний, неможливо.

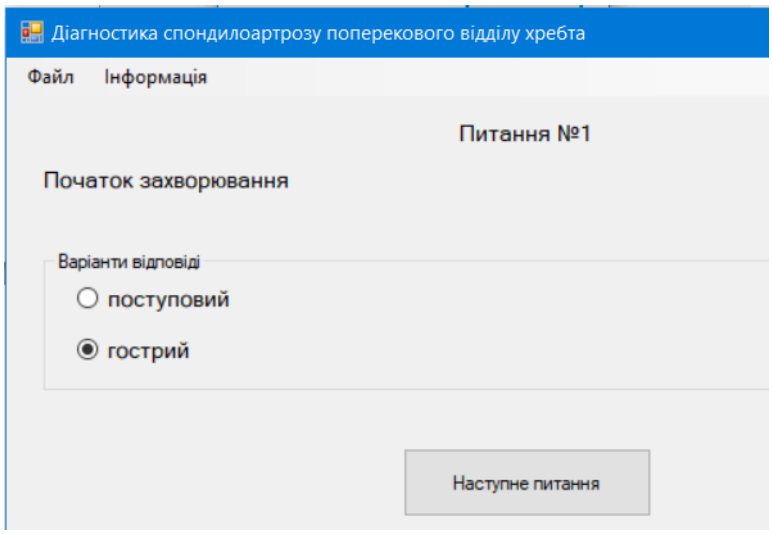

Рис. 2.4. Проведення тестування для діагности СПВХ

Після закінчення тестування на екран виводиться кількість балів та діагноз щодо захворювання на спондилоартроз поперекового відділу хребта.

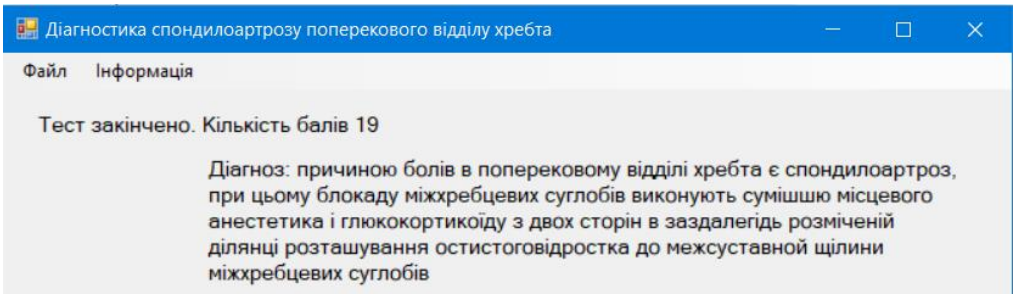

Рис. 2.5. Позитивний результат тестування для діагности СПВХ

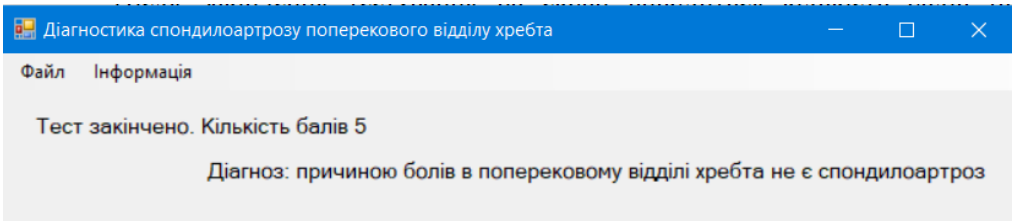

Рис. 2.6. Негативний результат тестування для діагности СПВХ

Діагноз може бути як позитивним, так і негативним.

2.4 Підсистема довідкової інформації

Для отримання інформації про програму потрібно вибрати команду «Інформація»

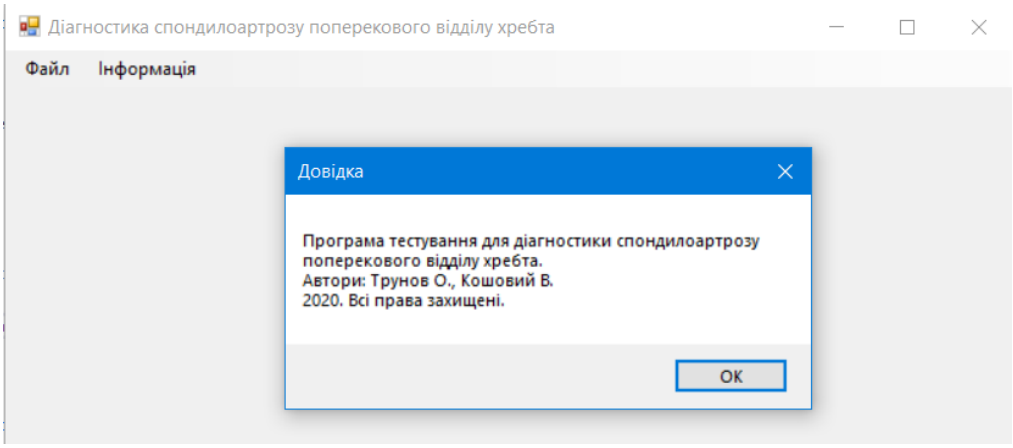

Рис. 2.7. Довідка про програму

3. РОБОТА З ФОРМАМИ

Розробка програмного забезпечення «Діагностика спондилоартрозу поперекового відділу хребта» виконана за допомогою наступних засобів програмування:

- Мова програмування C#(Sharp)
- Середовище програмування Visual Studio 2019
- Операційна система Windows 10 x64

3.1 Встановлення та налаштування Visual Studio 2019

Для створення додатків на C # будемо використовувати безкоштовне і повнофункціональне середовище розробки – Visual Studio Community 2019.

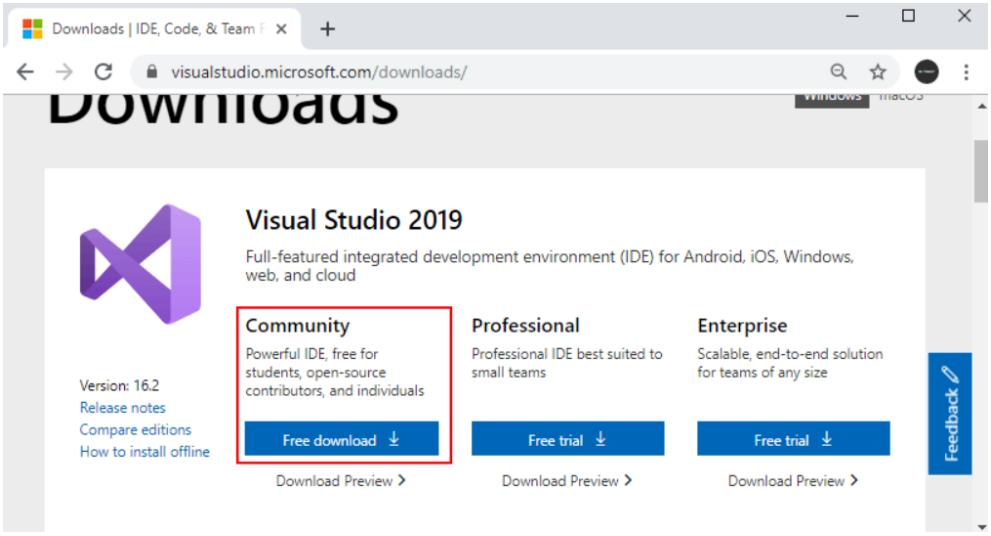

Рис. 3.1. Завантаженння Visual Studio Community 2019

Після завантаження запустимо програму установника. У вікні, нам буде запропоновано вибрати ті компоненти, які ми хочемо встановити разом з Visual Studio. Варто відзначити, що Visual Studio – дуже функціональне середовище розробки і дозволяє розробляти програми за допомогою безлічі мов і платформ. Нас буде цікавити перш за все C # і .NET Core. Тому в наборі робочих навантажень можна здійснювати установку компонентів тільки для пристрою «Кроссплатформенная розробка .NET Core». Можна вибрати і більше опцій або взагалі все опції, проте варто враховувати вільний розмір на жорсткому диску.

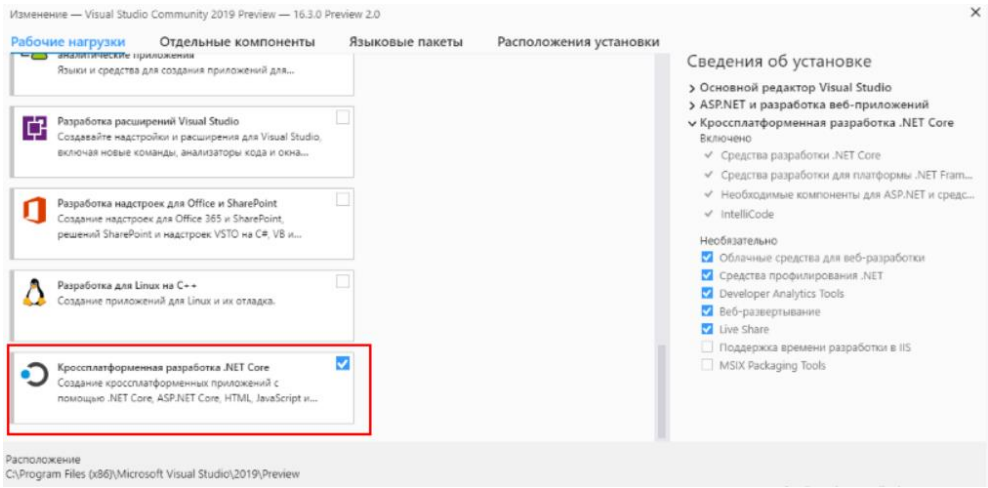

Рис. 3.2. Вибір опцій встановлення Visual Studio Community 2019

Після завершення установки створимо першу програму. Спочатку відкриємо Visual Studio. На стартовому екрані виберемо Create a new project (Створити новий проєкт).

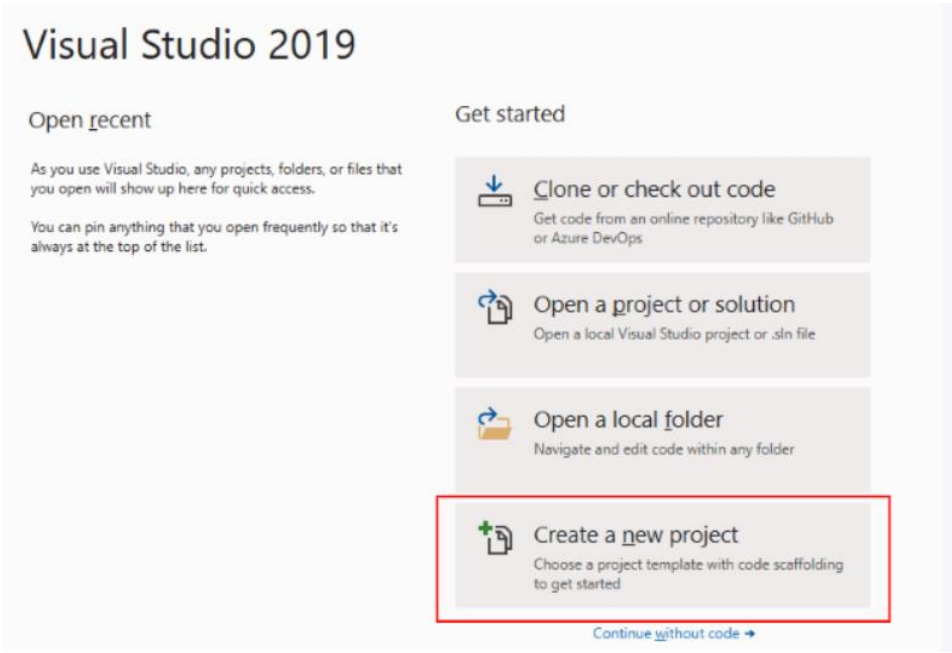

Рис. 3.3. Створення нового проєкту Visual Studio 2019

Наступним кроком виберемо тип проєкту Windows Forms (.Net Framework) і натиснемо кнопку Далі

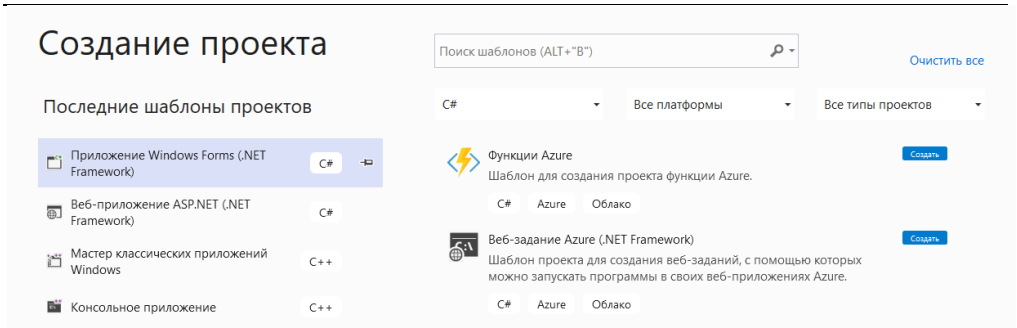

Рис. 3.4. Вибір типу нового проєкту Visual Studio 2019

Наступним кроком потрібно налаштувати новий додаток: вказати його ім’я та папку для зберігання проєкту

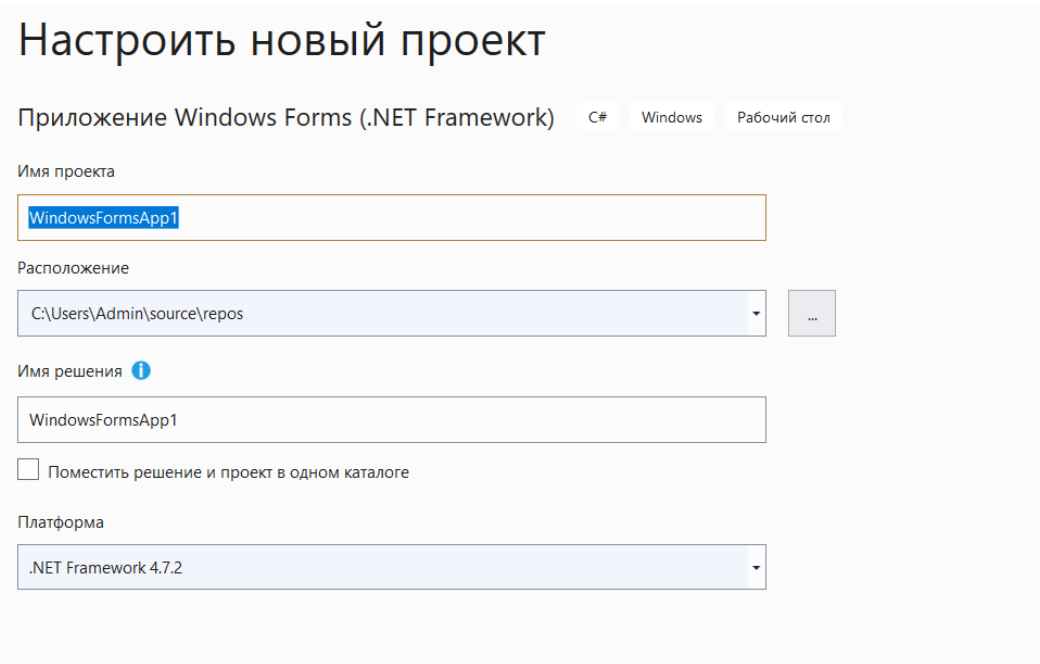

Рис. 3.5. Налаштування нового проєкту Visual Studio 2019

Після цього натиснемо кнопку Створити і створимо новий проєкт.

Після цього Visual Studio відкриє наш проєкт зі створеними за замовчуванням файлами. Більшу частину простору Visual Studio займає графічний дизайнер, який містить форму майбутньої програми. Поки що вона порожня і має тільки заголовок Form1. Праворуч знаходиться вікно файлів рішення/проєкту – Solution Explorer (Оглядач рішень). Там і знаходяться всі пов’язані з нашим додатком файли, в тому числі файли форми Form1.cs.

За допомогою спеціального вікна Properties (Властивості) праворуч Visual Studio надає нам зручний інтерфейс для управління властивостями елемента. Більшість цих

властивостей впливає на візуальне відображення форми. Стисло нагадаємо основні властивості:

- Name: встановлює ім'я форми – точніше ім'я класу, який успадковується від класу Form
- BackColor: вказує на фоновий колір форми. Клікнувши на цю властивість, ми зможемо вибрати той колір, який нам підходить зі списку запропонованих або кольорової палітри
- BackgroundImage: вказує на фонове зображення форми
- BackgroundImageLayout: визначає, як зображення, задане у властивості BackgroundImage, буде розташовуватися на формі.
- ControlBox: вказує, чи відображається меню форми. В даному випадку під меню розуміється меню найвищого рівня, де знаходяться іконка програми, заголовок форми, а також кнопки мінімізації форми і хрестик. Якщо ця властивість має значення false, то ми не побачимо ні іконку, ні хрестика, за допомогою якого зазвичай закривається форма
- Cursor: визначає тип курсора, який використовується на формі
- Enabled: якщо ця властивість має значення false, то вона не зможе отримувати введення від користувача, тобто ми не зможемо натиснути на кнопки, ввести текст в текстові поля і т.д.
- Font: задає шрифт для всієї форми і всіх поміщених на неї елементів управління. Однак, задавши у елементів форми свій шрифт, ми можемо тим самим перевизначити його
- ForeColor: колір шрифту на формі
- FormBorderStyle: вказує, як буде відображатися межа форми і рядок заголовка. Встановлюючи цю властивість в None, можна створювати зовнішній вигляд програми довільної форми
- HelpButton: вказує, чи відображається кнопка довідки форми
- Icon: задає іконку форми
- Location: визначає положення по відношенню до верхнього лівого кута екрана, якщо для властивості StartPosition встановлено значення Manual
- MaximizeBox: вказує, чи буде доступна кнопка максимізації вікна в заголовку форми
- MinimizeBox: вказує, чи буде доступна кнопка мінімізації вікна
- MaximumSize: задає максимальний розмір форми
- MinimumSize: задає мінімальний розмір форми
- Opacity: задає прозорість форми
- Size: визначає початковий розмір форми
- StartPosition: вказує на початкову позицію, з якою форма з'являється на екрані
- Text: визначає заголовок форми
- TopMost: якщо ця властивість має значення true, то форма завжди буде знаходитися поверх інших вікон
- Visible: чи видима форма, якщо ми хочемо приховати форму від користувача, то можемо поставити даній властивості значення false
- WindowState: вказує, в якому стані форма буде знаходитися при запуску: в нормальному, максимізованому або мінімізованому

Виберіть форму як елемент управління, тоді в цьому полі відображаються властивості, пов'язані з формою. В полі Text замість назви Form1 вкажіть наступну назву: *Діагностика спондилоартрозу поперекового відділу хребта.*

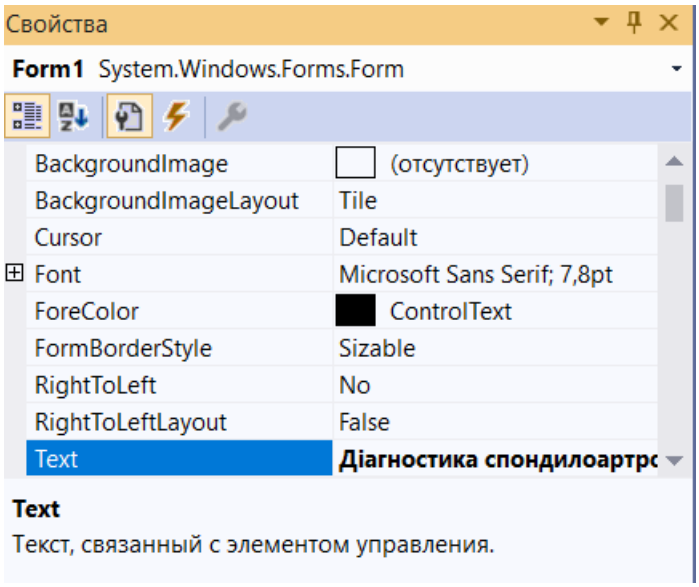

Рис. 3.6. Зміна заголовку форми Form1

3.2 Створення елементів інтерфейсу

Для створення меню в Windows Forms застосовується елемент MenuStrip. Даний клас успадкований від ToolStrip і тому успадковує його функціональність.

Найбільш важливі властивості компонента MenuStrip:

- Dock: Прикріплює меню до однієї зі сторін форми
- LayoutStyle: Задає орієнтацію панелі меню на формі. Може також, як і з ToolStrip, набувати наступних значень:
    - HorizontalStackWithOverflow: Розташування по горизонталі з переповненням – якщо довжина меню перевищує довжину контейнера, то нові елементи, що виходять за межі контейнера, не відображаються, тобто панель переповнюється елементами
    - StackWithOverflow: Елементи розташовуються автоматично з переповненням
    - VerticalStackWithOverflow: Елементи розташовуються вертикально з переповненням
    - Flow: Елементи розміщуються автоматично, але без переповнення – якщо довжина панелі меню менше довжини контейнера, то елементи, що виходять за межі, переносяться
    - Table: Елементи позиціонуються в вигляді таблиці
- ShowItemToolTips: Вказує, чи будуть відображатися спливаючі підказки для окремих елементів меню
- Stretch: Дозволяє розтягнути панель по всій довжині контейнера
- TextDirection: Задає напрямок тексту в пунктах меню
- MenuStrip: виступає свого роду контейнером для окремих пунктів меню, які представлені об’єктом ToolStripMenuItem

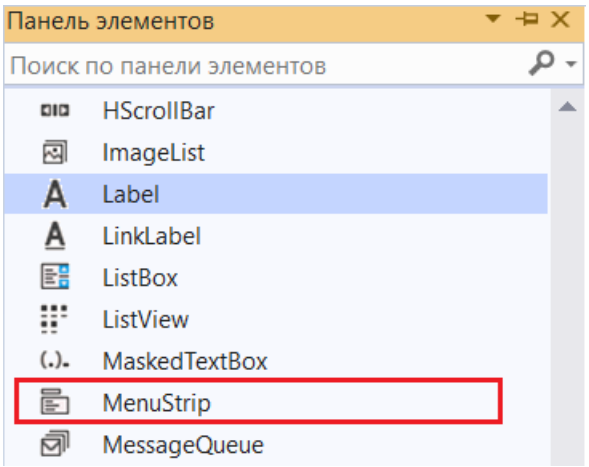

Рис. 3.7. Вибір інструмента для створення рядка меню

Створіть на формі наступні пункти меню: Файл та Інформація

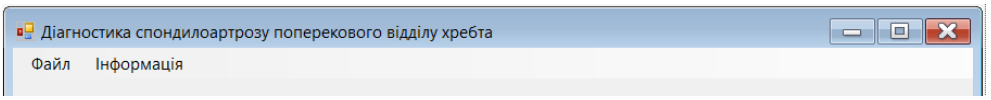

Рис. 3.8. Створення рядка меню

ToolStripMenuItem в конструкторі приймає текстову мітку, яка буде використовуватися в якості тексту меню. Кожен подібний об’єкт має колекцію DropDownItems, яка зберігає дочірні об’єкти ToolStripMenuItem. Тобто один елемент ToolStripMenuItem може містити набір інших об’єктів ToolStripMenuItem. Таким чином, утворюється ієрархічне меню або структура у вигляді дерева. Додайте до елемента меню Файл наступні пункти меню:

- Додати дані
- Читати дані
- Почати тест
- Вихід

3.3 Додавання елементів на форму

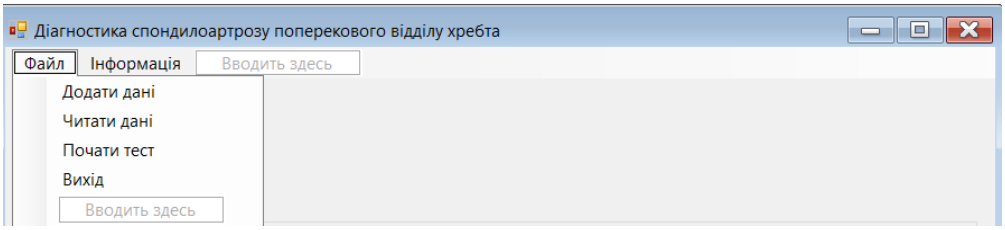

Рис. 3.9. Елементи пункту меню Файл

Зовнішній вигляд програми представляється нам переважно через форми. Форми є основними будівельними блоками. Вони надають контейнер для різних елементів управління. А механізм подій дозволяє елементам форми відгукуватися на введення, що внесені користувачем, і, таким чином, взаємодіяти з користувачем.

При відкритті проекту в Visual Studio в графічному редакторі ми можемо побачити візуальну частину форми – ту частину, яку ми бачимо після запуску програми і куди ми переносимо елементи з панелі управління. Але насправді форма приховує потужний функціонал, що складається з методів, властивостей, подій та інше. Розглянемо основні властивості форм.

Якщо ми запустимо додаток, то відобразиться одна порожня форма. Ще один файл – Form1.resx – зберігає ресурси форми. Як правило, ресурси використовуються для створення одноманітних форм відразу для декількох мовних культур. І більш важливий файл – Form1.cs, який в структурі проекту називається просто Form1, містить код або програмну логіку форми. За замовчуванням тут є тільки конструктор форми, в якому просто викликається метод InitializeComponent(), оголошений у файлі дизайнера Form1.Designer.cs. Саме з цим файлом ми і будемо працювати.

Для подальшої роботи за допомогою панелі елементів створіть на формі Form1 наступні компоненти, як вказано на малюнку. Зверніть увагу, що radioButton1 та radioButton2 потрібно згрупувати за допомогою groupBox1.

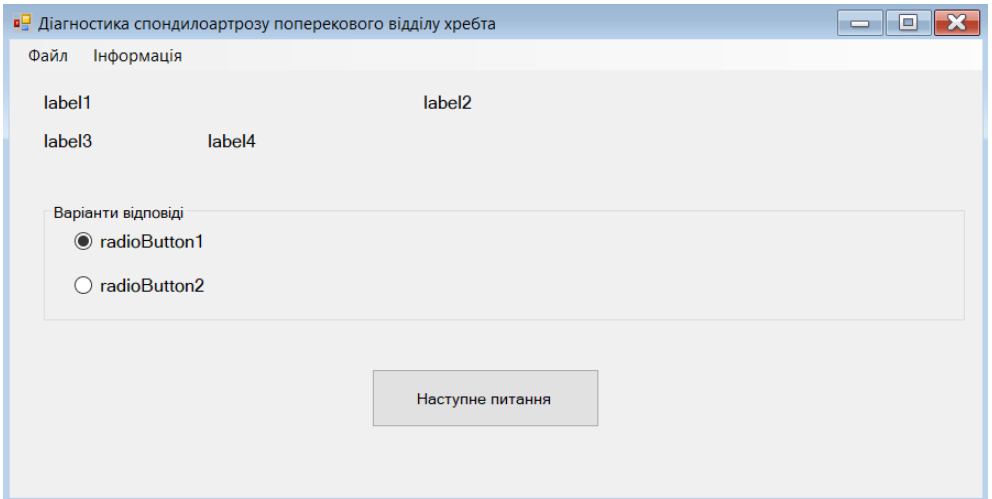

Рис. 3.10. Елементи форми

3.4 Створення нової форми

Щоб додати ще одну форму в проєкт, натиснемо на ім’я проєкту у вікні Solution Explorer (Оглядач рішень) правою кнопкою миші і виберемо Add (Додати) -> Windows Form (Форма)

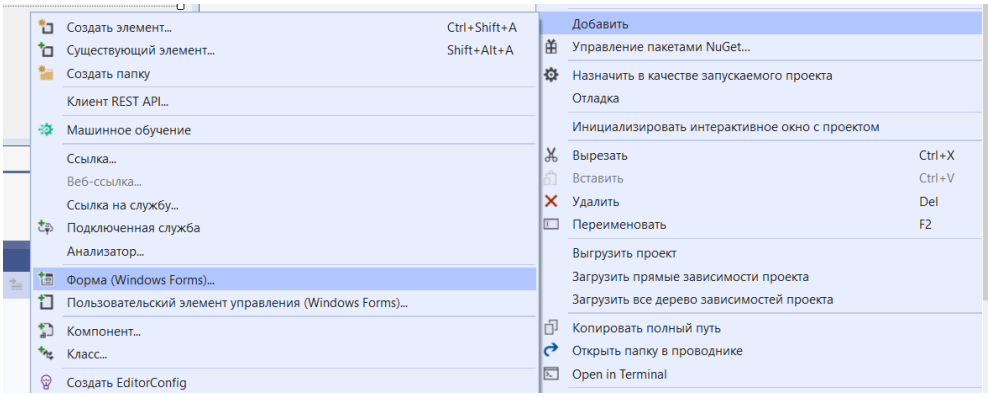

Рис. 3.11. Додавання до проекту нової форми

Дамо новій формі якесь ім’я, наприклад, Form2.cs:

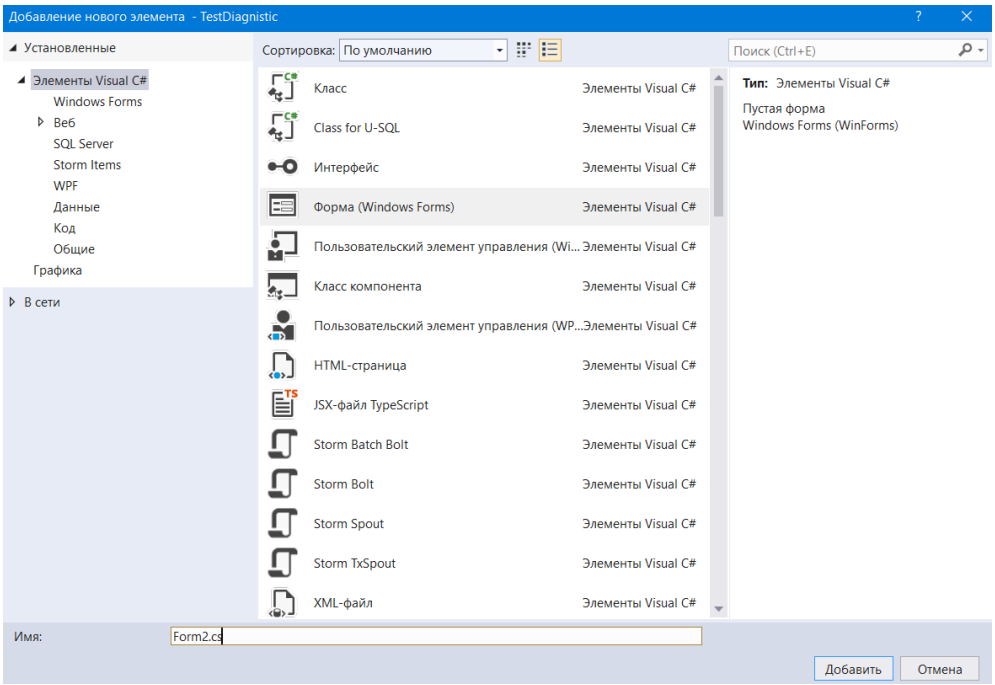

Рис. 3.12. Додавання нової форми

Отже, в наш проєкт була додана друга форма. Тепер спробуємо здійснити взаємодію між двома формами. Припустимо, перша форма після натискання на кнопку буде викликати другу форму. По-перше, виберемо пункт меню Файл->Додати дані формі Form1 кнопку і подвійним клацанням перейдемо в файл коду. Отже, ми потрапимо в обробник події натискання на пункт меню, який створюється за замовчуванням після подвійного клацання по кнопці. Введемо наступний код:

```
private void додатиДаніToolStripMenuItem_Click(object sender, EventArgs e)
{

    Form2 frm2 = new Form2(this);
    frm2.ShowDialog();

}
```

Рис. 3.13. Виклик нової форми

Оскільки в цьому випадку ключове слово this є посиланням на поточний об’єкт – об’єкт Form1, то при створенні другої форми вона буде отримувати її (посилання) і через неї керувати першою формою.

У формі Form2 потрібно створити наступні елементи та змінити заголовок форми у властивостях форми.

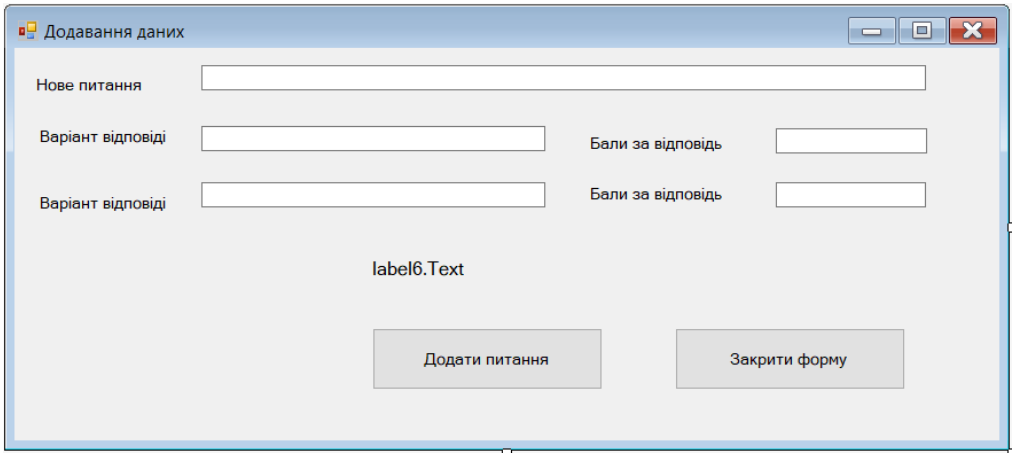

Рис. 3.14. Вигляд нової форми Form2

При роботі з декількома формами треба враховувати, що одна з них є головною – та, яка запускається першою в файлі Program.cs. Якщо у нас одночасно відкрито багато форм, то при закритті головної закривається весь додаток і разом з ним всі інші форми.

4. ПОДІЇ У ФОРМАХ. ЗБЕРІГАННЯ ДАНИХ ФОРМИ

Для взаємодії з користувачем в Windows Forms використовується певний механізм подій. Події в Windows Forms представляють стандартні події на C#, тільки що застосовуються до візуальних компонентів і підкоряються тим же правилам, що події в C#. Але створення обробників подій в Windows Forms все ж має деякі особливості.

Перш за все, в WinForms є певний стандартний набір подій, який здебільшого є у всіх візуальних компонентів. Окремі елементи додають свої події, але принципи роботи з ними будуть схожі. Щоб подивитися всі події елемента, нам треба вибрати цей елемент в поле графічного дизайнера і перейти до вкладки подій на панелі форм. Наприклад, події форми:

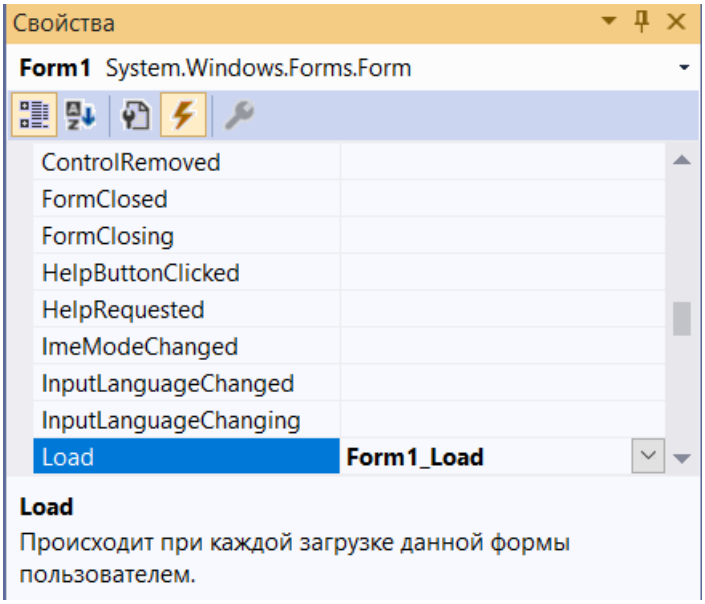

Рис. 4.1. Події, що відбуваються при завантаженні форми Form1

Щоб додати оброблювач, можна просто два рази натиснути по порожньому полю поруч з назвою події, і після цього Visual Studio автоматично згенерує обробник події. Наприклад, натиснемо для створення обробника для події Load. І в цьому полі відобразиться назва методу обробника події Load. За замовчуванням він називається Form1_Load.

Якщо ми перейдемо в файл коду форми Form1.cs, то побачимо автоматично згенерований метод Form1_Load:

```
private void Form1_Load(object sender, EventArgs e)
{
    label1.Text = "";
    label2.Text = "";
    label3.Text = "";
    label4.Text = "";
    початиТестToolStripMenuItem.Enabled = false; //команда меню Почати тест
    groupBox1.Visible = false;
    button5.Visible = false; //кнопка Наступне питання
}
```

Рис. 4.2. Обробник події при завантаженні форми Form1

При кожному завантаженні форми буде спрацьовувати код в обробнику Form1_Load. Як правило, більшість обробників різних візуальних компонентів мають два параметри: sender – об'єкт, який ініціював подію, і аргумент, який зберігає інформацію про подію (в даному випадку EventArgse).

4.1 Елементи управління форми

Елементи управління є візуальними класами, які отримують введені користувачем дані і можуть ініціювати різні події. Всі елементи управління успадковуються від класу Control і тому мають ряд загальних властивостей:

- Anchor: Визначає, як елемент буде розтягуватися
- BackColor: Визначає фоновий колір елемента
- BackgroundImage: Визначає фонове зображення елемента
- ContextMenu: Контекстне меню, яке відкривається при натисканні на елемент правою кнопкою миші. Задається за допомогою елемента ContextMenu
- Cursor: Представляє, як буде відображатися курсор миші при наведенні на елемент
- Dock: Задає розташування елемента на формі
- Enabled: Визначає, чи буде доступний елемент для використання. Якщо ця властивість має значення False, то елемент блокується.
- Font: Встановлює шрифт тексту для елемента
- ForeColor: Визначає колір шрифту
- Location: Визначає координати верхнього лівого кута елементу управління
- Name: Ім'я елемента керування
- Size: Визначає розмір елемента
- Width: ширина елемента
- Height: висота елемента
- TabIndex: Визначає порядок обходу елемента після натискання на клавішу Tab
- Tag: Дозволяє зберігати значення, асоційоване з цим елементом управління

Найбільш часто використовуваним елементом управління є кнопка чи пункт меню. Обробляючи подію натискання кнопки чи пункту меню, ми може виконувати ті чи інші дії.

При натисканні на кнопку чи на пункт меню на формі в редакторі Visual Studio, ми за замовчуванням потрапляємо в код обробника події Click, який буде виконуватися при натисканні:

```
private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Рис. 4.3. Обробник події закритті форми Form1

Для відображення на формі простого тексту, доступного тільки для читання, служить елемент Label. Щоб задати відображення тексту мітки, треба встановити властивість Text елемента.

```
label4.Text = "Діагноз: причиною болів в поперековому відділі хребта є спондилоартроз, " +
    "\nпри цьому блокаду міжхребцевих суглобів виконують сумішшю місцевого " +
    "\nанестетика і глюкокортикоїду з двох сторін в заздалегідь розміченій " +
    "\nділянці розташування остистоговідростка до межсуставной щілини " +
    "\nміжхребцевих суглобів";
```

Рис. 4.4. Встановлення чи змінення підпису мітки у коді програми

Для вибору одного з декількох варіантів використовується елемент RadioButton або перемикач. Перемикачі розташовуються групами, і включення одного перемикача означає відключення всіх інших.

Щоб встановити у перемикача включений стан, треба привласнити його властивості Checked значення true.

Для створення групи перемикачів, з яких можна б було вибирати, треба помістити кілька перемикачів в який-небудь контейнер, наприклад, в елементи GroupBox. Перемикачі. що знаходяться в різних контейнерах, будуть відноситися до різних груп. Наприклад, при натисненні кнопки Наступне питання на формі Form1 відбуваються наступні події:

```
public partial class Form1 : Form
{
    Quest[] quests = new Quest [100];
    long all_balls,num;
    Form2 frm2;
    int i;
    string str = "Питання №";
    ссылка: 1
    public Form1()
    {
        InitializeComponent();

    }

    ссылка: 1
    private void button5_Click_1(object sender, EventArgs e)
    {
        if (radioButton1.Checked == true)
            {
            all_balls+= Convert.ToInt32(quests[i].ball_1);
        }
            else
            {
            all_balls += Convert.ToInt32(quests[i].ball_2);
        }
```

Рис. 4.5. Перевірка параметру checked для елемента radioButton

Крім того, можливо програмно встановлювати та змінювати підписи елементів radioButton

```
i++;
if (i < num)
{

    label2.Text = str + Convert.ToString(i+1);
    label3.Text = quests[i].question;
    radioButton1.Text = quests[i].var_1;
    radioButton2.Text = quests[i].var_2;

}
```

Рис. 4.6. Динамічна зміна підпису елемента radioButton

Як правило, для виведення повідомлень застосовується елемент MessageBox. Однак, крім, власне, виведення рядка повідомлення, цей елемент може встановлювати ряд налаштувань, які визначають його поведінку.

Для виведення повідомлення в класі MessageBox передбачений метод Show, який має різні версії і може приймати ряд параметрів.

Тут застосовуються такі параметри:

- text: Текст повідомлення
- caption: Текст заголовка вікна повідомлення
- buttons: Кнопки, які використовуються в вікні повідомлення. Приймає одне зі значень перерахування MessageBoxButtons:
  - AbortRetryIgnore: Три кнопки Abort (Скасування), Retry (Повтор), Ignore (Пропустити)
  - OK: Одна кнопка OK
  - OKCancel: Дві кнопки OK і Cancel (Скасування)
  - RetryCancel: Дві кнопки Retry (Повтор) і Cancel (Скасування)
  - YesNo: Дві кнопки Yes і No
  - YesNoCancel: Три кнопки Yes, No і Cancel (Скасування)

Таким чином, в залежності від вибору вікно повідомлення може мати від однієї до трьох кнопок.

- icon: Значок вікна повідомлення. Може приймати одне з наступних значень перерахування MessageBoxIcon :
- Asterisk, Information: Значок, що складається з букви і в нижньому регістрі, вміщений в кружечок
- Error, Hand, Stop: Значок, що складається з білого знака "X" на колі червоного кольору.
- Exclamation, Warning: Значок, що складається з знаку оклику в жовтому трикутнику
- Question: Значок, що складається з знаку питання на периметрі кола
- None: Значок у повідомлення відсутній
- defaultButton: Кнопка, на яку за замовчуванням встановлюється фокус. Приймає одне зі значень перерахування MessageBoxDefaultButton:
  - Button1: Перша кнопка з тих, які задаються перерахуванням MessageBoxButtons
  - Button2: Друга кнопка
  - Button3: Третя кнопка
- options: Параметри вікна повідомлення. Приймає одне зі значень перерахування MessageBoxOptions:
  - DefaultDesktopOnly: Вікно повідомлення відображається на активному робочому столі.
  - RightAlign: Текст вікна повідомлення вирівнюється по правому краю
  - RtlReading: Всі елементи вікна розташовуються в зворотному порядку справа наліво
  - ServiceNotification: Вікно повідомлення відображається на активному робочому столі, навіть якщо в системі не зареєстрований жоден користувач

Нерідко використовується один параметр – текст повідомлення.

```
if (!(File.Exists(pathb)))
{
    MessageBox.Show("Файл даних не існує або пошкоджено", "Помилка");

}
```

Рис. 4.7. Виведення повідомлення про помилку читання даних з файлу

4.2 Читання і запис текстових файлів

Для роботи з текстовими файлами в просторі System.IO визначені спеціальні класи: StreamReader і StreamWriter.

Для запису в текстовий файл використовується клас StreamWriter. Деякі з його конструкторів, які можуть застосовуватися для створення об'єкта StreamWriter:

- StreamWriter (string path): Через параметр path передається шлях до файлу, який буде пов'язаний з потоком
- StreamWriter (string path, bool append): Параметр append вказує, чи треба додавати в кінець файлу дані або ж перезаписувати файл. Якщо одне true, то нові дані додаються в кінець файлу. Якщо одне false, то файл перезаписується заново
- StreamWriter (string path, bool append, System.Text.Encoding encoding): Параметр encoding вказує на кодування, яке буде застосовуватися при записі

Свою функціональність StreamWriter реалізує через такі методи:

- int Close(): Закриває записування файлу і звільняє всі ресурси
- void Flush(): Записує в файл дані, що залишилися в буфері, і очищує буфер.
- Task FlushAsync(): Асинхронна версія методу Flush
- void Write(string value): Записує в файл дані найпростіших типів, як int, double, char, string і т.д. Відповідно має ряд перевантажених версій для запису даних елементарних типів, наприклад, Write(char value), Write(int value), Write(double value)і т.д.
- Task WriteAsync(string value): Асинхронна версія методу Write
- void WriteLine(string value): Також записує дані, тільки після запису додає в файл символ закінчення рядка
- Task WriteLineAsync(string value): Асинхронна версія методу WriteLine

Розглянемо запис в файл на прикладі запису у файл даних, що введені в текстові поля форми Form2. Для цього створимо у формі Form1 спеціальний клас та визначимо в ньому методи для читання та запису даних.

```
public class Quest
{

    Ссылок: 3
    public string question { get; set; }
    Ссылок: 3
    public string var_1 { get; set; }
    Ссылок: 2
    public long ball_1 { get; set; }
    Ссылок: 3
    public string var_2 { get; set; }
    Ссылок: 2
    public long ball_2 { get; set; }

}
```

Рис. 4.8. Визначення класу і сетерів та гетерів цього класу

Тепер ми можемо визначити в коді об’єкт чи масив об’єктів даного класу та використати його для запису даних у файл даних.

```
ссылка: 1
private void button1_Click(object sender, EventArgs e)
{
    string pathb = @"quest.txt";

    if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" &&
        textBox4.Text != "" && textBox5.Text != "" && textBox4.Text != textBox5.Text)
    {
        Quest quest = new Quest { question = textBox1.Text, var_1 = textBox2.Text,
            var_2 = textBox3.Text, ball_1 = Convert.ToInt32(textBox4.Text), ball_2 = Convert.ToInt32(textBox5.Text) };
        string json = JsonConvert.SerializeObject(quest);
        using (StreamWriter sw = new StreamWriter(pathb, true, System.Text.Encoding.Default))
        {
            sw.WriteLineAsync(json);
            label6.Text = "Дані додано";
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
        }
    }
    else
    {
        label6.Text = "Помилка введення даних";
    }
```

Рис. 4.9. Запис даних елементів форми у файл даних

Клас StreamReader дозволяє нам легко зчитувати весь текст або окремі рядки з текстового файлу.

Деякі з конструкторів класу StreamReader:

- StreamReader (string path): Через параметр path передається шлях до зчитування файлу
- StreamReader (string path, System.Text.Encoding encoding): Параметр encoding задає кодування для читання файлу

Серед методів StreamReader можна виділити наступні:

- void Close(): Закриває зчитування файлу і звільняє всі ресурси
- int Peek(): Повертає наступний доступний символ, якщо символів більше немає, то повертає -1
- int Read(): Зчитує і повертає наступний символ в чисельному поданні. Має перевантажену версію: Read(char[] array, int index, int count), де array – масив, куди зчитуються символи, index – індекс в масиві array, починаючи з якого записуються зчитувальні символи, і count – максимальна кількість зчитувальних символів
- Task<int> ReadAsync(): Асинхронна версія методу Read
- string ReadLine(): Зчитує один рядок в файлі
- string ReadLineAsync(): Асинхронна версія методу ReadLine
- string ReadToEnd(): Зчитує весь текст з файлу
- string ReadToEndAsync(): Асинхронна версія методу ReadToEnd

```
private void прочитатиДаніToolStripMenuItem_Click(object sender, EventArgs e)
{
    string pathb = @"quest.txt";
    label1.Text = "";
    label4.Text = "";
    num = 0;
    i = 0;
    if (!(File.Exists(pathb)))
    {
        MessageBox.Show("Файл даних не існує або пошкоджено", "Помилка");

    }
    else
    {
        using (StreamReader sr = new StreamReader(pathb, System.Text.Encoding.Default))
        {
            string json;
            while ((json = sr.ReadLine()) != null)
            {
                Quest quest = JsonConvert.DeserializeObject<Quest>(json);
                quests[num] = quest;
                num++;

            }
        }
```

Рис. 4.10. Читання даних з файлу даних

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Абрамян, Михаил. Visual C# на примерах / Михаил Абрамян. – М.: БХВ-Петербург, 2015. – 436 с.
2. Албахари, Джозеф. C# 3.0. Справочник / Джозеф Албахари , Бен Албахари. – М.: БХВ-Петербург, 2009. – 499 с.
3. Блюстайн, Майкл. Изучаем MonoTouch. Создание приложений на платформе iOS с помощью C# и .NET / Майкл Блюстайн. – М.: ДМК Пресс, 2012. – 336 с.
4. Вагнер, Билл. C# Эффективное программирование / Билл Вагнер. – М.: ЛОРИ, 2018. – 224 с.
5. Гросс, Кристиан. C# 2008 и платформа .NET 3.5 Framework / Кристиан Гросс. – М.: Вильямс, 2009. – 480 с.
6. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел. – М.: Питер, 2014. – 312 с.
7. Зиборов, В. В. Visual C# 2012 на примерах / В. В. Зиборов. – М.: БХВ-Петербург, 2013. – 480 с.
8. Ишкова, Э. А. Самоучитель C#. Начала программирования / Э. А. Ишкова. – М.: Наука и техника, 2013. – 496 с.
9. Кариев, Ч. А. Разработка Windows-приложений на основе Visual C# (+ CD-ROM) / Ч.А. Кариев. – М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2007. – 767 с.
10. Культин, Никита. Основы программирования в Microsoft Visual C# 2010 / Никита Культин. – М.: БХВ-Петербург, 2013. – 389 с.
11. Ликнесс, Дж. Приложения для Windows 8 на C# и XAML / Дж. Ликнесс. – М.: Питер, 2016. – 368 с.
12. Магда, Ю. С. NI Measurement Studio. Практика разработки систем измерения и управления на C# / Ю. С. Магда. – М.: ДМК Пресс, 2016. – 189 с.
13. Нейгел, Кристиан. C# 2008 и платформа .NET 3.5 для профессионалов / Кристиан Нейгел и др. – М.: Вильямс, 2008. – 1392 с.
14. Ник, Рендольф. Visual Studio 2010 для профессионалов / Рендольф Ник. – М.: Диалектика / Вильямс, 2010. – 1184 с.
15. Рихтер, Джеффри. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Джеффри Рихтер. – М.: Питер, 2016. – 365 с.
16. Троелсен, Эндрю Язык программирования C# 2010 и платформа .NET 4 / Эндрю Троелсен. – Москва: Огни, 2011. – 1392 с.

ДОДАТОК А

Код програмного забезпечення

```
//Код для форми Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json;
using System.Runtime.Serialization;
using System.IO;
using System.Collections;

namespace TestDiagnistic
{
   public partial class Form1 : Form
   {
      Quest[] quests = new Quest [100];
      long all_balls,num;
      Form2 frm2;
      int i;
      string str = "Питання №";
      public Form1()
      {
         InitializeComponent();

      }

      private void button5_Click_1(object sender, EventArgs e)
      {
         if (radioButton1.Checked == true)
            {
            all_balls+= Convert.ToInt32(quests[i].ball_1);
         }
            else
            {
            all_balls += Convert.ToInt32(quests[i].ball_2);
         }
         i++;
         if (i < num)
         {
```

```
                label2.Text = str + Convert.ToString(i+1);
                label3.Text = quests[i].question;
                radioButton1.Text = quests[i].var_1;
                radioButton2.Text = quests[i].var_2;

            }
            else
            {
                label1.Text = "Тест закінчено. Кількість балів "+Convert.ToString(all_balls);
                label2.Text = "";
                label3.Text = "";
                button5.Visible = false;
                if (all_balls <= 10)
                {
                    label4.Text = "Діагноз: причиною болів в поперековому відділі хребта
не є спондилоартроз";
                }
                else
                {
                    label4.Text = "Діагноз: причиною болів в поперековому відділі хребта є
спондилоартроз, \nпри цьому блокаду міжхребцевих суглобів виконують сумішшю
місцевого \nанестетика і глюкокортикоїду з двох сторін в заздалегідь розміченій
\nділянці розташування остистоговідростка до межсуставной щілини \nміжхребцевих
суглобів";
                }
                groupBox1.Visible = false;
                початиТестToolStripMenuItem.Enabled = true;
                додатиДаніToolStripMenuItem.Enabled = true;
                прочитатиДаніToolStripMenuItem.Enabled = true;
            }
        }

        private void додатиДаніToolStripMenuItem_Click(object sender, EventArgs e)
        {

            Form2 frm2 = new Form2(this);
            frm2.ShowDialog();
            label1.Text = "";
            label4.Text = "";
            початиТестToolStripMenuItem.Enabled = false;
        }

        private void прочитатиДаніToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string pathb = @"quest.txt";
            label1.Text = "";
            label4.Text = "";
            num = 0;
            i = 0;
```

```
            if (!(File.Exists(pathb)))
            {
                MessageBox.Show("Файл даних не існує або пошкоджено", "Помилка");

            }
            else
            {
                using (StreamReader sr = new StreamReader(pathb, Sys-
tem.Text.Encoding.Default))
                {
                    string json;
                    while ((json = sr.ReadLine()) != null)
                    {
                        Quest quest = JsonConvert.DeserializeObject<Quest>(json);
                        quests[num] = quest;
                        num++;

                    }
                }

                початиТестToolStripMenuItem.Enabled = true;
                label1.Text = "Кількість питань " + Convert.ToString(num);
            }
        }

        private void початиТестToolStripMenuItem_Click(object sender, EventArgs e)
        {
            i = 0;
            label1.Text = "";
            label4.Text = "";
            початиТестToolStripMenuItem.Enabled = false;
            додатиДаніToolStripMenuItem.Enabled = false;
            прочитатиДаніToolStripMenuItem.Enabled = false;
            groupBox1.Visible = true;
            button5.Visible = true;
            all_balls = 0;
            label2.Text = str + Convert.ToString(i + 1);
            label3.Text = quests[i].question;
            radioButton1.Text = quests[i].var_1;
            radioButton2.Text = quests[i].var_2;
        }

        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void інформаціяToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            MessageBox.Show("Програма тестування для діагностики спондилоартрозу поперекового відділу хребта.\nАвтори: Трунов О., Кошовий В. \n2020. Всі права захищені.", "Довідка");
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            label1.Text = "";
            label2.Text = "";
            label3.Text = "";
            label4.Text = "";
            початиТестToolStripMenuItem.Enabled = false;
            groupBox1.Visible = false;

            button5.Visible = false;
        }
    }
    public class Quest
    {

        public string question { get; set; }
        public string var_1 { get; set; }
        public long ball_1 { get; set; }
        public string var_2 { get; set; }
        public long ball_2 { get; set; }

    }
}
//Код для форми Form2
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Collections;
using Newtonsoft.Json;
using System.Runtime.Serialization;

namespace TestDiagnistic
{
    public partial class Form2 : Form
    {

        public Form2(Form1 f1)
```

```
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string pathb = @"quest.txt";

            if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" &&
                textBox4.Text != "" && textBox5.Text != "" && textBox4.Text != text-
Box5.Text)
            {
                Quest quest = new Quest { question = textBox1.Text, var_1 = textBox2.Text,
                    var_2 = textBox3.Text, ball_1 = Convert.ToInt32(textBox4.Text), ball_2 =
Convert.ToInt32(textBox5.Text) };
                string json = JsonConvert.SerializeObject(quest);
                using (StreamWriter sw = new StreamWriter(pathb, true, Sys-
tem.Text.Encoding.Default))
                {
                    sw.WriteLineAsync(json);
                    label6.Text = "Дані додано";
                    textBox1.Text = "";
                    textBox2.Text = "";
                    textBox3.Text = "";
                    textBox4.Text = "";
                    textBox5.Text = "";
                }
            }
            else
            {
                label6.Text = "Помилка введення даних";
            }

            }

        private void Form2_Load(object sender, EventArgs e)
        {
            label6.Text = "";
        }
    }
}
```

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Віталій Володимирович
КОШОВИЙ
Олександр Миколайович
ТРУНОВ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «ДІАГНОСТИКА СПОНДИЛОАРТРОЗУ ПОПЕРЕКОВОГО ВІДДІЛУ ХРЕБТА»

Навчальний посібник
для підготовки бакалаврів у галузі знань
12 «Інформаційні технології» та магістрів за спеціальністю
174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Редактор *О. Михайлова*
Комп'ютерна верстка, дизайн обкладинки *К. Гросу-Грабарчук*
Друк *С. Волинець.* Фальцювально-палітурні роботи *О. Мішалкіна.*

Підписано до друку 07.11.2023.
Формат 60x84$^{1}/_{16}$. Папір офсет.
Гарнітура «Times New Roman». Друк ризограф.
Ум. друк. арк. 2. Обл.-вид. арк. 1.
Тираж 100 пр. Зам. № 6626.

Видавець і виготовлювач: ЧНУ ім. Петра Могили.
54003, м. Миколаїв, вул. 68 Десантників, 10.
Тел.: 8 (0512) 50–03–32, 8 (0512) 76–55–81, e-mail: rector@chmnu.edu.ua.
Свідоцтво суб'єкта видавничої справи ДК № 6124 від 05.04.2018.