Лабораторна робота 8. Підвищення зручності використання застосувань

Вправа 1. Створення контекстної довідки (2 бали)
Вправа 2. Використання довідкового файла (2 бали)
Вправа 3. Додавання спливаючих підказок (2 бали)
Вправа 4. Автоматичний вибір мови при запуску застосування (3 бали)
Вправа 5. Локалізація застосування (4 бали)

Мета роботи: Вивчення засобів для підвищення зручності роботи користувачів та отримання навичок зі створення контекстної довідки, спливаючої підказки, а також файлів з довідковою інформацією.

Вправа 1. Створення контекстної довідки (2 бали)

Важливою частиною будь-якого застосування є зрозуміла і точна документація. Забезпечити ваше застосування довідкою дозволяє компонент **HelpProvider**.

1. Скопіюйте Windows-застосування **WinAsynchMethod** (лабораторна робота №7, Вправа №3) у теку для вправ лабораторної роботи №8, Вправа №1.
2. Запустіть його.
3. Відкрийте форму в режимі конструктора.
4. Додайте елемент управління **HelpProvider** на форму.
5. Виділіть поле **txbA** для відображення її властивостей.
6. Для властивості **HelpString on helpProvider1** задайте значення **For input integer A**.
7. Побудуйте і запустіть застосування.
8. Перемістіться по формі, використовуючи клавішу **Tab**, до тих пір, поки поле **txbA** не опиниться у фокусі.
9. Натисніть на клавішу **F1** для відображення контекстної довідки для поля **txbA**.

Прості форми зазвичай у своєму заголовку мають кнопку з знаком питання, при натисненні на яку курсор міняє свій вигляд на зображення з питанням. При клацанні на вибраному елементі управління з'являється його короткий опис (підказка).

Створіть подібну функціональність на формі проекту **WinAsynchMethod**:

1. Додайте до наявних властивостей форми наступні властивості:

Властивість	Значення
MaximizeBox	False
MinimizeBox	False
HelpButton	True
FormBorderStyle	FixedDialog

2. Для полів введення **txbA, txbB** і двох кнопок у властивості **ShowHelp on helpProvider1** кожного з цих елементів встановіть значення **True**.

3. Текст, введений в поле властивості **HelpString on helpProvider1**, з'являтиметься в якості підказки для конкретного елемента. Встановіть наступні значення цієї властивості для кожного елемента:

txbA	For input integer A
txbB	For input integer B
btnRun	Sum
btnWork	Start work

4. Побудуйте і запустіть застосування.
5. Для активації контекстної довідки натисніть на кнопку **"?"**, розташовану в правому верхньому кутку застосування.
6. Натисніть на будь-яку кнопку, з'явиться маленьке віконце, яке пояснює, що відбувається при її натисненні.

Вправа 2. Використання довідкового файла (2 бали)

1. Скопіюйте Windows-застосування **WinAsynchMethod** (лабораторна робота №7, Вправа №3) у теку для вправ лабораторної роботи №8, Вправа №2.
2. Запустіть його.
3. Відкрийте форму в режимі конструктора.
4. Додайте елемент управління **HelpProvider** на форму (якщо він не був доданий раніше).
5. У теці з рішенням створіть файл довідки, наприклад, документ Microsoft Word. Текст вкажіть довільний.
6. Для елемента **helpProvider1** у властивості **HelpNamespace** вкажіть шлях до файла довідки (у мене: **G:\SPandOS\Lab_08\Example_2\Що таке DirectX і навіщо він потрібний.docx**).
7. Реалізуйте можливість виклику файла довідки створенням або команди меню, або кнопки (і команда меню і кнопка може називатися, наприклад, **Help**. Тут кнопка).
8. Створіть обробник події вибору файла довідки (**button1_Click**). У тілі обробника вкажіть наступний рядок:

```
Help.ShowHelp(this, helpProvider1.HelpNamespace);
```

9. Побудуйте і запустіть застосування.
10. Виберіть команду виклику довідки. Перевірте, що відкрився потрібний файл.

Вправа 3. Додавання спливаючих підказок (2 бали)

Компонент **ToolTip** дозволяє призначити елементам управління підказки. Вони з'являються у вікнах, коли миша знаходиться над елементом управління, і можуть надавати користувачеві короткі відомості про нього.

1. Скопіюйте Windows-застосування **WinAsynchMethod** (лабораторна робота №7, Вправа №3) у теку для вправ лабораторної роботи №8, Вправа №3.
2. Запустіть його.
3. Додайте на форму елемент управління **ToolTip**.
4. У вікні **Properties** розташованих на формі елементів і в самій формі з'явилася властивість **ToolTip on toolTip1**. Встановіть наступні значення цієї властивості для кожного з елементів:

txbA	For input integer A
txbB	For input integer B
btnRun	Sum
btnWork	Start work

5. Побудуйте і запустіть застосування. Переконайтесь, що при наведенні курсора на елемент управління з'являється його підказка.

Вправа 4. Автоматичний вибір мови при запуску застосування (3 бали)

При поширенні застосування часто виникає потреба забезпечувати користувачам можливість працювати у своєму мовному середовищі. Через це, при розробці застосувань доводиться замислюватися про переклад інтерфейсу користувача іншими мовами. На практиці використовується два способи вирішення цієї проблеми, ***перший***: створюється локальна версія цілком на одній мові і ***другий***: програми містять багатомовний інтерфейс, що дозволяє міняти оформлення застосування безпосередньо під час роботи.

Під час встановлення операційної системи Windows (у мене Windows 10) при визначенні регіональних параметрів користувачеві пропонується вибрати мову стандартів і форматів. Вибране значення доступне для зміни в подальшому - меню **Пуск → Парамметры → Время и язык → Регион и язык → Страна или регион**.

У цій вправі ви створите застосування, яке автоматично визначатиме встановлену мову стандартів і виводить відповідний інтерфейс користувача.

1. Створіть нове Windows-застосування і назвіть його **WinLanguage**.
2. Додайте на форму кнопку і головне меню.
3. Встановіть наступні значення властивості **Text** для елементів:

головне меню - **menu**,
перша команда - **command one**,
друга команда - **command two**,
кнопка - **Close**,
для самої форми - **Form**.

4. Для кнопки реалізуйте обробник, який закриває форму:

```
this.Close();
```

5. Для форми встановіть властивості **Localizable** значення **True**. Ця властивість дозволяє ***підтримку багатомовного інтерфейсу***.
6. У властивості **Language** виберіть значення **English** (United States).
7. Побудуйте застосування. У результаті вийшла версія програми з інтерфейсом англійською мовою.
8. У властивості **Language** виберіть **Ukraine** (Ukraine).
9. Змініть властивість **Text** елементів, замінивши назви англійською мовою відповідними назвами українською.
10. Перейдіть в код форми і підключіть простір імен **Threading**:

```
using System.Threading;
```

11. У конструкторі форми до **InitializeComponent();** встановіть культуру інтерфейсу користувача рівній поточній культурі:

```
Thread.CurrentThread.CurrentUICulture = Thread.CurrentThread.CurrentCulture;
```

12. Побудуйте і запустіть застосування. Середовище CLR перевіряє встановлену мову і виводить застосування з інтерфейсом на мові, встановленій на вкладці **"Регіональні параметри"** в налаштуваннях інструменту **"Мова і регіональні параметри"**.
13. Закрийте застосування. Перейдіть в **"Панель управління"** і встановіть іншу мову (наприклад, "Англійський (США)") на вкладці **"Регіональні параметри"** в налаштуваннях інструменту **"Мова і регіональні параметри"**. Знову запустіть застосування. Тепер інтерфейс застосування має бути на іншій мові (наприклад, англійській).
14. При локалізації застосування середовище Visual Studio .NET створює збірку, в якій зберігаються всі дані про застосування. У вікні **Solution Explorer** натисніть на кнопку (**Show All Files**) для перегляду доданих файлів. Назви файлів-ресурсів (**Form1.en-US і Form1.uk-UA**) містять в собі вказівку на мову (перша частина - **en** або **uk**) і регіон (друга частина **US** або **UA**).

Вправа 5. Локалізація застосування

Реалізувати ***локалізацію***, тобто надати інтерфейс користувача, характерний для поточного регіону, можна з допомогою вбудованих у Visual Studio засобів локалізації.

Visual Studio дозволяє створювати альтернативні версії форм залежних від регіональної культури та автоматично управляє пошуком ресурсів, відповідних цій культурі.

Для інтерфейсу користувача культура надається екземпляром **CultureInfo** і відрізняється від властивості **CulturInfo.CurrentCulture**. Тоді як вона визначає формат, застосовуваний до даних, які системно форматуються, **CurrentUICulture** визначає ресурси, які завантажуються в локалізовані форми під час виконання. Культура інтерфейсу користувача встановлюється у властивості **CurrentThread.CurrentUICulture**.

У цій вправі ви локалізуєте інтерфейс користувача застосування і додасте в нього локалізовані рядкові ресурси.

Локалізація форми Windows-застосування