Лекція №5 ПОіП «Інструментальні системи програмування. Середовище розробки Visual Studio.NET »

(Модуль 1 -)

План лекції.

До інструментальних систем, які відносять до системного програмного забезпечення, належать засоби для розробки та налагодження програм, а саме: системи програмування та *системи керування БД (СКБД)*.

Поняття системи програмування

Системи програмування включають в себе мову програмування, транслятор, різні обслуговувальні програми для редагування текстів і налагодження програм. Ці засоби служать для розробки нових програм. Комп'ютер "розуміє" і може виконувати програми у машинному коді. Кожна команда при цьому має вигляд послідовності нулів й одиниць. Писати програми машинною мовою дуже незручно, а їх надійність низка. Тому програми розробляють мовою, зрозумілою людині (інструментальна мова або алгоритмічна мова програмування), після чого спеціальною програмою, яка називається транслятором, текст програми перекладається (транслюється) на машинний код.

Транслятори бувають двох типів:

інтерпретатори;

компілятори.

Інтерпретатор читає один оператор програми, аналізує його і відразу виконує, після чого переходить до оброблення наступного оператора. **Компілятор** спочатку читає, аналізує та перекладає на машинний код усю програму і тільки після завершення всієї трансляції ця програма виконується. Інструментальні мови поділяються на мови **низького рівня** (близькі до машинної мови) та мови **високого рівня** (близькі до мови людини). До мов низького рівня належать асемблери, а високого - Pascal, Basic, C/C++, мови баз даних і т.д. Систему програмування, крім

транслятора, складають текстовий редактор, компонувальник, бібліотека стандартних програм, налагоджувач, візуальні засоби автоматизації програмування. Прикладами таких систем є Delphi, Visual Basic, Visual C++, Visual FoxPro та ін.

Склад системи програмування:

- мова програмування;
- транслятор,
- текстовий редактор,
- компонувальник,
- бібліотека стандартних програм,
- налагоджувач,
- візуальні засоби автоматизації програмування.

Класифікацію систем програмування подано в табл. 1.2.

Таблиця 1.2 КЛАСИФІКАЦІЯ СИСТЕМ ПРОГРАМУВАННЯ

Ознака	Назва	Приклади
Ступінь алгоритмізації	Процедурно-орієнтована	Паскаль, Сі, Бейсік, КОБОЛ, ФОРТРАН
	Проблемно-орієнтована	ПРОЛОГ, ЛІСП
	Об'єктно-орієнтована	Visual Basic, Visual FoxPro, Delfi, Smaltalk
Залежність від конструктивних особливостей ЕОМ	Машинонезалежна	Паскаль, Сі, Бейсік, КОБОЛ, ФОРТРАН, ПЛ/1
	Машинозалежна	Ассемблер

Спинимося на різновидах систем програмування *за ступенем алгоритмізації*:

- **процедурно-орієнтована** — мова, що призначена для опису алгоритмів розв'язування задач у вигляді послідовності операторів, тобто для опису процедур розв'язування задачі;
- **проблемно-орієнтована** — мова, за допомогою якої виконується опис постановки задачі (проблеми) та зазначаються вхідні дані. При цьому передбачається, що алгоритм розв'язування задачі буде побудовано автоматично під час трансляції програми у послідовність машинних кодів (мова логічного програмування);
- **об'єктно-орієнтована** — система, що реалізує концепцію поєднання даних із програмами їх обробки (механізм приховування даних) та організації доступу до даних лише за допомогою певних інтерфейсів.

Різновиди систем програмування залежно від конструктивних особливостей ЕОМ:

машинонезалежна — мова, в якій не використовуються особливості конструкції конкретної ЕОМ, тобто її можна реалізувати на будь-якій моделі ЕОМ універсального призначення;

машинозалежна — мова, що призначена для реалізації на ЕОМ певного типу.

Інструментальні засоби середовища розробки Visual Studio.NET.

Середовище розробки Visual Studio .NET, як і середовище всіх сучасних засобів розробки, може бути налагоджене згідно потребам конкретного користувача. Розглянемо способи налаштування середи Visual Studio .NET, а також взнаємо, які вікна і інструменти вона містить.

Вікна середовища розробки

Розглянемо вікна, наявні в середовищі розробки (Integrated Development Environment, **IDE**) Visual Studio .NET.

Як і більшість сучасних засобів, середовище розробки Visual Studio .NET містить меню і набір інструментальних панелей.

Вікна:

Toolbox - список елементів управління, які можна використовувати на формах додатка.

Solution Explorer - склад проектів, що входять в рішення, у вигляді ієрархічної структури, а також зв'язку між проектами і їх компонентами (з яких проектів складається рішення і які файли входять до складу цих проектів). Компонентами проектів можуть бути форми, класи, модулі, а також інші файли, які потрібні для створення додатка.

Class View (доступно за допомогою команди меню View | Class View) дозволяє проглянути список властивостей і методів створених в додатку класів

Server Explorer (команда меню View | Server Explorer), дозволяє переглядати відомості про служби, що виконуються на конкретних серверах. До таких служб, зокрема, відносяться служби Crystal Reports Services, журнал подій, черги повідомлень, служби серверів баз даних, таких як Microsoft SQL Server.

Properties (команда меню View | Properties Window) призначене для зміни властивостей елементів управління і інших класів створюваного застосування.

Object Browser, так само як і вікно Class View, дозволяє проглянути список класів, їх властивостей і методів. Проте Object Browser дозволяє проглянути всі компоненти, на які посилається клас, а також, при необхідності, компоненти, на які немає заслань в даному проекті, тоді як за допомогою вікна Class View можна переглядати відомості лише про класи з даного проекту. За допомогою Object Browser можна також проглянути оголошення властивостей і методів.

Вікно **Task List**, містить список завдань (TODO list), помилок компіляції і іншу інформацію.

Режими роботи середовища розробки

Середовище розробки Visual Studio .NET містить двох типів вікон:

- вікна інструментів
- вікна документів.

Вікна інструментів (частина з яких була описана вище) доступні за допомогою команд меню View і деяких інших, і їх доступність залежить від типа додатка і від того, які модулі розширення (додаткові утиліти і інструменти, у тому числі проведені сторонніми розробниками) додані до середи розробки.

У вікнах же документів можна редагувати компоненти проектів. Вікна документів призначені для редагування компонентів проектів. Їх взаємне розташування залежить від вибраного режиму відображення вікон в середі розробки.

Редактор коду

Після набору імені об'єкту і введення крапки на екрані з'являється список властивостей і методів даного об'єкту. При введенні імені методу можна побачити на екрані опис методу і його параметрів.

Вікно редагування можна розбити на декілька частин, в яких відображуватимуться різні фрагменти коди. Допустимо також відображувати друге вікно редагування за допомогою пункту меню Window | New Window.

У редакторові коди можна здійснювати контекстний пошук і заміну тексту в поточній процедурі, поточному модулі або у виділеному фрагменті коду.

РОБОТА З ФОРМАМИ І ЕЛЕМЕНТАМИ УПРАВЛІННЯ

У додатках VB.Net ФОРМИ – об'єкти самого високого рівня, і проектування будь-якого застосування завжди починається із створення форми.

Форми – це контейнери для елементів управління, що входять до складу призначеного для користувача інтерфейсу. Форми, які складають візуальний інтерфейс, називаються Windows-форми.

Діалогові окна- форми особливого типа з досить обмеженими функціями, з їх допомогою у користувача запрошують якісь дані. Властивість DialogResult вказує яка кнопка була нариснута користувачем.

DialogResult	Отримує або встановлює результат діалогу для форми.

Значення	**Опис**
Abort	Діалогове вікно закрите за допомогою кнопки Abort
Cancel	-//-
Ignore	-//-
No	-//-
None	Діалогове вікно ще не закрите
Ok	-//-
Retry	-//-

Yes -//-

Не обов'язково щоб в діалоговому вікні були ці кнопки. Установку властивості можна проводити програмно. Коли ви привласнюєте значення властивості, Діалогове вікно закривається і не треба використовувати метод **Close**.Например, для кнопки Ок, що завершує введення даних в цьому вікні, обробник події міститиме код:

```
Me.DialogResult= DialogResult.Ok
```

Використання декількох форм в проекті

У додатках, що використовують декілька форм, необхідно маніпулювати ними.

Для отримання доступу до форми з іншої форми, потрібно:

1. Створити змінну, що містить послання на цю форму і помістити її поза кодом процедур:

```
Public Class Form1
    Dim frm as New Form2
End class
```

2. Маючи таке оголошення, можна вивести можна вивести на екран **Form2** з форми **Form1** з обробника події кнопки **Click** або команди меню след.образом:

   Frm.Show()

   Метод відкриває форму в немодальному режимі, тобто обидві форми мають на робочому столі рівні права, і користувач може вільно перемикатися між ними. Щоб відкрити форму в модальному режимі, тобто щоб користувач не зміг повернутися в першу форму, поки відкрита друга, необхідно виконати слід. інструкцію:

   Frm.ShowDialog()

Спільне використання змінних в декількох формах:

Змінні оголошуються в розділі оголошень форми поза процедурами за допомогою ключових слів **Public Shared**.

Public Shared Name as String

Для доступу до загальнодоступною зміною, оголошеною в першій формі, з коду іншої форми потрібно передувати імені змінної ім'ям форми:

Frm.Name=”pppp”

Елементи управління MenuItem

св-во Enabled (enable [I'neIbl]) - команда в даний момент недоступна

Елементи управління контексное меню ContexMenu стр.11

Елементи управління TextBox

PasswordChar = * не можна ні скопіювати ні вирізувати текст

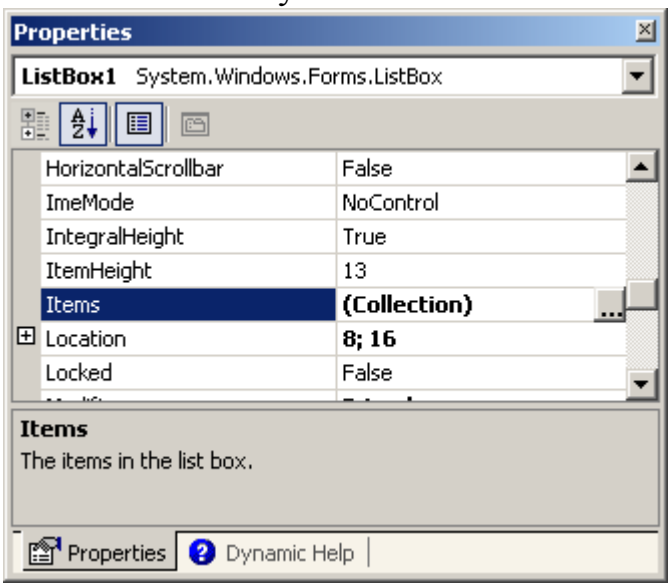
Елементи управління ListBox, CheckedListBox і ComboBox

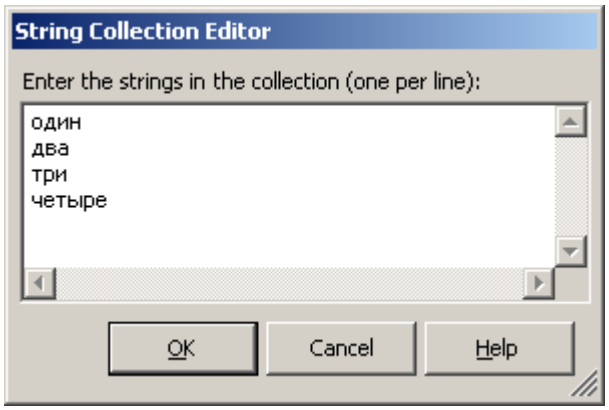
Елементи ListBox, CheckedListBox і ComboBox надають користувачеві списки вибраних ним альтернатив.

Елемент управління **ListBox**, що є списком, займає у формі область певного розміру. Пункти цього списку (поодинці або групами) вибираються за допомогою миші. Якщо вміст списку більший, ніж може відображуватися в елементі управління, поряд з ним з'являється вертикальна смуга прокрутки.

Вміст списку ListBox формується або за допомогою програмної коди або у вікні властивостей **Properties.**

Щоб заповнити список під час проектування, виберіть властивість Items у вікні властивостей елементу і клацніть на кнопці з багатокрапкою. Відкриється нове вікно редактора рядків, **String Collection Editor,** в якому можна додати в список необхідні пункти. Кожен пункт слід вводити окремим рядком, тому порожні рядки також можуть з'явитися в списку.





Нові пункти відображуються в списку при завантаженні форми, але з коду додатка у будь-який момент можна додати в список нові пункти (або видалити існуючі).

Елемент управління **ComboBox** також є списком, але на відміну від елементу **ListBox** зазвичай займає на екрані значно менше місця. Користувач може розширити список елементу **ComboBox**, вибрати потрібний пункт, а потім звернути список

Елемент управління **CheckedListBox** — це різновид елементу **ListBox**. Відрізняються ці елементи лише тим, що перед кожним пунктом списку елементу **CheckedListBox** відображується ще і прапорець. Щоб вибрати пункт в списку, користувачеві досить відзначити або скинути відповідний прапорець.

Vb6 і VB/NET

-

Сімейство Items

Для доступу до окремих пунктів списку слід звернутися до властивості **Items**, яка є сімейством. Перший об'єкт сімейства — **Items** (0), другий, — **Items (1)** і так далі.

Сімейство Items - це список елементу управління . Члени цього сімейства можна використовувати для доступу до окремих пунктів списку, а також для їх додавання або видалення.

Методи сімейства Items

Метод Add - Щоб додати новий пункт в список, використовуйте методи Items.Add і Items.Insert. Синтаксис методу Add такий:

```
ListBoxl .Items. Add(Item)
```

Параметр Item є об'єктом, що додається до списку. Якщо свойство Sorted не рівне True, метод Add додає нові пункти в кінець списку.

Метод Clear - видаляє всі пункти з елементу управління. Його синтаксис:

ListBoxl.Items.Clear

Властивість Count задає кількість пунктів в списку. Звернутися до всіх пунктів списку можна, якщо змінна циклу For. Next змінюється від 0 до ListBoxl.Items.Count - 1, що показане на прикладі методу Add.

Або

```
Dim itm As Object
For Each itm In ListBox1.Items
{ обробка поточного пункту представленого змінною itm }
Next
```

Метод CopyTo витягує всі пункти з елементу управління ListBox і зберігає їх в масиві, який передається методу як аргумент. Метод Copyто має наступний синтаксис:

```
ListBox1. CopyTo(destination, index)
```

Тут `destination` — це ім'я масиву, де зберігається список, а `index` — це індекс елементу в масиві, де зберігається перший пункт списку. Масив, який міститиме пункти елементу управління, необхідно оголосити явно і зробити достаточно великим, щоб можна було зберегти весь список.

Метод Insert поміщає новий пункт в конкретну позицію списку, синтаксис якого представлений нижчим:

```
ListBox1.Items.Insert(index, item)
```

Тут `item` — об'єкт, що додається, а `index` — позиція нового пункту. Номер первого пункту в списку — 0. Не слід вставляти пункти строго певні позиции відсортованого списку. Це порушить порядок сортування.

Метод Remove - Для видалення пункту із списку потрібно спочатку знайти його позицію (параметр index в списку, а потім викликати метод Remove, передавав індекс пункту як аргумент

```
ListBoxl.Items.Remove(index)
```

Обов'язковий параметр index — це номер пункту, що видаляється. Наступного оператора видаляє перший пункт списку:

```
ListBox1.Remove(O)
```

Вказати пункт, що видаляється, можна також по посиланню. Щоб видалити конкретний пункт із списку, використовуйте такий синтаксис:

```
ListBoxl.Items.Remove(item)
```

Метод Contains приймає як аргумент об'єкт і повертає значення True/False, яке вказує, містить сімейство заданий об'єкт чи ні Використання методу **Contains** допомагає уникнути вставки однакових об'єктів в список ListBox. Слід. оператори додають до сімейства Items новий рядок лише в тому разі, її такого рядка в сімействі ще немає:

```
    Dim itm As String = "Remote Computing"
If Not ListBoxl.Items.Contains(itm) Then
    ListBoxl.Items.Add(itm)
 End If
```

Виділення пунктів списку

Залежно від значення властивості **SelectionMode** користувач може вибирати в списку **ListBox** одночасні один або декілька пунктів. Якщо елемент **ListBox** допускає вибір лише одного пункту, дістати доступ до вказаного пункту можно за допомогою властивості **Selectedltem**, а до його індексу — за допомогою властивості **Selectedlndex**. Властивість Sele**ct**edltem повертає виділений пункт списку. Цей пункт може бути об'єктом. Рядок, відмічений користувачем клацанням миші, определяется властивістю Text.

Якщо елемент управління допускає вибір декількох пунктів, вони передаються за допомогою властивості **Selected** Items. Ця властивість є сімейством об'єктів **Item** і містить ті ж члени, що і сімейство **Items**. Властивість **Selectedltems**. **Count** возвращает кількість вибраних пунктів.

Для виводу всіх виділених в списку **ListBox** пунктів використовується ось такий цикл

```
    Dim itm As Object
For Each itm In ListBoxl.Selectedltems
    Console.WriteLine(itm)
Next
```

методи	**призначення**	
Add	для додавання в сімействі нових об'єктів	ListBoxl .Items. Add(Item) Параметр Item є об'єктом, що додається до списку
Remove	для видалення об'єктів по індексу та значенню	ListBoxl.Items.Remove(index) ListBoxl.Items.Remove(item)
Insert	поміщає новий пункт в конкретну позицію списку	ListBox1.Items.Insert(index, item) item — об'єкт, що додається, index — позиція нового пункту. Номер першого пункту в списку — 0
Clear	видаляє всі пункти з елементу управління	ListBoxl.Items.Clear

Contains	приймає як аргумент об'єкт і повертає значення True/False, яке вказує, містить сімейство заданий об'єкт чи ні	Dim itm As String = "Remote Computing"
		If Not ListBoxl.Items.Contains(itm) Then
		ListBoxl.Items.Add(itm)
		End If

SelectedIndex	отримати індекс вибраного пункту	
SelectedItem	отримати значення вибраного пункту	Для виводу всіх виділених в списку ListBox пунктів використовується ось такий цикл Dim itm As Object For Each itm In ListBoxl.SelectedItems Console.WriteLine(itm) Next
SelectedIndices	здобуття індексів вибраних пунктів	
SelectedItems	здобуття значень вибраних пунктів	SelectedItems.Count повертає кількість вибраних пунктів.
FindString	відшукує в списку пункти, які по написанню щонайближче до заданому зразку пошуку	
FindStringExact	знаходить ті пункти, які точно збігаються із заданим зразком	
Властивості сімейства		
Count	кількість об'єктів в сімействі	
SelectionMode	визначає спосіб вибірки користувачем пунктів списку None, One, MultiSimple, MultiExtended	
Sorted	True - відсортований	

Розширена також можливість обробки безлічі вибраних пунктів. Якщо в списку вибирається лише один пункт, то отримати індекс і значення вибраного пункту можна за допомогою властивостей **SelectedIndex** і **SelectedItem** елементу управління. Якщо ж допускається вибір декількох пунктів, то для здобуття індексів і значень вибраних пунктів використовуються сімейства **SelectedIndices** і **SelectedItems**.

Найбільш важлива новина, реалізована в елементі управління **ListBox**, - це функція пошуку. Тепер для знаходження того або іншого пункту в списку можуть бути використані методи **FindString** і **FindStringExact**. Метод **FindString** відшукує в списку пункти, які по написанню щонайближче до заданому зразку пошуку, а метод **FindStringExact** знаходить ті пункти, які точно збігаються із заданим зразком (якщо такі взагалі є). ці методи однаково добре працюють як з відсортованими, так і з невідсортованими списками.

Нарешті, введено два нові методи для підвищення швидкості відображення списку при додаванні до елементу нових пунктів. Якщо вам необхідно відразу додати до елементу безліч пунктів, спочатку звернетеся до методу **BeginUpdate**, потім викличте метод **Add** або **Insert** сімейства **Items** стільки раз, скільки потрібний, і лише після цього скористайтеся методом **EndUpdate**. Елемент управління не почне оновлюватися при кожному додаванні нового пункту, а список буде оновлений відразу після виконання методу **EndUpdate**. Ця методика дозволяє уникнути безперервного мерехтіння елементу управління при додаванні в список нових пунктів.

-

Основні властивості

IntegralHeight - має типа **Boolean** (значення True/False) і вказує, чи змінюватиметься висота елементу управління, щоб в списку відображувалося ціле число рядків

SelectionMode - визначає спосіб вибірки користувачем пунктів списку. Ця властивість встановлюється під час проектування .Путем завдання різних значенні властивості можна дозволити (або заборонити) користувачеві вибирати в списку декілька пунктів одночасно, а також вказати спосіб, який використовуватиметься для такого вибору. Можливі значення даної властивості

Значення	**Опис**
None	Вибір значень заборонений
One	Може бути вибраний лише один пункт.. (Значення за умовчанням)
MultiSimple	Простий вибір декількох пунктів. За допомогою клацання мишею (або натиснення клавіші пропуску) можна вибрати пункт або відмінити його вибір в списку
MultiExtended	Розширений вибір декількох пунктів. Для вибору декількох пунктів підряд натискуйте клавішу [Shift] і клацайте кнопкою миші (або натискуйте одну з клавіш управління курсором). Всі пункти між раніше вибраним і поточним пунктами будуть виділені. Щоб вибрати в списку окремі пункти (або відмінити вибір), натискуйте клавішу [Ctrl] і клацніть на потрібному пункті мишею

Sorted - Якщо ви хочете, щоб список завжди був відсортований, привласніть властивості Sorted значення True. Цю властивість можна встановлювати як під час розробки, так і під час виконання додатка. Елемент LstBox є текстовим **і** орієнтований на обробку тексту, тому він не забезпечує коректне сортування числових даних

Text

Властивість **Text** повертає виділений в елементі управління текст. Звернете увагу, що пункти в списку не обов'язково є рядками. За умовчанням кожним пунктом списку є окремий об'єкт і кожному об'єкту відповідає окремий рядок. Цей рядок і повертає метод **ToString** об'єкту. Для доступу до виділеного рядка елементу управління

використовується властивість **Text**, а для доступу до об'єкту (пункту списку) — властивість **Selectedlten.**

Елемент управління ComboBox

Подібно до елементу ListBox, елемент управління ComboBox містить безліч доступних для вибору пунктів, але зазвичай займає на екрані менше місця. Елемент ComboBox фактично є розширюваним елементом управління ListBox який розвертається, коли користувач хоче зробити вибір, і згортається після того, як вибір зроблений. Як правило, в полі елементу управління ComboBox.отображается один рядок з вибраним пунктом. Відмінність між елементу ComboBox і LstBox полягає в тому, що ComboBox дозволяє користувачеві задавати нові пункти, яких немає в списку.

У Visual Basic .NET доступні елементи управління ComboBox трьох типів. Значення властивості DropDownStyle:

DropDown	Елемент містить список, що розкривається, і текстове поле. Користувач може вибирати пункти із списку або вводити нові пункти в текстове поле (значення за умовчанням)
DropDownList	Складається із списку, що розкривається, де користувач може вибрати один з пунктів, але не може ввести новий пункт
Simple	Включає текстове поле і не містить списку, що розкривається, Користувач може вибирати пункти із списку за допомогою клавиш із стрілками або вводити їх в текстове поле

Новий рядок, що вводиться в текстове поле елементу Управління, не стає черговим пунктом списку. Рядок лише відображується в полі до тих пір, поки користувач не вибере інший пункт або не видалить її.

Під час виконання додатка елемент ComboBox дозволяє користувачеві вводить текст в полі редагування, але не передбачає простого і зручного способа для додавання в список нових пунктів

Додавання нового пункту в елемент ComboBox під час виконання

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
    Dim itm As String
    itm = InputBox("Enter new item", "New item")
    If itm <> "" Then AddElement(itm)
End Sub
```

Далі розглянемо код підпрограми AddElement (), аргументом которой є рядок, що додається в список елементу управління

Як ви увидите пізніше, цю ж підпрограму можна використовувати для додавання пунктів до елементу управління

Підпрограма **AddElement**()

```
Sub AddElement(ByVal newItem As String)
    Dim idx As Integer
```

```
    If Not ComboBox1.Items.Contains(newItem) Then
       idx = ComboBox1.Items.Add(newItem)
    Else
       idx = ComboBox1.Items.IndexOf(newItem)
    End If
    ComboBox1.SelectedIndex = idx
  End Sub
```

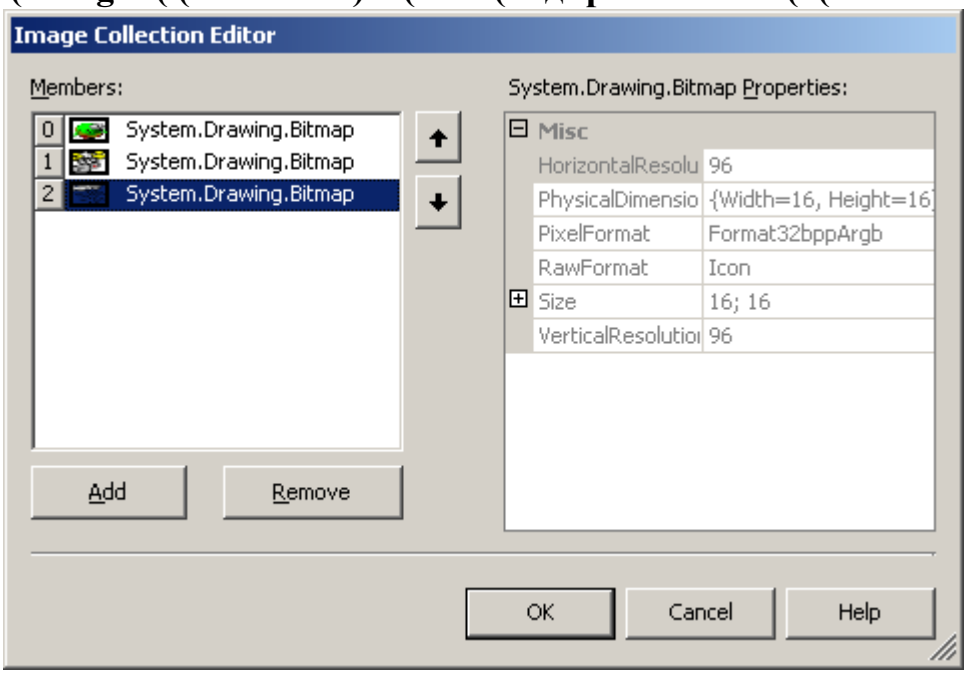
Нові пункти можна також додавати під час виконання, якщо в обробник події LostFocus (втрата фокусу) елементу управління помістити наступний код:

```
Private Sub ComboBox1_LostFocus(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox1.LostFocus
        Dim newItem As String = ComboBox1.Text
        AddElement(newItem)
    End Sub
```

Елемент управління ToolBar(панель інструментів)

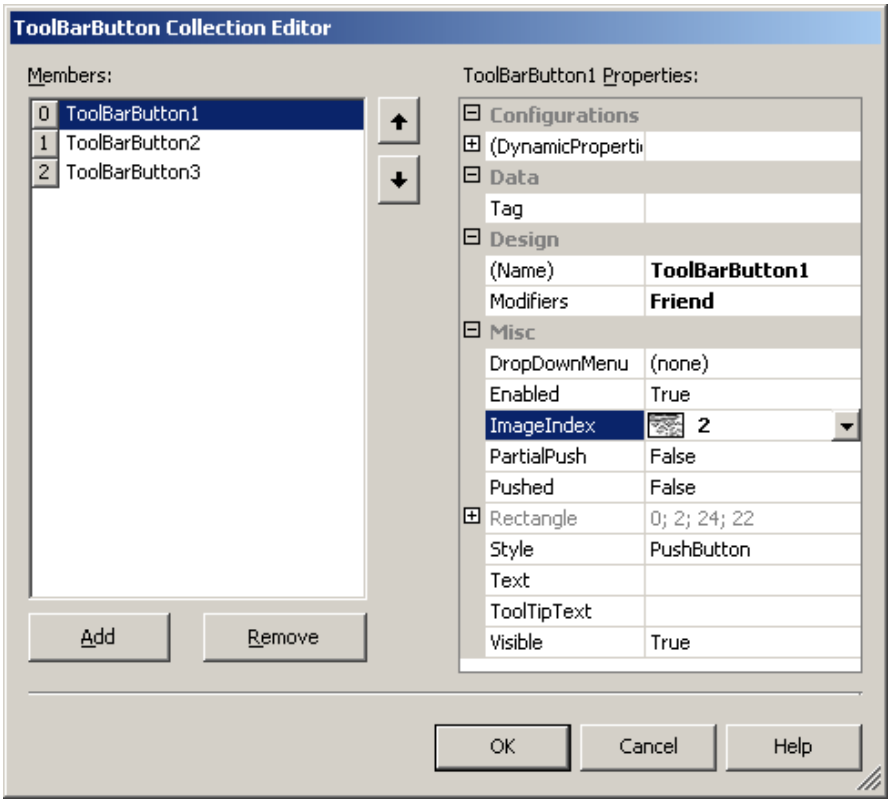
1. ToolBar1

2. ImageList1 (Images ((Collection) . (Add(відкрити *.ICO (.(OK.



3. **ToolBar1 (ImageList (ImageList1**
4. **ToolBar1 (Buttons ((Collection) . (Add (ImageIndex (№ ікони (Text (підпис кнопки (OK.**
5.
```
Private Sub ToolBar1_ButtonClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.ToolBarButtonClickEventArgs) Handles
ToolBar1.ButtonClick
  Select Case ToolBar1.Buttons.IndexOf(e.Button)
    Case 0
      MessageBox.Show("First toolbar button clicked")
      OpenFileDialog1.ShowDialog()
```

```
        t = OpenFileDialog1.FileName()
        MessageBox.Show("вибраний файл " & t)
    Case 1
        MessageBox.Show("Second toolbar button clicked")
        SaveFileDialog1.ShowDialog()
    Case 2
        MessageBox.Show("Third toolbar button clicked")
        PrintDialog1.ShowDialog()
End Select
 End Sub
```



Елементи управління ScrollBar| і TrackBar|

Елементи управління Scrol1Bar і TrackBar дозволяють користувачеві встановлювати значення деякої величини шляхом переміщення повзунка (або бігунка) між мінімальним і максимальним значеннями.

Елемент управління TrackBar (задаючий переміщення *повзунка)* подібний до елементу ScrollBar (у якому переміщуваним елементом є *бігунок),* але призначений для роботи з дискретними значеннями. Елемент TrackBaã має фіксована кількість відміток (ділень) з написами розробника, наприклад Off, S1ow і Speedy (мал. 6.12). В даному випадку користувач може вибрати лише одне з декількох допустимих значень. Робота ж елементу ScrollBar заснована на візуальному зворотному зв'язку, що допомагає

встановити бігунок в потрібне положення. Елемент Scroll Bar застосовується тоді, коли точність значення не важлива. А елемент управління TrackBar служить для введення точних числових значень, які знаходяться в певному діапазоні

Елемент управління ScrollBar

Елемент Scrol1Bar — це смуга з покажчиком (бігунком), яка дозволяє користувачеві вибирати значення усередині заданого діапазону. Залежно від значення властивості **Orientation** смуга може розташовуватися або вертикально, або горизонтально. Лівий (або нижний) кінець елементу відповідає мінімальному значенню, правий (або верхній) — максимальному.

Основні властивості елементу ScrollBar

Minimum -Минимальное значення елементу управління. Значення за умовчанням дорівнює 0, але оскільки це ціла величина типа Integer, для неї можна встановлювати також негативні значення.

Maximum -Максимальное значення елементу управління. Значення за умовчанням дорівнює 100, але допускається встановити будь-яке значення, яке можна представити типом даних Integer.

Value - Поточне значення елементу, визначуване положенням бігунка

Властивості Minimum і Maximum— цілі позитивні величини типа Integer. Для позначення негативних або дробових чисел слід ввести код, отображающий фактичні значення у вигляді цілих чисел.

Події елементу управління ScrollBar

Користувач може змінити значення елементу **ScrollÂàã** трьома способами:

- Шляхом виконання клацань на двох кінцевих стрілках. Значення елементу управління змінюється на величину, визначувану властивістю **SmallChange**.

-Шляхом виконання клацань в області, що знаходиться між бігунком і стрілками. В цьому випадку значення елементу управління змінюється на величину, визначувану властивістю **LargeChange**.

- Шляхом перетягання бігунка за допомогою миші.

Зміна значення елементу **ScrollBar** з коди додатка контролюється за допомогою двох подій: **ValueChanged** і **Scroll**. Обидві події відбуваються, коли користувач міняє положення бігунка. При зміні значення елементу управління з коди виконується лише подія **ValueChanged**. Подія **Scroll** відбувається безперервно при переміщенні бігунка і у відповідь на багато інших дій, наприклад клацання на кінцевих стрілках. Щоб додаток реагував на дії користувача, можна перевірити властивість **Type** другого аргументу обробника події. Настановні параметри властивості **e.Type** представлені в таблиці.

EndScroll	Користувач припинив переміщення бігунка
First	Бігунок переміщений в положення Minimum
LargeDecrement	Бігунок переміщається з великим декрементом (користувач клацав на смузі прокрутки між покажчиком і лівою стрілкою) .
Largelncrement	Бігунок переміщається з великим інкрементом (користувач клацав на смузі прокрутки між покажчиком і правою стрілкою)

Last	Бігунок переміщений в положення Maximum
SmallDecrement	Бігунок переміщається з малим декрементом (користувач клацав на лівій стрілці) :
Smalllncrement	Бігунок переміщається з малим інкрементом (користувач клацав на правій стрілці)
ThumbPosition	Бігунок переміщений
ThumbTrack	Бігунок переміщається