MIHICTEPCTBO ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

ФАКУЛЬТЕТ МІЖНАРОДНИХ ВІДНОСИН

**Роман Вовк**

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

**ДО ПРОВЕДЕННЯ ЗАНЯТЬ З КУРСУ**

**ПРИКЛАДНА ІНФОРМАТИКА**

**для студентів спеціальності**

**Міжнародна інформація**

Львів 2009

Методичні рекомендації для студентів факультету міжнародних відносин Львівського національного університету імені Івана Франка

Укладач

доцент кафедри міжнародних відносин і дипломатичної служби, кандидат фізико-математичних наук

Вовк Роман Володимирович

Відповідальний за випуск

завідувач кафедри міжнародних відносин і дипломатичної служби, доктор економічних наук, професор

Мальський Маркіян Зіновійович

**I. МЕТА I ЗАВДАННЯ КУРСУ**

У нормативній дисципліні "Прикладна інформатика", викладаються концептуальні основи алгоритмізації процесів обробки інформації з елементами програмування з використанням мови програмування Visual Basic та сучасні методи і засоби створення Web-сторінок.

Основними цілями курсу є:

1) формування сучасного наукового (інформаційного) світогляду;

2) підготовка студентів до праці в умо­вах інформатизації практично всіх областей професійної діяльності й освоєння засобів комп'ютерно-інформаційних і телекомунікацій­них технологій.

Завдання курсу:

а) формування вміння формального представлення обчислювальних процесів, процесів прийняття рішень;

б) формування практичних навичок побудови алгоритмів та їх практичної реалізації засобами Visual Basic;

в) формування уявлень студентів про функціонування міжнародної інформаційної мережі Інтернет;

г) формування практичних навичок створення інтерактивних документів та їх публікації в мережі Інтернет.

**2. ЗМІСТ ДИСЦИПЛІНИ**

**Тема 1. АЛГОРИТМИ, АЛГОРИТМІЗАЦІЯ, ПРЕДСТАВЛЕННЯ АЛГОРИТМІВ (2 год.)**

Визначення алгоритму, властивості алгоритмів: дискретність, визначеність, результативність, масовість. Форми запису алгоритмів: текстова, формульна, блок-схеми, алгоритмічні мови. Базові алгоритмічні структури: лінійна, розгалужена, циклічна. Алгоритмічна мова та її компоненти. Базові поняття, що використовуються в алгоритмічних мовах: імена, константи, змінні, операції, вирази, дані, оператори, функції.

**Тема 2. СЕРЕДОВИЩЕ РОЗРОБКИ ПРОЕКТІВ MICROSOFT VISUAL STUDIO NET (2 год.)**

Робоча область Visual Studio. Створення і відкриття проектів. Візуальні компоненти. Об’єкти і класи. Властивості об’єктів. Події, процедури обробки подій. Засоби від лагодження програм. Змінні, спостереження за значеннями змінних. Арифметичні вирази. Оператори. Логічні оператори. Типи змінних. Опис змінних.

**Тема 3. ОСНОВНІ ЕЛЕМЕНТИ УПРАВЛІННЯ (2 год.)**

Спільні властивості об’єктів. Визначення властивостей. Елемент TextBox і Label та їх застосування для вводу і виводу інформації. Створення елементів управління типу Button та обробка подій, пов’язаних з ними. Елементи управління вибору: CheckBox, RadioButton. Елементи списків ListBox, ComboBox.

**Тема 4. ПРОГРАМУВАННЯ ЛІНІЙНИХ ТА РОЗГАЛУЖЕНИХ АЛГОРИТМІВ (4 год.)**

Лінійні та нелінійні алгоритми. Умовні ба безумовні розгалуження. Оператори розгалуження: If, Select Case. Обчислювальні процеси з повторенням. Оператори циклів For, Do. Приклади застосування циклічних процесів для обробки даних у списках. Масиви. Опис масивів. Динамічні масиви. Методи знаходження найбільшого і найменшого значення масиву, суми, середнього значення, сортування масивів.

**Тема 5. ДІАЛОГОВІ ВІКНА (2 год.)**

Використання вікон повідомлень. Діалогові вікна. Використання функцій InputBox і MsgBox. Управління виглядом форми. Створення підпорядкованих форм.. Використання графічних компонент для оформлення форм. Графічні методи. Обмін даними між формами. Створення власних форм вводу і виводу інформації.

**Тема 6. ДОДАТКОВІ ЕЛЕМЕНТИ УПРАВЛІННЯ (2 год.)**

Автоматичне встановлення розмірів і положення елементів управління. Фіксація елементів управління. Спеціалізовані елементи управління для вводу даних: DateTimePicker, MonthCalendar. Елементи Timer, LinkLabel, ProgressBar. Використання програмного коду для управління об’єктами. Обробка подій та методів додаткових елементів управління.

**Тема 7. ПРОГРАМУВАННЯ ІНТЕРАКТИВНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА (4 год.)**

Створення меню. Обробка подій, викликаних командами меню. Створення та використання панелей інструментів. Використання стандартних елементів управління, створених незалежними виробниками. Взаємодія із зовнішніми об’єктами. Динамічні зв’язки. Робота із файлами. Збереження даних.

**Тема 8. WEB-ДОДАТКИ І WEB-СЛУЖБИ. (2 год.)**

Створення Web-форм. Серверні елементи управління. Обмін даними. Програми сценаріїв. Управління проектами Web-додатків. Програмування взаємодії з клієнтом. Публікація проектів.

**Тема 9. ВЗАЄМОДІЯ VISUAL BASIC ІЗ БАЗАМИ ДАНИХ (4 год.)**

Конструювання баз даних. Запити і зв’язки. Основи мови запитів SQL. Використання запитів SQL Access. Використання виразів SELECT. Вибір списків полів. Фільтрування записів. Зв’язування таблиць. Обробка баз даних засобами Visual Basic. Використання об’єктів ADO. Набори даних DataSet.

**Тема 10. ЗАСТОСУВАННЯ VISUAL BASIC FOR APPLICATION (4 год.)**

Особливості застосування операторів та функцій у VBA. Використання стандартних функцій. Взаємодія із змінними та константами зовнішнього середовища. Створення та використання макросів Microsoft Word. Автоматизація обробки табличних даних Microsoft Excel. Застосування програмного коду для створення нових команд меню та панелей інструментів.

**Тема 11. ОСНОВИ МОВИ HTML (2 год.)**

Загальні поняття, синтаксис. Структура HTML документів. Форматування документів. Списки. Таблиці. Фрейми. Форми. Створення гіперпосилань. Вставка мультимедійних об'єктів. Ключові слова. Регіональне налаштування відображення документів. Навігаційні карти посилань.

**Тема 12. СТВОРЕННЯ WEB-СТОРІНОК ЗА ДОПОМОГОЮ FRONT PAGE ТА MACROMEDIA DREAMWEAVER (2 год.)**

Створення документів. Шаблони документів. Основні параметри сторінки. Ввід і форматування тексту. Таблиці. Використання об'єктів. Форми. Створення меню. Фреймова структура документів. Публікація Web-сторінок.

**Тема 13. FLASH-ТЕХНОЛОГІЇ (2 год.)**

Основні принципи створення документів Flash. Середовище програми Flash. Порядок відображення кадрів. Використання об’єктів та їх взаємодія. Управління шарами. Анімаційні ефекти. Сцени. Символи. Елементи управління. Включення Flash-документів у Web-сторінки.

**3. НАВЧАЛЬНО-МЕТОДИЧНІ МАТЕРІАЛИ**

1. Баженов В.А., Венгерський П.С., Горлач О.М. та ін. Інформатика. Комп'ютерна техніка. Комп'ютерні технології. Підручник для студентів вищих закладів освіти. - К.: Каравела, 2003. - 464 с.
2. Браун С. Visual Basic. Учебный курс. Питер. 1999. - 688с.
3. Волчёнков Н.Г. Программирование на Visual Basic 6: В 3-х ч. - М.: ИНФРА-М, 2000.
4. Гончаров И. Самоучитель HTML. Питер 2000. – 240 с.
5. Дибкова Л.М. Інформатика та комп’ютерна техніка: Посібник для студ.вищих навч.закладів. – К.: "Академвидав", 2002.- 320 с. (Альма-матер)
6. Долженков В., Мозговой М. Visual Basic NET. Учебный курс. - Питер. 2003. – 464 с.
7. Ивьен Б., Берес Дж. Visual Basic NET. Библия пользователя. – Диалектика. 2002. – 1024 с.
8. Мак-Манус Дж., Голдштейн Дж., Прайс К. Обработка баз данных на Visual Basic NET. – Вильямс. – 416 с.
9. Мэтьюз М., Полсен Э. Front Page 2000: полное руководство. – К. 2003. – 480 с.
10. Негус К. Internet Explorer 4. Библия пользователя. - К., М., СПб.: Диалектика, 1998.
11. Новиков Ф.А., Яценко А.Д. Microsoft Office XP в целом. - СПб.: БХВ-Петербург, 2002.
12. Пауерс Л. Microsoft Visual Studio 2008. – СПб. 2009. – 1200 с.
13. Пери П.Дж. Секрети World Wide Web. К. Диалектика, 1997. – 540 с.
14. Хестер Н. Front Page 2002 для Windows
15. Halvorson M. Visual Basic 2008. Step by step. Microsoft Press. 2008. – 545 p.

**4. ЛАБОРАТОРНІ РОБОТИ**

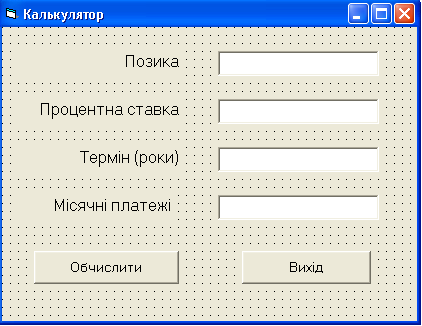
**Лабораторна робота 1**

**Форми та основні елементи управління**

1. Створити новий проект. Для цього у діалоговому вікні New Project створити проект Standard EXE.

*Проект – набір файлів, в яких міститься інформація про компоненти програми.*

2. У створеній формі вставити елементи управління TextBox, Label і Command Button за такою схемою

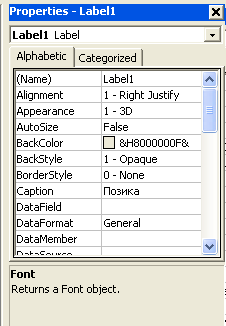


*Елемент управління (Control) - об’єкт, що міститься у формі, за допомогою якогоздійснюється взаємодія з користувачем або програмами.*

***TextBox*** *– для вводу і відображення текстової інформації.*

***Label*** *– для відображення різних відомостей інформативного характеру (написів, заголовків, назв полів ... ).*

***Command Button*** *– для ініціювання різних дій програми після натискування кнопки мишки.*



Для встановлення властивостей об’єктів, що містяться у формі, використовують вікно Properties. Внести зміни у властивості:

***Name*** *– назва об’єкта*

***Alignment*** *– вирівнювання*

***Caption*** *– заголовок*

***Font*** *– шрифт (гарнітура, розмір, написання)*

***Text*** *– значення, яке буде відображатися у текстовому полі.*

3. Зберегти проект. Змінити стандартні назви форми та проекту на свої.

4. Написати процедури обробки подій. Процедури містяться у вікні Code, що відноситься до активного проекту.

У процедуру, що відповідає кнопці **Вихід,** вставити команду End, яка здійснить вихід із програми при натискування кнопки мишки.

Private Sub Command2\_Click()

End

End Sub

Для проведення розрахунків потрібно написати програму (програмний код). Перед використанням змінні потрібно оголосити, тобто вказати їх назву та тип даних, що в них зберігатиметься. Для цього використовується оператор Dim, який у процедурі іде після назви процедури.

Private Sub Command1\_Click()

Dim dpozyka, dplata As Double

Dim fproc As Single, nterm As Integer

End Sub

*У Visual Basic прийнято у назвах змінних використовувати префікси, які б вказували на ти змінної. Таким чином у тексті програми легше буде визначати тип змінних. Найчастіше використовуються такі змінні і відповідні їм префікси:*

|  |  |  |
| --- | --- | --- |
| Префікс | Тип змінної | Опис |
| S | String | Текстовий |
| N | Integer | Цілий, одинарної точності |
| L | Long Integer | Цілий, подвійної точності |
| F | Single | З плаваючою крапкою, одинарної точності |
| D | Double | З плаваючою крапкою, подвійної точності |
| C | Currency | Грошовий, для фінансових розрахунків |
| B | Boolean | Логічний |
| V | Variant | Варіантний |

5. Описаним змінним присвоїти значення, яке вводитиметься в елементах управління типу TextBox записати формулу для обчислення місячних платежів. Остаточно код матиме такий вигляд.

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim dpozyka, dplata As Double

Dim fproc As Single, nterm As Integer

dpozyka = Val(TextBox1.Text)

fproc = Val(TextBox2.Text) / 100

' Річну процентну ставку перетворюємо у місячну

fproc = fproc / 12

' Знаходимо кількість місяців

nterm = Val(TextBox3.Text) \* 12

' Обчислюємо місячні платежі

dplata = dpozyka \* (fproc / (1 - (1 + fproc) ^ -nterm))

TextBox4.Text = Format(dplata, "Fixed")

End Sub

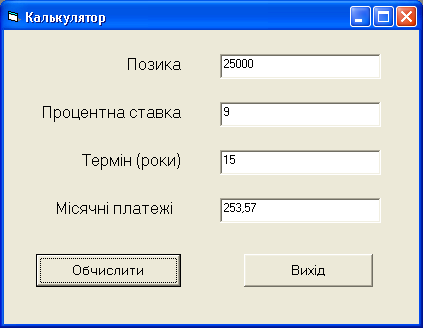
*Після апострофа міститься коментар, який не впливає на хід програми. У тексті він відображається зеленим кольором.*

6. Запуск програми на виконання. Для цього можна виконати одну із дій:

- клацнути по кнопці Start на панелі інструментів;

- вибрати із меню Run команду Start;

- натиснути клавішу F5.



Якщо програма написана і виконується правильно, то при внесенні відповідних даних і натискування кнопки **<Обчислити>** буде отримано результат.

Написана програма визначає платежі, які слід вносити кожного місяця для погашення позики при заданих наперед величині позики, процентної ставки та терміну позики.

Завершити виконання програми натиснувши кнопку **<Вихід>.**

7. У меню Fіle виконати команду Make … для створення програмного модуля, який виконуватиметься поза середовищем Visual Basic.

8. Завершити програму Visual Basic.

**Лабораторна робота 2**

**Методи та події**

*Методи – це функції, вмонтовані в об’єкти, за допомогою яких можна виконувати деякі операції без написання додаткового коду.*

Наприклад, для форми існує метод PrintForm, який забезпечує вивід образу форми на стандартний принтер. Так, якщо потрібно видрукувати форму Form1, достатньо вказати оператор Form1.PrintForm. Різні об’єкти мають свої набори методів. Спільними для всіх об’єктів є наступні методи:

Drag – метод, який забезпечує підтримку операції перетягування в межах контейнера (форми, фрейма);

Move – дозволяє змінити положення об’єкту на екрані;

SetFocus – робить активним деякий об’єкт форми, тобто фокусує його. Після розміщення об’єкта в фокус, можна взаємодіяти з ним, наприклад вводити текст, натиснути кнопку, зробити відмітку тощо. Свідченням того, що даний об’єкт у фокусі є поява курсора у полі, виділення рамкою;

ZOrder – визначає, чи знаходиться об’єкт попереду чи позаду інших об’єктів контейнера.

Крім виконання основних функцій за допомогою методів, об’єкти можуть реагувати на деякі події, які виникають в результаті певних дій користувача. Реакція на події визначається програмним кодом, що міститься у спеціальних процедурах для їх обробки, так званих *процедурах обробки подій*. Наприклад, при натискуванні кнопки мишки виникає подія Click, яка поступає від від об’єкта типу CommandButton. Набір подій, які може викликати даний об’єкт у відповідь на дії користувача та стандартні процедури обробки цих подій визначаються на етапі створення об’єкта.

Методи, які виконують певні завдання, можуть впливати на деякі властивості об’єктів. Наприклад, для зміни положення об’єкта можна використовувати метод Move, при цьому значення властивостей Left і Top будуть змінюватися. Значення цих властивостей, та усіх інших, в яких використовуються відстані, задаються в твіпах. *Твіп* – універсальна, незалежна від типів пристроїв одиниця виміру, еквівалентна 1/20 стандартної щільності точок принтера. При стандартній роздільній здатності в одному дюймі міститься 72 точки принтера, тобто один дюйм рівняється 1440 твіпам. Зрозуміло, що реальний розмір об’єкта залежатиме від роздільної здатності екрану.

1. Створити форму за таким зразком із об’єктами типу Label та CommandButton.

**

2. Ввести наступний код у процедури обробки подій.

Private Sub Command1\_Click()

Form1.Width = Form1.Width + 100

Label5.Caption = Form1.Width

Label6.Caption = Form1.Height

Label7.Caption = Form1.Left

Label8.Caption = Form1.Top

Command1.Move 100, 200

End Sub

Private Sub Command2\_Click()

Form1.Height = Form1.Height + 100

Label5.Caption = Form1.Width

Label6.Caption = Form1.Height

Label7.Caption = Form1.Left

Label8.Caption = Form1.Top

Command2.Left = 100

Command2.Top = Form1.ScaleHeight - Command2.Height

End Sub

3. Зберегти та виконати проект. Змініть розміри форми і натискуючи на кнопки. Зверніть увагу на зміну значень властивостей форми.

При створенні простої програми, яка складається з однієї форми не потрібно слідкувати за процесом відображення чи приховування форми на екрані, усе відбувається автоматично. З початком виконання форма відображається, а після завершення – зникає. При цьому форма є стартовим об’єктом. У складніших програмах, коли проект містить декілька форм, необхідно у діалоговому вікні Project Properties вказати що буде стартовим об’єктом (Startup Object, Startup Form). В якості стартового об’єкта можна вибрати процедуру з іменем Main.

На стан форми впливають стандартні оператори Visual Basic Load і Unload та методи Show і Hide, передбачені для кожної форми. За допомогою команди Load форму можна завантажити в пам’ять, але на екрані вона не відображатиметься. Даний оператор завантажує форму явним чином. Неявно форма завантажується при звертанні до її властивостей, методів чи будь-якого об’єкту цієї форми. При завантаженні форми відбувається подія Form\_Load, до якої прикріплена своя процедура. Після завантаження за допомогою методу Show форма буде відображена. Якщо цей метод використати без попереднього завантаження форми, то вона спочатку завантажиться неявно, а потім відобразиться.

У методі Show передбачено необов’язковий аргумент vbModal, який визначає буде форма відображатися у модальному (modal) чи звичайному (modaless) режимі. При використанні модального режиму керування не повертається до процедури, що викликала метод Show, а програма перебуває у стані очікування до того моменту, коли буде закрита ця форма. У звичайному режимі користувач може перейти в інше діалогове вікно або іншу форму програми. Наприклад, відобразити форму Form2 у модальному режимі можна за командою

Form2.Show vbModal.

Забрати форму з екрана можна методом Hide.

4. Створити нову форму з довільним набором об’єктів. Серед цих об’єктів помістити кнопку для закриття форми (використати оператор End). В якості цієї форми можна використати форму Калькулятор із попередньої лабораторної роботи.

5. До першої форми додати командну кнопку, яка б викликала нову форму.

6. Завершити роботу з Visual Basic.

**Лабораторна робота 3**

**Розгалужені обчислювальні процеси**

Форми, як і більшість об’єктів у Visual Basic, мають набір властивостей, які визна­ча­ють їх зовнішній вигляд та поведінку. Багато із них можна змінювати в процесі виконання програми.

### Основні властивості форми

|  |  |  |
| --- | --- | --- |
| Властивості | Опис | Зміна при виконанні програми |
| BorderStyle | Визначає тип обрамлення для форми | Ні |
| ControlBox | Визначає, чи буде під час виконання програми у стрічці заголовку міститися кнопка виклику системного меню, в якому є дві команди – Move і Close | Ні |
| Font | Визначає тип шрифту, який використовується для відображення тексту у формі | Так |
| Icon | Визначає піктограму, яка буде знаходитися у стрічці заголовку форми та на панелі задач коли вікно форми буде згорнуто в піктограму | Так |
| MaxButton | Визначає, чи буде під час виконання програми активною кнопка максимізація, яка знаходиться у стрічці заголовку форми. | Ні |
| MDIChild | Визначає дочірню форму багатодокументної програми | Так |
| MinButton | Визначає, чи буде під час виконання програми активною кнопка мінімізація, яка знаходиться у стрічці заголовку форми. | Так |
| StartUpPosition | Визначає початкове положення форми на екрані при запуску програми | Ні |
| WindowState | Вказує вид форми при запуску програми (чи буде форма розгорнутою, згорнутою чи звичайною) | Так |

Властивість BorderStyle може приймати одне із шести значень, які визначають тип обрамлення відображуваної форми:

|  |  |
| --- | --- |
| 0 – None | Рамка кругом форми не відображається. Немає також стрічки заголовку і системних кнопок |
| 1 – Fixed Single | Навколо форми відображається рамка, що складається з одинарної лінії. У стрічці заголовку міститься кнопка виклику системного меню та кнопка закриття діалогового вікна. Розмір форми змінити не можна. |
| 2 – Sizable | Форма представляється в рамці, на якій мишка набирає вигляду двосторонньої стрілки. У цьому випадку можна змінити розмір форми. У стрічці заголовку відображаються кнопки системного меню та управління діалоговим вікном. Дане значення задається автоматично, по замовчуванню. |
| 3 – Fixed Dialog | Навколо форми відображається рамка, розмір змінити не можна. Стрічка заголовку містить кнопки виклику системного меню та закриття форми. |
| 4 – Fixed ToolWindow | Рамка навколо форми у вигляді одинарної лінії. У стрічці заголовку відображається тільки кнопка закриття вікна форми. Розміри форми не змінюються. |
| 5 – Sizable ToolWindow | Так як попереднє, але розміри можна змінювати. |

Після вибору значення властивості BorderStyle можна присвоїти значення True або False властивостям ControlBox, MaxButton або MinButton, що дозволить відображати чи не відображати відповідні кнопки у стрічці заголовку.

Властивість StartupPosition може мати такі значення:

|  |  |
| --- | --- |
| 0 – Manual | Початкове розміщення визначається значеннями властивостей форми Top і Left; |
| 1 – CenterOwner | Форма відображається у центрі робочого стола Windows. Якщо активна форма є дочірньою, то вона розміститься у центрі батьківської форми; |
| 2 – CenterScreen | Форма відображається у центрі робочого стола Windows. |
| 3 – Windows Default | Розміщення форми визначається системою Windows в залежності від кількості відкритих на даний момент вікон. |

Елементи управління для прийняття рішень

|  |  |  |
| --- | --- | --- |
| Назва |  | Призначення |
| CommandButton | Кнопка | Служить для виклику певних дій програми після натискування кнопки мишки |
| CheckBox | Прапорець | Дозволяє встановити один або декілька параметрів у положення Вкл./Викл. або надати логічним змінним значення True|False |
| OptionButton | Перемикач | Дозволяє вибрати один елемент із запропо­нованої групи. |
| ListBox | Список | Дозволяє вибрати елементи із наперед сформованого списку |
| ComboBox | Поле зі списком | Те саме, що і ListBox, додатково містить стрічку для вводу нового значення |

Об’єкт Frame служить контейнером для інших елементів управління, дозволяє згрупувати декілька елементів управління.

Перед використанням у процедурах змінні описують за допомогою оператора Dim. Такі змінні називаються локальними або внутрішніми і можуть використовуватися лише в межах цієї процедури, де вони описані. Однакові змінні можуть використовуватися у різних процедурах, при цьому зачення їх не впливатимуть одні на одних. Локальні змінні не можуть бути викороистані за межами процедури.

Для того, щоб можна було використовувати одні і ті ж змінні в різних процедурах, необхідно їх описати як глобальні. Такий опис здійснюється перед процедурами у вікні коду з використанням оператора Public.

Поряд із змінними, описаними користувачем, у програмному коді можуть використовуватися стандартні змінні Visual Basic. Такими змінними можуть бути, наприклад, значення стандартних кольорів – vbBlack, vbRed, vbBlue і т.д.

**Умовний оператор**

If <умова> Then

блок операторів, що виконуються у випадку істинності умови

Else

блок операторів, що виконуються тоді, коли умова не виконується

End If

В якості умови використовується логічний вираз або логічна змінна. Для заперечення значення логічного виразу використовується оператор Not. Умовний оператор допускає вкладення один в другий. Умовний оператор може використовуатися у скорочекому виді:

If <умова> Then <команда>

If <умова> Then

блок операторів, що виконуються у випадку істинності умови

End If.

# Оператор розгалуження

Select Case <змінна>

Case <вираз1>

фрагмент програми 1

Case <вираз2>

фрагмент програми 2

.

.

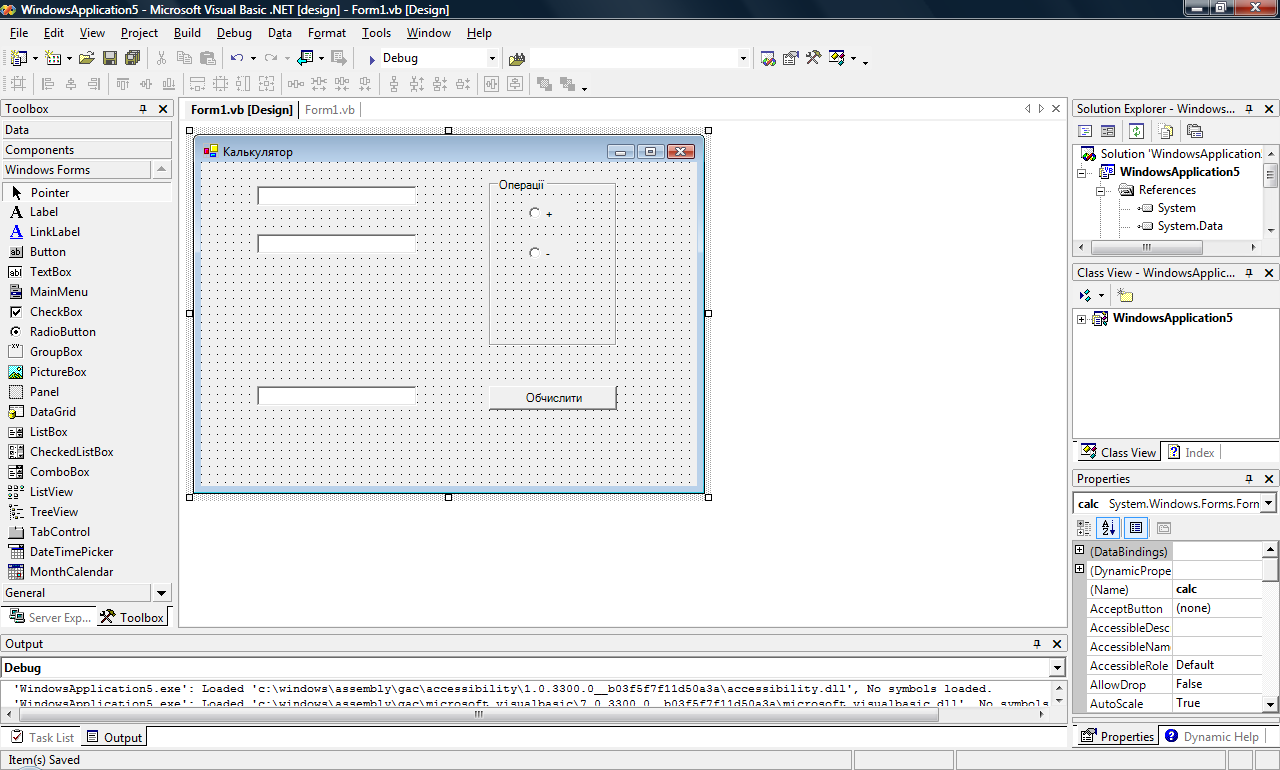
.

Case <виразN>

фрагмент програми N

End Select

1. Створити форму за зразком



2. До процедури обробки події Click кнопки додати програмний код

*TextBox3.Text = ""*

*If RadioButton1.Checked Then*

*TextBox3.Text = Val(TextBox1.Text)+Val(TextBox2.Text)*

*ElseIf RadioButton2.Checked Then*

*TextBox3.Text = Val(TextBox1.Text)-Val(TextBox2.Text)*

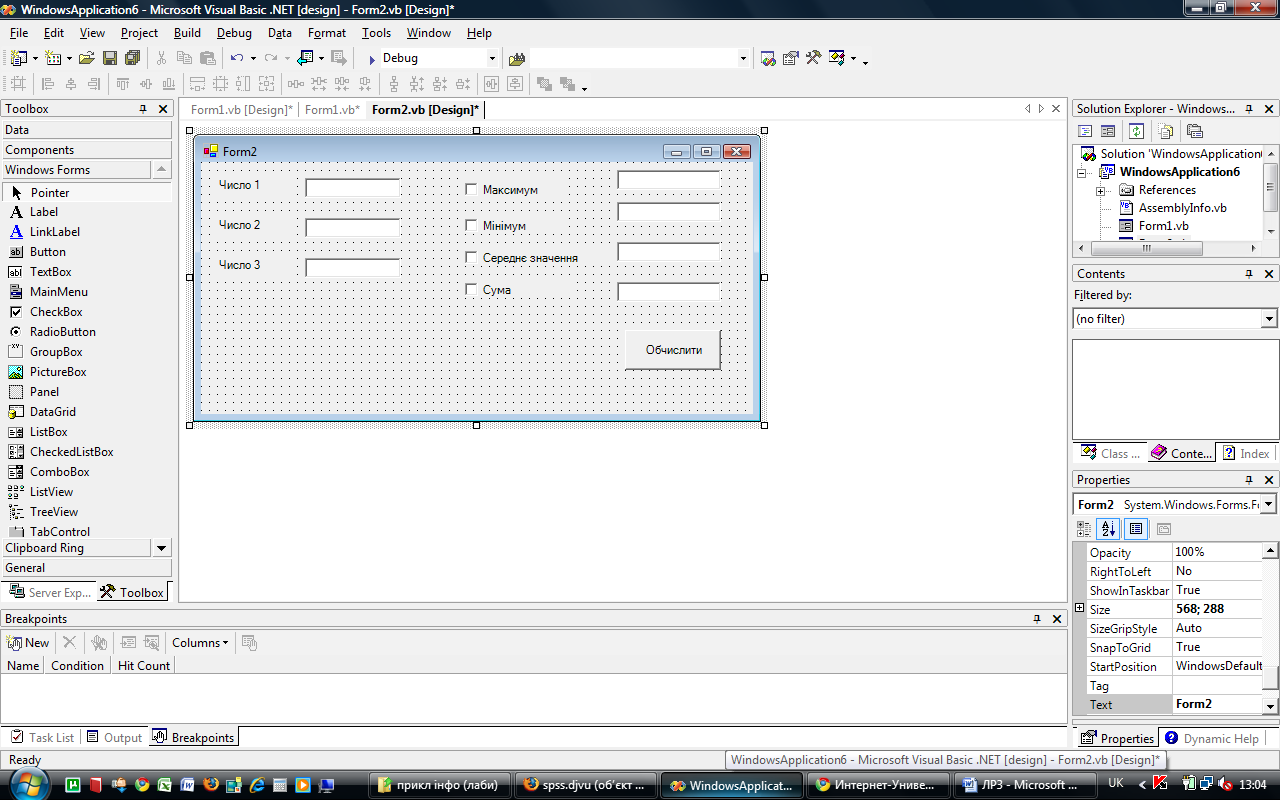
*Else*

*MsgBox("Виберіть операцію")*

*End If*

3. Змінити код для забезпечення виконання операцій множення і ділення.

4. Створити форму і забезпечити виконання вказаних функцій. При цьому використати умовні оператори та об’єкти типу Checkbox



5. Завершити роботу з Visual Basic.

**Лабораторна робота 4**

**Меню та панель інструментів**

Система багаторівневих меню у Visual Basic створюється за допомогою засобів Menu Editor, що викликається із меню Tools (див. рис.1). Стрілки "вліво", "вправо" служать для створення команд чи підменю.

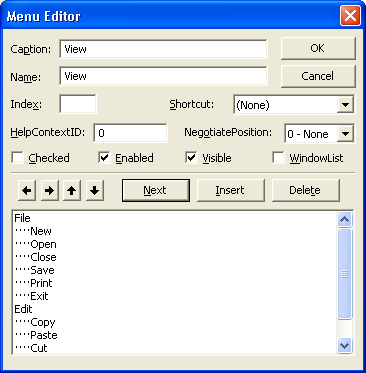


Рис.1 Діалогове вікно Menu Editor

Команди наступного рівня відображаються з префіксом із крапок. Стрілки "вниз" "вверх" служать для заміни порядку розміщення команд в меню. Для виклику кожної стрічки меню можна вказати комбінацію клавіш вибираючи її із списку Shortcut. Задана на рис.1 структура меню у формі відтворить меню, показане на рис.2.

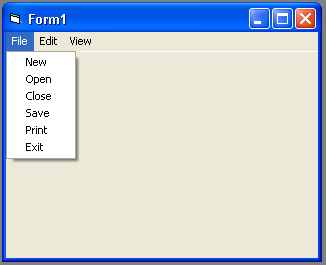


Рис.2 Меню

Створене таким чином меню буде розміщено в активній формі зверху. Для кожної команди із меню буде створено свій набір процедур обробки подій.

Внесення змін у створене раніше меню відбувається також з використанням Menu Editor.

Цілком аналогічно створюється контекстне меню, тільки для відповідних команд слід вказати значення властивості Visible рівним False, тобто галочка не буде поставлена. Виклик цього меню відбувається за допомогою метода PopUpMenu. Подія, яка викликатиме контекстне меню, як правило, активується натискуванням правої клавіші мишки.

Більшість сучасних програм, якщо не всі, містять панелі інструментів, які дозволяють при допомозі мишки виконувати ряд найчастіше вживаних команд на звертаючись до системи меню. Visual Basic дозволяє досить ефективно створювати такі панелі інструментів.

Доповнити набір об’єктів, що є доступними у середовищі Visual Basic можна виконуючи команду Components викликаючи контекстне меню на блоці інструментів (див.рис.3.). Для вибору необхідних блоків інструментів слід поставити відмітку. Ці інструменти потім відображаються у наборі елементів управління.

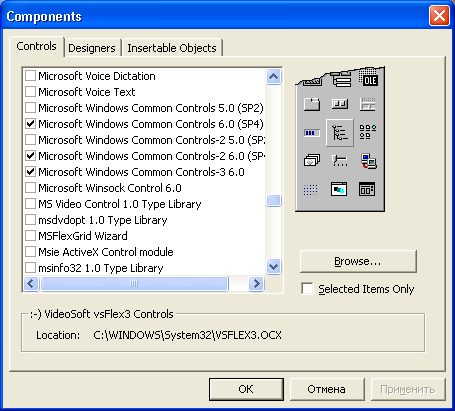


Рис.3 Перелік блоків елементів управління

Набір піктограм, малюнків, які будуть задіяні на панелі інструментів, буде доступним після поміщення у формі елементу управління ImageList. Використовуючи властивості цього елементу (діалогове вікно Property Pages) вибирають необхідні зображення кнопок панелі інструментів. Елемент управління ToolBar дозволяє автоматично створити панель інструментів в активній формі у верхній частині, ширина панелі відповідатиме ширині форми. Змінюючи властивості цього елементу можна розмістити панель інструментів біля будь-якої межі форми. При цьому слід зважати на розміри панелі інструментів. Дії, які відбуватимуться при натискування кнопок, задаються кодом процедур для обробки подій ButtonClick і ButtonMenuClick.

**Лабораторна робота 5**

**Програмування циклічних алгоритмів**

**Циклічні оператори**

**Цикл з лічильником**

For i=A To B Step C

...

Next i

Тут змінна i виконує функції лічильника, вирази A та B є відповідно початковим та кінцевим значеннями, які набуватиме змінна i. Вираз С визначає крок циклу.

**Цикли з умовою**

Do While <умова>

...

Loop

Даний цикл виконується доки умова є істинною. Сама умова перевіряється перед початком виконання оператоів, що є в циклі.

Do

…

Loop While <умова>

Такий оператор виконається хоч би один раз і повторюватиметься до тих пір, поки умова залишатиметься істинною.

Наступні оператори діють аналогічно до попередніх, а виконуватимуться у випадку хибності умови.

Do Until <умова>

…

Loop

Do

…

Loop Until <умова>

# Використання списків

Списки у Visual Basic зручно обробляти використовуючи елементи управління ListBox. Наповнення списку найзручніше виконувати використовуючи метод AddItem. При викликові цього методу йому слід передати текстову стрічку, яка буде поміщена у список.

Наприклад

Private Sub Form\_Load()

Dim i As Integer

For i=1 to 10

List1.AddItem “Стрічка № “ & i

Next i

End Sub

Очищується список методом Clear.

Елементи списку зберігаються у вигляду масиву. До будь-якого з них можна звернутися при допомозі властивості List елемента управління ListBox. Індексація починається з нуля.

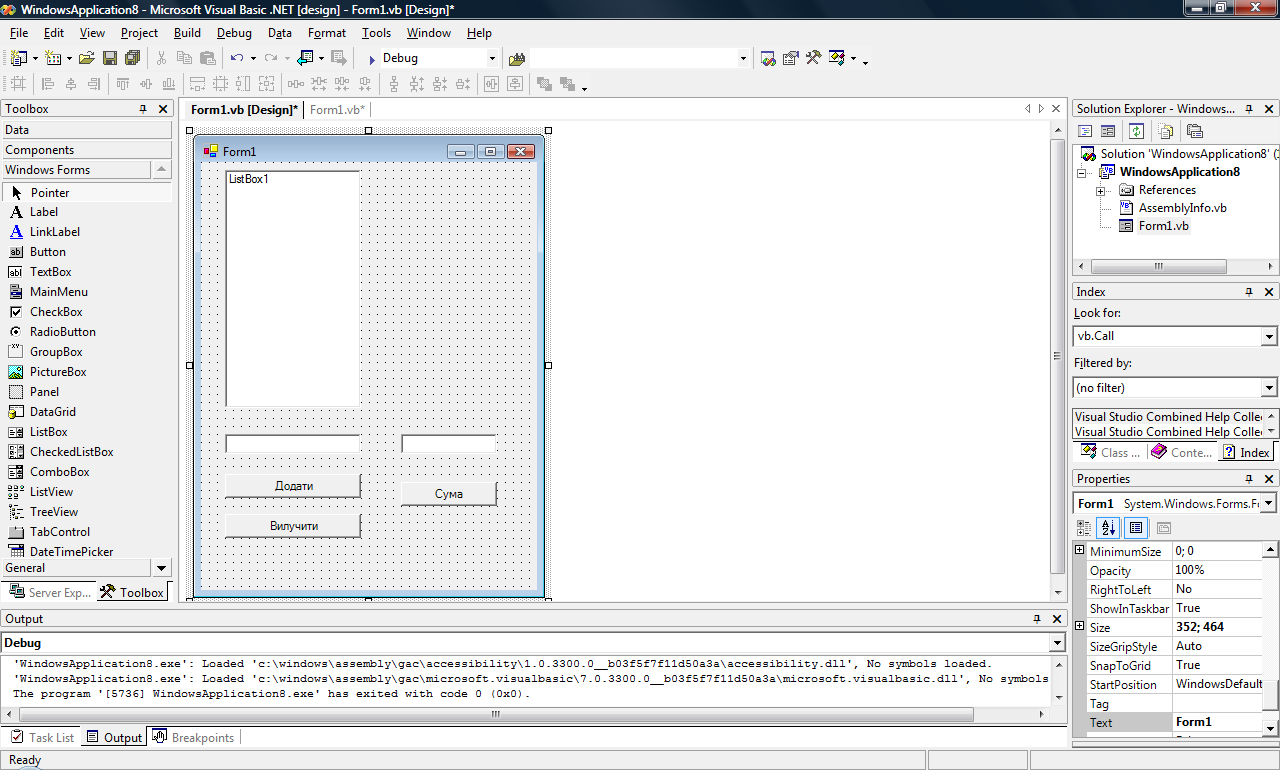
Наприклад, для виводу на екран першого елементу списку можна використати оператор Print List1.List(0).

Властивість ListCount визначає кількість елементів у списку, а ListIndex містить номер вибраного елементу списку.

Властивість Text містить значення вибраного елемента списку. Якщо жодний елемент списку не вибрано, то значення властивості Text рівне пустій стрічці (“”).

1. Створити новий проект.

2. Створити форму за таким зразком



3. До відповідних кнопок додати програмний код:

*Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click*

*If TextBox1.Text <> "" Then*

*ListBox1.Items.Add(TextBox1.Text)*

*TextBox1.Text = ""*

*Else*

*MsgBox("Повторіть ввід даних")*

*End If*

*End Sub*

*Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click*

*Dim i, n As Integer*

*Dim s As Decimal*

*s = 0*

*n = ListBox1.Items.Count - 1*

*For i = 0 To n*

*s = s + Val(ListBox1.Items(i))*

*Next*

*TextBox2.Text = s*

*End Sub*

*Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click*

*ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)*

*End Sub*

4. Виконати проект. При необхідності – відлагодити програмний код.

5. Сформувати список із 10 чисел.

6. Обчислити суму.

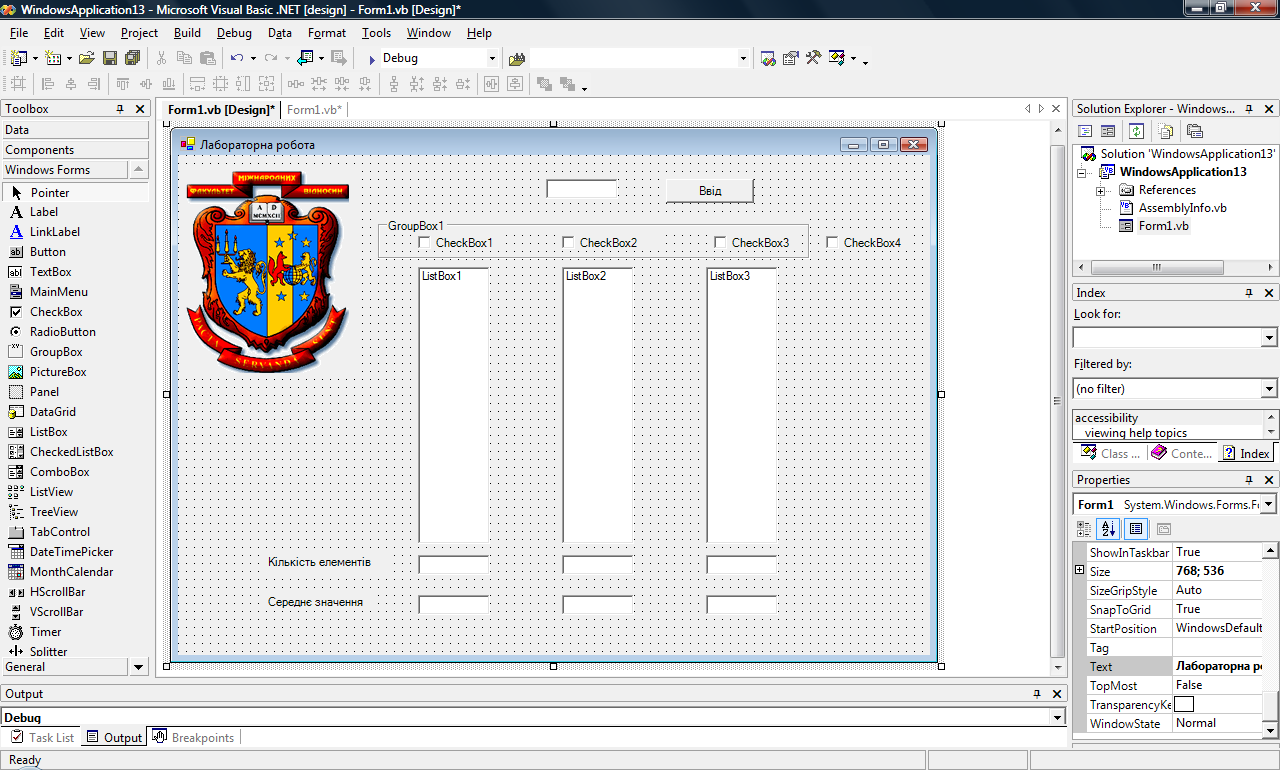
7. Зберегти проект і завершити роботу.

**Лабораторна робота 6**

**Побудова складних циклічних та розгалужувальних алгоритмів**

1. Створити новий проект.

2. Створити форму за таким зразком



3. Організувати ввід даних у кожний із списків, залежно від вказаної відмітки. Для цього використати програмний код

*Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click*

*Dim s1, s2 As Decimal*

*Dim i, j As Integer*

*If TextBox1.Text <> "" Then*

*If CheckBox1.Checked Then*

*ListBox1.Items.Add(TextBox1.Text)*

*End If*

*If CheckBox2.Checked Then*

*ListBox2.Items.Add(TextBox1.Text)*

*End If*

*If CheckBox3.Checked Then*

*ListBox3.Items.Add(TextBox1.Text)*

*End If*

*For j = 0 To ListBox1.Items.Count - 2*

*For i = 0 To ListBox1.Items.Count - 2*

*s1 = Val(ListBox1.Items(i))*

*s2 = Val(ListBox1.Items(i + 1))*

*If s2 > s1 Then*

*ListBox1.Items(i) = s2*

*ListBox1.Items(i + 1) = s1*

*End If*

*Next*

*Next*

*Else*

*MsgBox("Введіть дані")*

*End If*

*End Sub*

*Private Sub CheckBox4\_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CheckBox4.CheckedChanged*

*If CheckBox4.Checked() Then*

*CheckBox1.CheckState = 1*

*CheckBox2.CheckState = 1*

*CheckBox3.CheckState = 1*

*Else*

*CheckBox1.CheckState = 0*

*CheckBox2.CheckState = 0*

*CheckBox3.CheckState = 0*

*End If*

*End Sub*

4. Для кожного списку обчислити кількість елементів і середнє значення.

5. Забезпечити сортування другого списку за спаданням.

6. Виконати проект. При необхідності – відлагодити програмний код.

5. Сформувати списки із 10 чисел.

6. Провести обчислення.

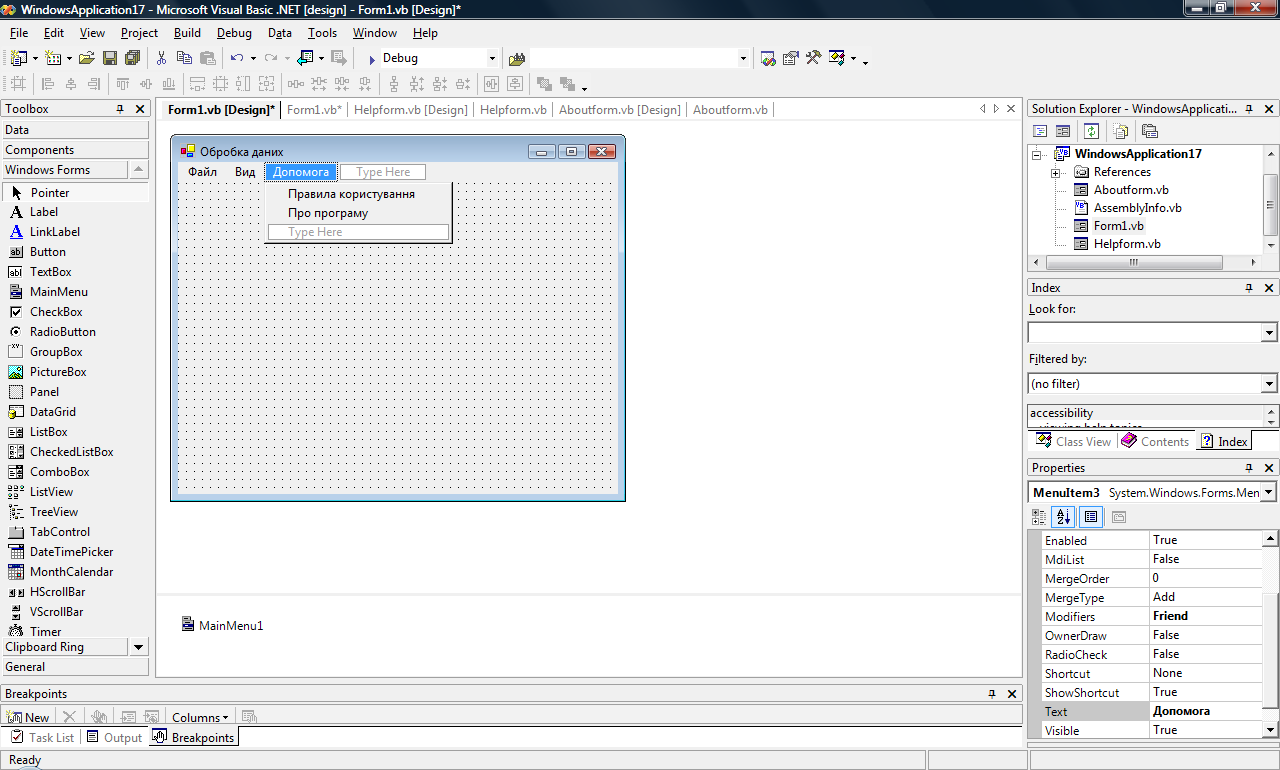
7. Зберегти проект і завершити роботу.

**Лабораторна робота 7**

**Створення проектів з багатьма формами**

1. Створити новий проект.

2. Створити форму, яка буде стартовою для даного проекту за таким зразком



3. Вставити головне меню, в якому передбачити команди “Правила користування” з кодом

Dim frmHelpDialog As New Helpform()

frmHelpDialog.ShowDialog()

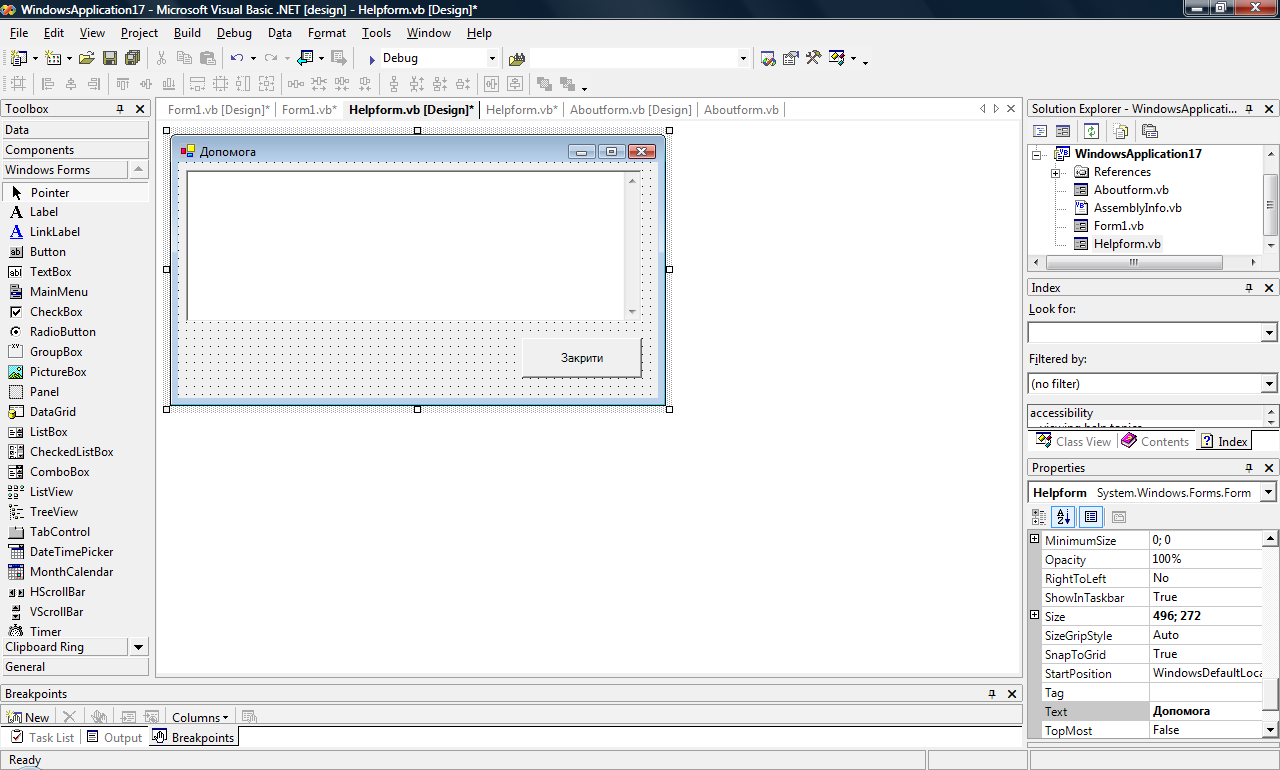
і “Про програму” з кодом

Dim frmAboutDialog As New Aboutform()

frmAboutDialog.Show()

4. Створити текстовий файл з інформацією про правила користування програмою і зберегти його під назвою “help.txt” у папці проекту в підкаталог “bin”.

5. Створити нову форму Helpform за таким зразком:



6. У процедурі обробки події натискання кнопки “Закрити” вставити код

Me.DialogResult = DialogResult.OK

7. У процедурі Helpform\_Load (для її виклику двічі клацнути мишкою по формі) вставити код

Dim StreamHelp As StreamReader

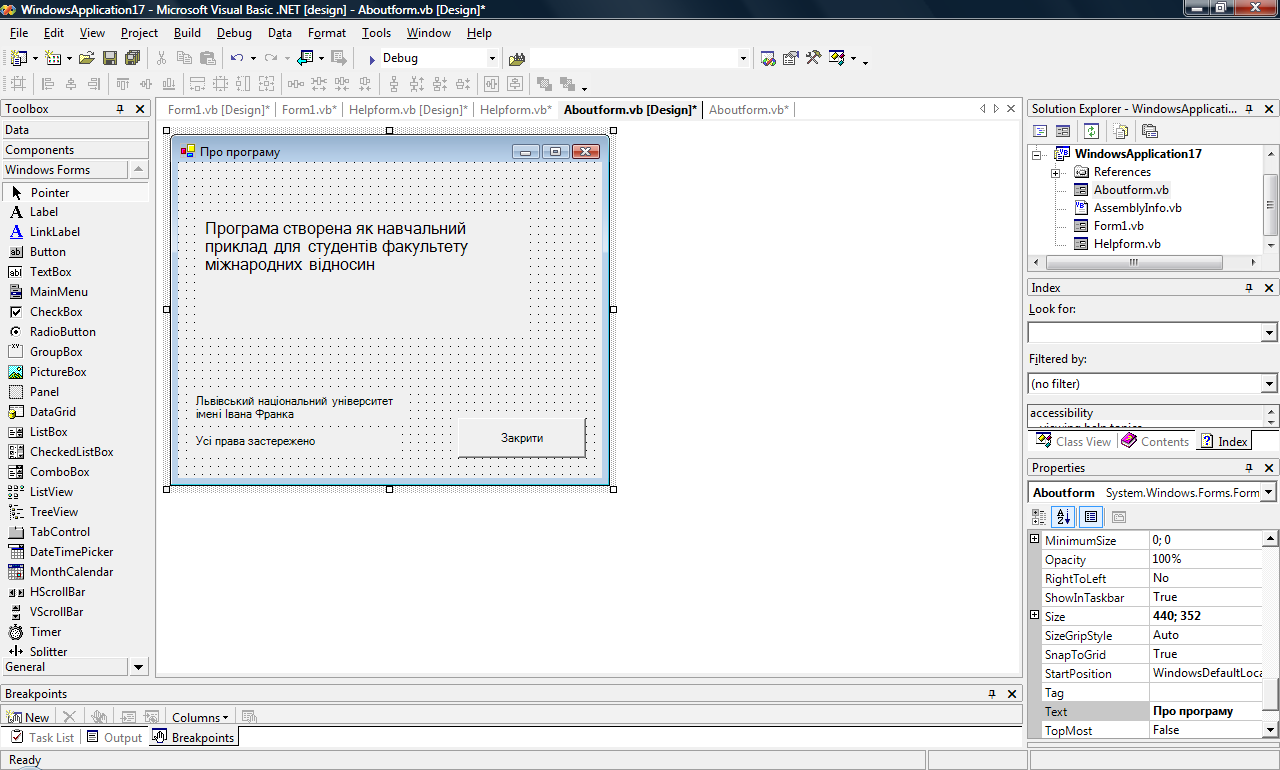
StreamHelp = New StreamReader("help.txt")

TextBox1.Text = StreamHelp.ReadToEnd

StreamHelp.Close()

TextBox1.Select(0,0)

8. Створити нову форму Aboutform за таким зразком



9. У процедурі обробки події натискання кнопки “Закрити” вставити код

Me.Close()

10. Виконати проект. При необхідності – відлагодити програмний код.

11.Зберегти проект і завершити роботу.

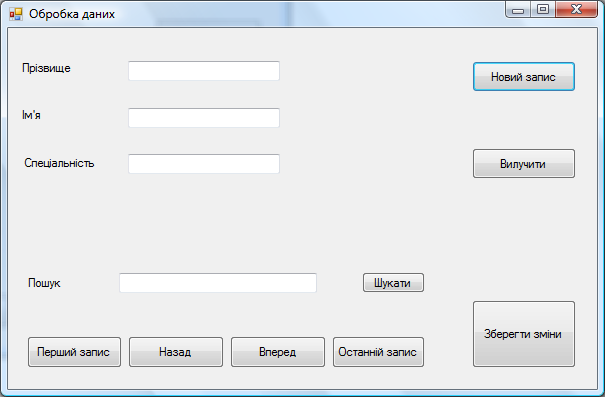
**Лабораторна робота 8**

**Обробка баз даних засобами Visual Basic**

1. Створити базу даних Students.

2. Вставити таблицю Stud з такими полями: prizv, name, spec і т.д.

3. Створити новий проект з формою такого змісту



4. Встановити з’єднання із базою даних.

Із меню Tools вибрати команду Connect to Database … Використовуючи закладку Підключення вказати безпосередньо вказати назву бази. Можна скористатися закладкою Постачальник даних і за допомогою OLE DB Provider вибрати базу даних із списку.

5. Створити адаптер даних.

Із області елементів управління в закладці Data вибрати елемент OleDbDataAdapter і перетягнути його в форму. Майстер налаштування адаптера запропонує вибрати базу даних (якщо вона ще не вибрана) і задати тип запитів – Use SQL statements. Задати SQL-запит, в якому організувати доступ до потрібних полів даної бази (для початку можна взяти всі). Внизу буде дві піктограми: адаптер даних і підключення.

6. Створити набір даних.

Із меню Data вибрати Generate Dataset і вказати назву нового набору даних. Внизу додасться піктограма набору даних.

7. Вставити необхідну кількість елементів Textbox і Label для відображення потрібної інформації з бази даних. Текстові поля слід зв’язати з відповідними полями бази даних. Для цього необхідно у вікні властивостей вибрати DataBindings/Text і вказати потрібне поле.

8. У форму вставити кнопку «Завантаження даних».

В процедуру обробки події \_Click для даної кнопки ввести код:

DataSet1.Clear()

OleDbDataAdapter1.Fill(DataSet1)

(Увага! Назви адаптера даних та набору даних слід вказувати такі, які є у Вашій формі)

9. Виконайте проект і перевірте завантаження даних.

10. Для навігації по базі створіть кнопки

«перший запис» з кодом

Me.BindingContext(DataSet1, «Stud»).Position = 0

«останній запис» -

Me.BindingContext(DataSet1, «Stud»).Position = Me.BindingContext(DataSet1, «Stud»).Count – 1

«наступний запис» -

Me.BindingContext(DataSet1, «Stud»).Position += 1

«попередній запис» -

Me.BindingContext(DataSet1, «Stud»).Position -= 1

(Тут «Stud» – назва таблиці в базі Students.

11. Для пошуку даних використати код

Dim f As Integer

f = Me.StudBindingSource.Find(«prizv», TextBox4.Text)

If f > -1 Then

Me.StudBindingSource.Position = f

Else

MsgBox(«Запис не знайдено»)

End If

12. Зберегти проект і завершити роботу.

**5. ОСНОВИ БАЗ ДАНИХ**

**Дванадцять правил Кодда, яким повинна відповідати реляційна СУБД**

1. *Правило інформації.* Вся інформація в базі даних повинна бути представлена винятково на логічному рівні і лише одним способом – у вигляді значень, що містяться в таблицях. Фактично це неформальне визначення реляційної бази даних.
2. *Правило гарантованого доступу.* Логічний доступ до всіх і кожного елементу даних (атомарному значенню) в реляційній базі даних повинна забезпечуватися шляхом використання комбінації імені таблиці, первинного ключа і імені стовпця. Правило 2 вказує на роль первинного ключа при пошуку інформації в базі даних. Ім'я таблиці дозволяє знайти необхідну таблицю, ім'я стовпця дозволяє знайти необхідний стовпець, а первинний ключ дозволяє знайти рядок, що містить шуканий елемент даних.
3. *Правило підтримки недійсних значень.* В реляційній базі даних повинна бути реалізована підтримка недійсних значень, які відрізняються від рядка символів нульової довжини, рядки пробільних символів і від нуля або будь-якого іншого числа і використовуються для представлення відсутніх даних незалежно від типу цих даних. Правило 3 вимагає, щоб відсутні дані можна було представити за допомогою недійсних значень (NULL).
4. *Правило динамічного каталога, заснованого на реляційній моделі*. Опис бази даних на логічному рівні повинен бути представлений в тому ж виді, що і основні дані, щоб користувачі, що володіють відповідними правами, могли працювати з ним за допомогою тієї ж реляційної мови, яку вони застосовують для роботи з основними даними. Правило 4 свідчить, що реляційна база даних повинна сама себе описувати. Іншими словами, база даних повинна містити набір системних таблиць, що описують структуру самої бази даних.
5. *Правило вичерпної підмови даних.* Реляційна система може підтримувати різні мови і режими взаємодії з користувачем (наприклад, режим питань і відповідей). Проте повинна існувати принаймні одна мова, оператори якої можна представити у вигляді рядків символів відповідно до деякого чітко певного синтаксису і який повною мірою підтримує наступні елементи:

- визначення даних;  
- визначення представлень;  
- обробку даних (інтерактивну і програмну);  
- умови цілісності;  
- ідентифікація прав доступу;  
- межі транзакцій (початок, завершення і відміна).  
  
Правило 5 вимагає, щоб СУБД використовувала мову реляційної бази даних. Така мова повинна підтримувати всі основні функції СУБД - створення бази даних, читання і ввід даних, реалізацію захисту бази даних і т.д.

1. *Правило оновлення представлень.* Всі представлення, які теоретично можна відновити, повинні бути доступні для оновлення. Правило 6 відноситься представлень, які є віртуальними таблицями, які дозволяють показувати різним користувачам різні фрагменти структури бази даних.
2. *Правило доповнення, оновлення і вилучення.* Можливість працювати з відношенням як з одним операндом повинна існувати не тільки при читанні даних, але і при доповненні, оновленні і видаленні даних. Правило 7 акцентує увагу на тому, що бази даних по своїй природі орієнтовані на множини. Воно вимагає, щоб операції додавання, видалення і оновлення можна було виконувати над множинами стрічок.
3. *Правило незалежності фізичних даних.* Прикладні програми і утиліти для роботи з даними повинні на логічному рівні залишатися незмінними при будь-яких змінах способів зберігання даних або методів доступу до них.
4. *Правило незалежності логічних даних.* Прикладні програми і утиліти для роботи з даними повинні на логічному рівні залишатися незмінними при внесенні в базові таблиці будь-яких змін, які теоретично дозволяють зберегти незмінні дані, що містяться в цих таблицях. Правила 8 і 9 означають відокремлення користувача і прикладної програми від низькорівневої реалізації бази даних.
5. *Правило незалежності умов цілісності.* Повинна існувати можливість визначити умови цілісності, специфічні для конкретної реляційної бази даних, на мові реляційної бази даних і зберігати їх в каталозі, а не в прикладній програмі. Правило 10 свідчить, що мова бази даних повинна підтримувати обмежені умови, що накладаються на дані і дії, які можуть бути виконані над даними, що вводяться.
6. *Правило незалежності розповсюдження.* Реляційна СУБД не повинна залежати від потреб конкретного клієнта. Правило 11 свідчить, що мова бази даних повинна забезпечувати можливість роботи з розподіленими даними, розташованими на інших комп'ютерних системах.
7. *Правило єдиності.* Якщо в реляційній системі є низькорівнева мова (яка обробляє один запис за один раз), то повинна бути відсутня можливість використання її для того, щоб обійти правила і умови цілісності, виражені на реляційній мові високого рівня (яка обробляє декілька записів за один раз). Правило 12 запобігає використанню інших можливостей для роботи з базою даних крім мови бази даних, оскільки це може порушити її цілісність.

# SQL

SQL є інструментом для вибірки і обробки інформації, яка міститься в комп'ютерній базі даних. Унікальність цієї мови полягає в тому, що він є єдиною стандартною мовою баз даних. Мову SQL підтримують понад 100 СУБД (Системи Управління Базами Даних), які працюють як на мейнфреймах, так і на звичайних ПК. Причму, кожна СУБД по-своєму інтерпретує стандарт, вносить свої доповнення, стандартний SQL підтримують всі бази даних.

Для отримання необхідної інформації з бази даних, користувач вносить запит у за допомогою SQL. СУБД обробляє запит, знаходить (або не знаходить) необхідні дані і повертає результат користувачу.

Що таке запит?

Запит - команда яку ви даєте вашій програмі бази даних, і яка повідомляє її щоб вона вивела певну інформацію з таблиць в пам'ять. Ця інформація звичайно посилається безпосередньо на екран комп'ютера або терміналу яким ви користуєтеся, хоч, в більшості випадків, її можна також надіслати принтеру, зберегти у файлі (як об'єкт в пам'яті комп'ютера), або представити як ввідну інформацію для іншої команди або процесу.

Функціональні можливості SQL. SQL використовується для реалізації всіх функціональних можливостей, СУБД, що надаються. До них відносяться:

* Організація даних. SQL надає користувачу можливість змінювати структуру представлення даних, а також встановлювати відносини між елементами бази даних.
* Вибірка даних. SQL дає користувачу або додатку можливість витягувати з бази даних інформацію, що міститься в ній, і користуватися нею.
* Обробка даних. SQL дає користувачу або додатку можливість змінювати базу даних, тобто додавати в неї нові дані, а також видаляти або модифікувати наявні дані.
* Управління доступом. За допомогою SQL можна обмежити можливості користувача по вибірці і зміні даних і захистити їх від несанкціонованого доступу.
* Сумісне використання даних. SQL дозволяє координувати сумісне використання даних користувачами, які працюють паралельно, щоб вони не заважали один одному.
* Цілісність даних. SQL дозволяє забезпечити цілісність бази даних, захищаючи її від пошкодження через зміни або відмову системи.

SQL не є повноцінною комп'ютерною мовою. SQL - це слабо структурована мова, інструкції якої звичайно вбудовуються в базову мову, наприклад C++ ч Visual Basic, і дають можливість діставати доступ до баз даних. SQL є невід'ємною частиною СУБД і працює тільки з реляційними базами даних.

# Оператори мови SQL.

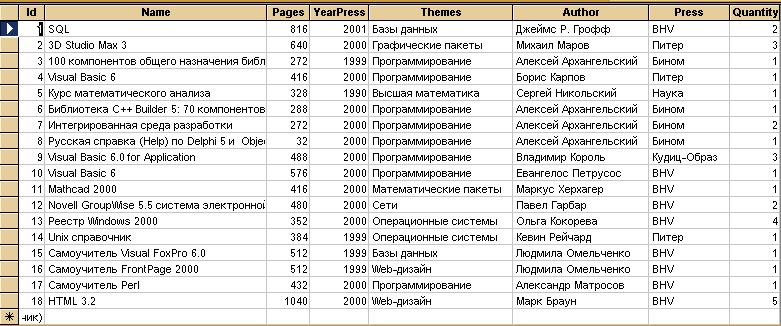
Команди мови SQL можна поділити на три категорії:

* DDL - **Data Definition Language** (Мова Визначення Даних) - складається з команд, які створюють об'єкти (таблиці, індекси, представлення, і так далі) в базі даних.
* DML - **Data Manipulation Language** (Мова Маніпулювання Даними) - це набір команд, які визначають які значення представлені в таблицях у будь-який момент часу.
* DCL - **Data Control Language** (Мова Управління Даними) - складається із засобів, які визначають, чи дозволити користувачу виконувати певні дії чи ні.

Розглянемо основні оператори DDL.

### Вибірка даних - оператор SELECT.

Припустимо у нас є таблиця, в якій міститься інформація про наявні книги і їх атрибути (див. малюнок).



Запит, що дозволяє відобразити всю таблицю (вона називається Books) цілком, виглядає таким чином:

Select \* From Books

Тут: Select - оператор вибірки даних; \* - указує, що повинні вибиратися всі стовпці в тому ж порядку, як вони визначені в базі даних; ключове слово From указує звідки повинні вибиратися дані (ім'я таблиці).

Якщо ж користувачу не потрібна повна інформація, він може вказати які стовпці необхідно вибирати і в якому порядку, розташувавши імена стовпців після інструкції Select, і розділивши їх комами.

Select Name, Author, Press, Pages

From Books

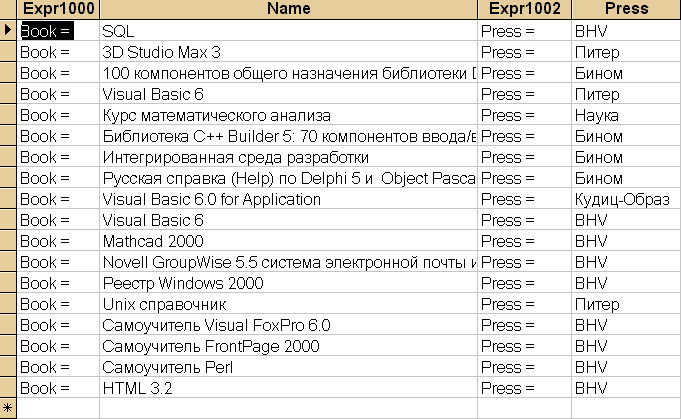
В цьому випадку користувач вибирає тільки назви книг, їх авторів, видавництво і кількість сторінок.

Користувач також має нагоду форматувати виведення результату запиту на екран. Наприклад, результат запиту

Select 'Book = ', Name, 'Press = ', Press

From Books

дає наступні результати (див. малюнок).

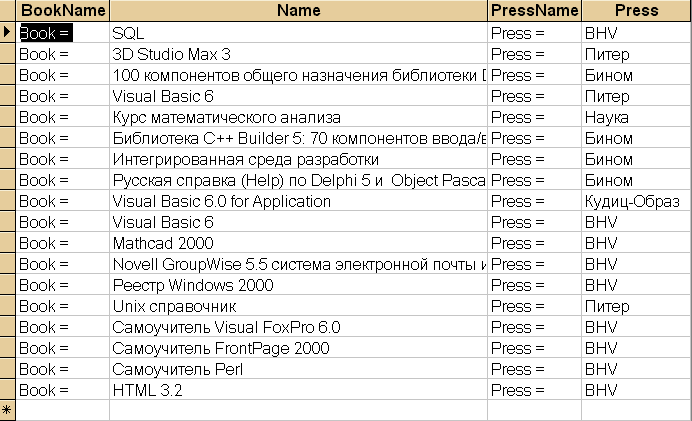


Як можна відмітити, у текстових стовпців не дуже зрозуміла назва (Expr2000 і т. п.). Так відбувається тому, що таких стовпців в базі не існує і Access дає їм власні назви. Щоб цього уникнути, можна дати цим стовпцям псевдоніми.

Select 'Book = ' as BookName, Name, 'Press = 'as PressName, Press

From Books

Результат представлений на малюнку.



До речі, такий псевдонім можна призначити навіть існуючому в таблиці стовпцю.

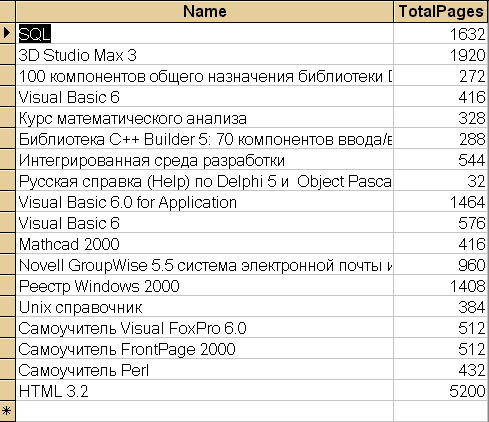
SQL також дозволяє створювати "на льоту" стовпці, значення в яких будуються на основі якого-небудь виразу, в якому беруть участь значення "реальних" стовпців, в кожному рядку відповідно. Такий стовпець називається обчислюваним.

Наприклад, обчислимо загальну кількість сторінок по наявних в наявності книгах.

Select Name, Quantity \* Pages as TotalPages

From Books

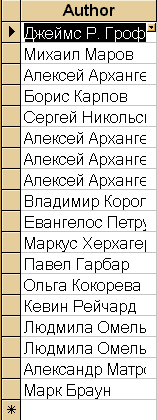
Результат виконання запиту представлений на малюнку.



Бувають випадки, коли в результуючому наборі рядків з'являються дублікати (ідентичні рядки). Наприклад, розглянемо наступний запит:

Select Author From Books

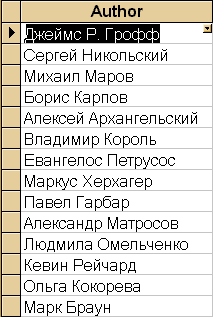
Результат виконання запиту представлений на малюнку.



У даному випадку ми одержуємо список всіх авторів в таблиці, але оскільки деякі з них написали більше однієї книги, то в результуючому наборі зустрічаються однакові рядки. Якщо нам це не потрібно, то існує можливість виключити з результуючого набору рядків дублікати за допомогою ключового слова Distinct. Перепишемо запит:

Select Distinct Author From Books

Результат виконання запиту представлений на малюнку.



Як бачите, рядки-дублікати більше не з'являються перед наші світлі очі.

далеко не завжди потрібно вибирати всі рядки таблиці. Якщо користувачу потрібна якась конкретна інформація, то він може обмежити результуючий набір рядків, використовуючи додаткові умови (уточнюючи запит).

Для установки умови використовується ключове слово Where і набір логічних операторів:

* > - більше
* >= - більше або рівно
* < - менше
* <= - менше або рівно
* = - рівно
* <> - не рівно (в деяких базах використовується знак !=)
* and - логічне "И" (множення)
* or - логічне "або" (складання)
* not - логічне "не" (заперечення)
* between - приналежність діапазону
* in - перевірка на членство в множині
* like - перевірка на відповідність шаблону
* is null - перевірка на рівність значенню NULL

Під значенням NULL розуміють невизначене значення. Результатом логічного виразу в SQL може бути або істина, або брехня, або NULL (коли будь-яка з частин виразу рівна NULL). Рядок включатиметься в результуючий набір тільки в тому випадку, якщо результат перевірки умов відбору рівний істині.

Приведемо декілька прикладів:

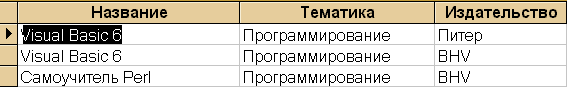
Запит: необхідно вивести книги по програмуванню издательств "Пітер" і "BHV"

Select Name as Назва, Themes as Тематика, Press as Видавництво

From Books

Where Themes = 'Программирование' and (Press = 'BHV' or Press = 'Питер')

Результат виконання запиту представлений на малюнку.



Зауваження: рядки в SQL беруться в одинарні лапки

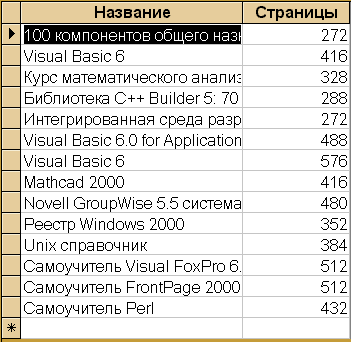
Запит: відобразити всі книги, у яких кількість сторінок лежить в межах від 200 до 600

Select Name as Назва, Pages as Сторінки

From Books

Where Pages Between 200 And 600

Результат виконання запиту представлений на малюнку.



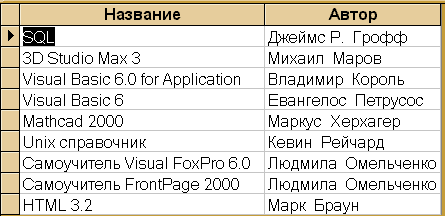
Запит: відобразити всі книги, імена авторів яких лежать в діапазоні від букви 'В' до 'О'

Select Name as Назва, Author as Автор

From Books

Where Name Between 'В' And 'О'

Результат виконання запиту представлений на малюнку.



Зауваження: при порівнянні з текстом, відбір здійснюється згідно ASCII-кодам символів

Запит: вибрати книги, що відносяться до програмування або до баз даних, і видавництва яких не 'Питер'и не 'Бином'

Select Name as Назва, Themes as Тематика, Press as Видавництво

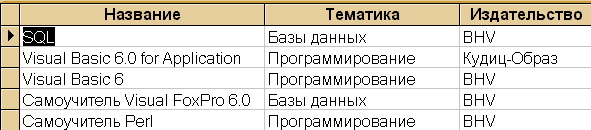
From Books

Where Press not in ('Питер','Бином')

and

Themes in ('Программирование','Базы данных')

Результат виконання запиту представлений на малюнку.



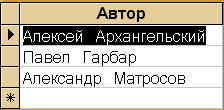
Запит: вибрати з таблиці тих авторів, у яких в імені і прізвищі не менше трьох букв 'а'

Select Name as Автор

From Books

Where Name Like '\*а\*а\*а\*'

Результат виконання запиту представлений на малюнку.



Оператор Like реалізує пошук за шаблоном:

* \* - означає, що даній позиції може бути присутній 0 або більш будь-яких символів (в інших СУБД застосовується символ %);
* ? - означає, що в даній позиції зобов'язаний бути присутній 1 будь-який символ (в інших СУБД застосовується символ \_);
* # - означає, що в даній позиції зобов'язана бути присутній 1 будь-яка цифра;
* [а-z] - означає, що в даній позиції зобов'язаний бути присутній 1 символ з вказаного діапазону;
* [dfaf] - означає, що в даній позиції зобов'язаний бути присутній 1 символ з вказаної множини;
* [!safgwe] - означає, що в даній позиції зобов'язаний бути присутній 1 символ, що не входить у вказаний діапазон.

Для сортування результуючих рядків використовується оператор Order з необов'язковим параметром Asc (стоїть за умовчанням) - сортування за збільшенням (за абеткою для рядків), або Desc - сортування по убуванню.

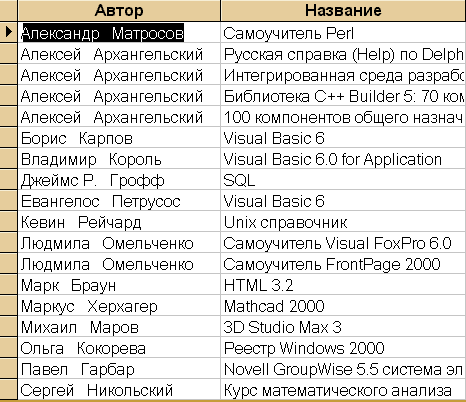
Запит: відобразити всіх авторів і їх книги, авторів відсортувати за збільшенням, а назви книг (по авторах) по убуванню (вторинне сортування)

Select Author as Автор, Name as Назва

From Books

Order Author, Name Desc

Результат виконання запиту представлений на малюнку.



#### Підсумок:

Запит на вибірку даних підкоряється строгій структурі:

1. Спочатку указується що вибрати і як відобразити (Select)
2. Потім звідки вибрати (From)
3. Як вибрати (Where)
4. Як сортувати (Order)

### Вставка даних - оператор INSERT.

Інструкція вставки працює за наступними правилами:

Insert Into Books

Values (100, 'Колобок', 10, 2001, 'Сказки', 'Народ', 'Москва', 1)

в даному випадку значення вставляються у всі стовпці таблиці відповідно до їх фізичного порядку. Якщо необхідно вставити значення тільки в деякі стовпці, то необхідно їх явно указувати:

Insert Into Books (Id, Name, Themes, Quantity)

Values (1000, 'Колобок', 'Сказки', 1)

Значення стовпців повинні йти у відповідному порядку і бути відповідного типу.

### Модифікація даних - оператор UPDATE.

Ця інструкція дозволяє модифікувати існуючі значення стовпців

Наприклад:

Update Books

Set Quantity = 0

Where Themes = 'Web-дизайн'

даний запит встановлює кількість книг по дизайну в 0.

В інструкції Set можна обновляти відразу декілька стовпців, перерахувавши їх через кому:

Update Books

Set Quantity = Quantity + 1, YearPress = 2002

Where Themes = 'Web-дизайн'

Також можна встановлювати нові значення, базуючись на попередніх (див. попередній запит).

Примітка: не забувайте ставити умову для оновлення даних (Where), інакше інструкція Update встановить нове значення для всього стовпця.

### Видалення даних - оператор DELETE.

А ось цю інструкцію слід використовувати акуратно. Наприклад, запит

Delete From Books

видалить вміст всієї таблиці

Знову-таки не забувайте ставити умови. Наприклад, необхідно видалити з таблиці всі записи для авторів з ім'ям, яке починається на букву 'А'

Delete

From Books

Where Author Like 'А\*'

Примітка: інструкції Insert, Update, Delete не повертають набору рядків - вони просто виконуються.

# Нормалізація. Нормальні форми

Розглянемо докладніше процес проектування бази даних. Отже перед Вами стоїть якась задача. Наприклад, створити базу даних, в якій зберігатиметься інформація про співробітників підприємства. Ви подумали...Виконали пару-трійку рухів тіла...Сели за комп'ютер...І ось результат в наявності!!! У вас з'явилася БАЗА. Проте, щось вам заважає спокійно пити пиво після виконаної роботи. А чи є Ваше дітище оптимально спроектованим, а може їсти якась надмірність (наприклад в двох таблицях повторюється одна і та ж інформація) і т.д.. Як перевірити базу на оптимальність? Не варто винаходити велосипед і придумувати якісь свої способи аналізу. Потрібно просто скористатися так званими нормальними формами. Це закони, за допомогою яких можна поліпшити структуру бази. Перейдемо до самого поняття нормалізації і нормальних форм.

*Нормалізація* – це розбиття таблиці на дві або більш, володіючих кращими властивостями при включенні, зміні і видаленні даних. Остаточна мета нормалізації зводиться до отримання такого проекту бази даних, в якому *кожний факт з'являється лише в одному місці,* тобто виключена надмірність інформації. Це робиться не стільки з метою економії пам'яті, скільки для виключення можливої суперечності бережених даних.

Кожна таблиця в реляційній БД задовольняє умові, відповідно до якої в позиції на перетині кожного рядка і стовпця таблиці завжди знаходиться єдине атомарне значення( т.е в одному осередку знаходиться одне значення: в текстовому - текст, в числовому тільки одне число, а не група чисел перерахованих через кому ), і ніколи не може бути безлічі таких значень. Будь-яка таблиця, що задовольняє цій умові, називається *нормалізованою*. Фактично, ненормалізовані таблиці, тобто таблиці, що містять групи, що повторюються, навіть не допускаються в реляційній БД.

Всяка нормалізована таблиця автоматично вважається таблицею в *першій нормальній формі*, скорочено *1НФ. Таким чином, строго кажучи, "нормалізована" і "знаходилася в 1НФ" означають одне і те ж. Проте на практиці термін "нормалізована" часто використовується в більш вузькому значенні – "повністю нормалізована", який означає, що в проекті не порушуються ніякі принципи нормалізації.*

Тепер на додаток до 1НФ можна визначити подальші рівні нормалізації – *другу нормальну форму* (*2НФ*), *третю нормальну форму* (*3НФ*) і т.д. По суті, таблиця знаходиться в 2НФ, якщо вона знаходиться в 1НФ і задовольняє, крім того, деякій додатковій умові, суть якої буде розглянута нижче. Таблиця знаходиться в 3НФ, якщо вона знаходиться в 2НФ і, крім цього, задовольняє ще іншій додатковій умові і т.д.

Таким чином, кожна нормальна форма є в деякому розумінні більш обмеженою, але і *бажанішої*, ніж передуюча. Це зв'язано з тим, що "(N+1)-а нормальна форма" не володіє деякими непривабливими особливостями, властивим "N-й нормальній формі". Загальне значення додаткової умови, що накладається на (N+1) -у нормальну форму по відношенню до N-й нормальної форми, полягає у виключенні цих непривабливих особливостей.

А зараз перейдемо до більш строгого визначення 1НФ, 2НФ і 3НФ.

Таблиця знаходиться в *першій нормальній формі* (*1НФ*) тоді і тільки тоді, коли жоден з її рядків не містить в будь-якому своєму полі більше одного значення і жодне з її ключових полів не порожнє.

Для розгляду 2НФ і 3НФ введемо додаткові поняття. *Функціональна залежність. Поле В таблиці функціонально залежить від поля А тієї ж таблиці в тому і лише у тому випадку, коли в будь-який заданий момент часу для кожного з різних значень поля А обов'язково існує тільки одне з різних значень поля В. Відзначимо, що тут допускається, що поля А і В можуть бути складовими.*

*Повна функціональна залежність*. Поле В знаходиться в повній функціональній залежності від складового поля А, якщо воно функціональне залежить від А і не залежить функціонально від будь-якої підмножини поля А.

Таблиця знаходиться в другій *нормальній формі* (2НФ*)*, якщо вона задовольняє визначенню 1НФ і всі її поля, що не входять в первинний ключ, зв'язані повною функціональною залежністю з первинним ключем.

Таблиця знаходиться в *третій нормальній формі* (*3НФ*), якщо вона задовольняє визначенню 2НФ і не одне з її неключових полів не залежить функціонально від будь-якого іншого неключового поля.

Існують також інші нормальні форми, проте звичайно нормалізацію закінчують на 3НФ.