

**Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ**



**МЕХАНІКО-МАШИНОБУДІВНИЙ ФАКУЛЬТЕТ**  
*Кафедра основ конструювання механізмів і машин*

**МАТЕРІАЛИ МЕТОДИЧНОГО ЗАБЕЗПЕЧЕННЯ**  
**дисципліни «Інформатика»**  
**модуль 4**  
для студентів напрямку підготовки  
6.070106 Автомобільний транспорт

**Дніпропетровськ**  
**НГУ**  
**2008**

Матеріали методичного забезпечення дисципліни «Інформатика» для студентів напряму підготовки 6.070106 Автомобільний транспорт / Упоряд.: І.М. Мацюк, І.В. Вернер, Т.О. Письменкова. – Д.: НГУ, 2008. – 26 с.

Упорядники:

І.М. Мацюк, доц. (розділи 3-6)

І.В. Вернер, асс. (розділи 3-5, 7);

Т.О. Письменкова, асс. (розділи 1-3, 6-7);

Затверджено методичною комісією з напряму 070106 Автомобільний транспорт (протокол № 11 від 18.06.2008) за поданням кафедри основ конструювання машин і механізмів (протокол № 1 від 24.06.2008).

Подано методичні рекомендації з виконання лабораторних робіт модуля 4 з дисципліни «Інформатика» освітньо-кваліфікаційної програми підготовки бакалаврів для студентів напряму підготовки 6.070106 Автомобільний транспорт

Відповідальний за випуск завідувач кафедри основ конструювання механізмів і машин к.т.н., доц. К.А. Зіборов.

## Передмова

Лабораторна робота – форма навчального заняття, при якому студент під керівництвом викладача особисто проводить натуральні або імітаційні експерименти з метою практичного підтвердження окремих теоретичних положень певної навчальної дисципліни. При цьому він набуває навичок у роботі з лабораторним устаткуванням, обладнанням, обчислювальною технікою, вимірювальною апаратурою, методикою експериментальних досліджень у певній галузі.

Лабораторні роботи відіграють важливу роль у підготовці фахівців через самостійну виконавчу діяльність.

Лабораторні заняття будуються відповідно до робочої програми дисципліни. Програмою курсу передбачено чотири лабораторні модулі.

Зміст четвертого лабораторного модуля передбачає вивчення:

Загальних відомостей та принципів роботи глобальної мережі Інтернет. Протоколи передачі даних. Пошукові системи. Принцип роботи електронної пошти.

Основ програмування, структур мови Visual Basic. Типів змінних.

Середовища розробки Visual Basic 6.0. Принципів роботи. Елементів керування.

Роботи зі змінними й константами. Особливостей роботи з конструкціями керування.

Роботи із циклами. Створення користувальницьких процедур і функцій.

Об'єктно-орієнтованого програмування. Роботи з об'єктами

Розробки додатків з декількома формами. Налаштування додатків. Обробку помилок.

Малювання в Visual Basic. Поняття координатної системи. Координатні властивості й методи. Роботи з файлами.

Розвиваюча мета передбачає вдосконалення здібностей студентів, їх пам'яті, логічного мислення, пов'язаних з досягненнями в навчальній діяльності.

Мета даних методичних вказівок – допомогти студентам в опануванні навчального матеріалу та в підготовці до контрольних заходів 4 модуля дисципліни «Інформатика».

### Цілі лабораторної роботи

Дидактичними цілями лабораторної роботи є:

- навчання студента умінню використовувати теоретичні положення для рішення конкретних практичних задач;
- оволодіння специфікою роботи Visual Basic;
- набуття навичок програмування в Visual Basic.

### Організація виконання лабораторної роботи

Лабораторні заняття проводяться в спеціально обладнаних навчальних лабораторіях з використанням комп'ютеру.

Лабораторні заняття мають на меті придбання та закріплення базових знань, умінь та навичок.

Перед початком виконання лабораторних робіт, проводиться вступний інструктаж з правил техніки безпеки. Для реєстрації інструктажу з техніки безпеки використовується спеціальний журнал, у якому кожний студент повинний розписатися про ознайомлення з правилами поведінки в комп'ютерному класі і використанням комп'ютеру.

Студент під час проведення лабораторних робіт повинен:

- беззаперечно дотримуватись правил охорони праці;
- ознайомитись з методичними рекомендаціями щодо проведення лабораторних робіт;
- виконати лабораторну роботу за відповідною методикою;
- захистити результати лабораторної роботи;
- отримати оцінку за лабораторний модуль через визначену форму модульного контролю.

Викладач повинен:

- провести інструктаж студентів щодо правил безпеки;
- керувати проведенням лабораторної роботи;
- здійснити поточний контроль опанування студентами методичних рекомендацій;
- забезпечити дотримання правил безпеки при виконанні лабораторних робіт;
- скласти графік консультацій;
- додержуватись графіку консультацій;
- оцінити навчальну діяльність студента з опанування лабораторного модуля.

### **Лабораторна робота №1**

**Тема:** *Робота з інтерфейсом середовища розробки Visual Basic 6.0. Робота зі змінними і програмним кодом.*

**Ціль:** *Вивчити структуру програми, типи змінних, основні операції та пріоритет їх виконання, синтаксис запису операторів.*

#### **Теоретичні відомості щодо ключових питань завдання**

##### *Структура програми*

Програмування на Visual Basic 6.0. складається з декількох етапів:

- візуальне програмування, де складається дизайн додатка, який складається з вікон додатків та керуючих елементів;
- написання програмного коду в відповідності з язиком програмування.

Усередині модуля й форми може розташовуватися послідовність процедур і функцій у довільному порядку. Кожна процедура або функція складається з послідовності операторів. Кожний оператор записується в окремому рядку. Дуже довгі оператори можна переносити на наступний рядок, указавши наприкінці рядка символ підкреслення «\_». При записі операторів не розрізняються прописні й малі літери. Можна використовувати російські букви (у зв'язку із цим потрібно бути дуже уважним при наборі однакових за написанням латинських і російських букв!). Програма може містити рядки коментарів. Рядок коментарів починається зі знака апостроф « ' ».

Текст процедури і функції містить оголошення змінних, коментарі і виконуючі оператори. Оголошення змінної повинне передувати її першому використанню (можливий варіант роботи без оголошення змінних). Найкраще оголошення змінних виконувати на початку програми. Змінні, оголошені в модулі, можуть використовуватися як загальні змінні у всіх процедурах модуля і форм.

#### *Типи змінних*

Кожна змінна має ім'я (ідентифікатор). Ідентифікатор змінної може містити до 255 букв і цифр. Першим символом повинна бути буква.

У процедурі або функції змінна оголошується за допомогою оператора Dim, наприклад, у такий спосіб:

Dim X as Single, Y as Long, Z as Single

Якщо в програмі змінна не оголошена, то за замовчуванням вважається, що вона має тип Variant.

Оголошення змінної - це не формальна процедура, тому що залежно від оголошення реалізуються різні алгоритми кодування й декодування числових значень у комірках пам'яті комп'ютера.

Змінні можуть бути наступних основних типів:

Single - це речовинне (із дробовою частиною) число. Займає 4 байти пам'яті. Точність подання - 7 десяткових знаків після коми. Наприклад, 4/3 буде представлено як 0,1333333x10<sup>1</sup>. Межі подання Single: від 1.4x10<sup>-45</sup> до 3.4x10<sup>38</sup>.

Double - це речовинне число. Займає 8 байт пам'яті. Точність подання - 14 десяткових знаків після коми. Межі подання Double - від ±4.9x10<sup>-324</sup> до ±1.79x10<sup>308</sup>.

Integer - це ціле зі знаком число. Займає 2 байти пам'яті. Межі - від -32768 до 32767.

Long - це ціле зі знаком число. Займає 4 байти пам'яті. Межі: ~±2 млрд.

Byte - це ціле позитивне число в межах від 0 до 255.

Variant - може приймати один з вище перелічених типів залежно від значення, що привласнюється. Займає 16 байт пам'яті. Використовується при роботі з об'єктами.

String - рядок символів. Може зберігати від 0 до майже 2 млрд. символів.

#### *Способи подання чисел*

Цілі числа - +5, -10, 5.

Речовинні числа - +8.1 означає 8,1

(Як роздільник цілої й дробової частини використовується не кома, крапка).

+8.1E2 означає 8,1x10<sup>2</sup>= 810

-8.1E-3 означає -8,1x10<sup>-3</sup>=-0.0081<sup>-3</sup> (експонентна форма)

#### *Оператор присвоювання*

Y =5 - змінної Y привласнює значення 5

Y=A+B - обчислюється значення виразу A+Y и привласнюється змінної Y.

У виразі можуть використовуватися наступні операції:

+ - додавання, - - віднімання \* - множення / - ділення \ - цілочислового ділення ^ - зведення в ступінь, ультрасучасний - залишок від цілочислового ділення.

Пріоритет арифметических операцій - загальноприйнятий, тобто ^ (зведення в ступінь) - найвищий пріоритет + або - - найнижчий пріоритет. Можна використовувати круглі дужки.

У виразах можуть використовуватися звертання до функцій у вигляді функція (аргумент). Звертання до функції має більш високий пріоритет, ніж зведення в ступінь. Можуть, наприклад, здійснюватися звертання до наступних стандартних функцій:

Sin (аргумент) - обчислення синуса

Cos (аргумент) - обчислення косинуса

Sqr (аргумент) - обчислення квадратного кореня

Tan (аргумент) - обчислення тангенса

Atn (аргумент) - обчислення арктангенса

Abs (аргумент) - абсолютне значення

Log (аргумент) - натуральний логарифм

Exp (аргумент) - e в ступені «аргумент»

Int (аргумент) - ціла частина числа без округлення

Str (число) - перетворення числа в рядок символів

Val (рядок) - перетворення рядка символів у числове значення. Якщо рядок не є числом, то результатом буде нуль.

PI - константа π. Функція без аргумента.

Приклад оператора присвоєння для виразу:

$$y = \frac{e^{-x^2} + \sin^3(x)}{4(x+5)} + \operatorname{tg}x$$

наведено нижче:

$$Y = (\exp(-x^2) + \sin(x)^3) / (4 * (x + 5)) + \tan(x)$$

### Завдання

Скласти програму з розрахунку ПДВ і обчисленню суми двох чисел.

Додаток складається з 2-х форм. Виклик другої форми здійснюється з першої за допомогою командної кнопки.

Зовнішній вигляд форм наведених на малюнках 1.1., 1.2. відповідно.

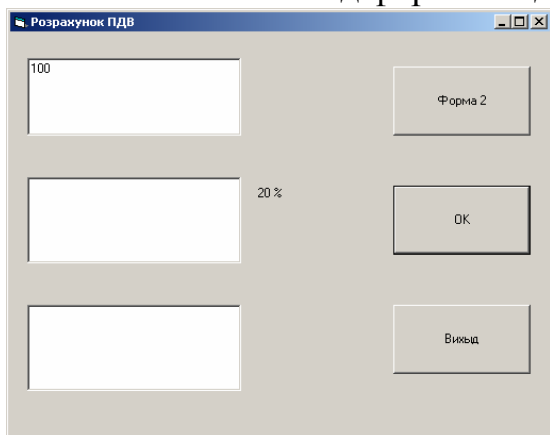


Рис. 1.1. Форма 1. «Розрахунок відсотків»

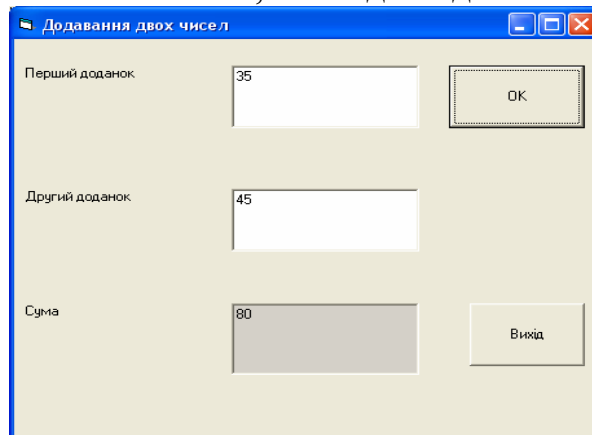


Рис. 1.2. Форма 2. «Додавання двох чисел»

### Хід роботи

Перше завдання - створити вікно додатка з розрахунку ПДВ за допомогою Visual Basic.

1. Запустіть Visual Basic за допомогою команди Пуск, Visual Basic 6.0. На екрані відобразиться вікно Project1 - Microsoft Visual Basic (рис. 1.3.). Це вікно має стандартний вигляд вікон додатків Windows: рядок заголовка, меню, панелі інструментів і декілька допоміжних вікон. Серед усіх допоміжних вікон нас цікавить вікно New Project менеджера проектів Project Wizard. Вікно New Project пропонує різні варіанти типів шаблонів проектів, які істотно полегшують процес проектування додатків.

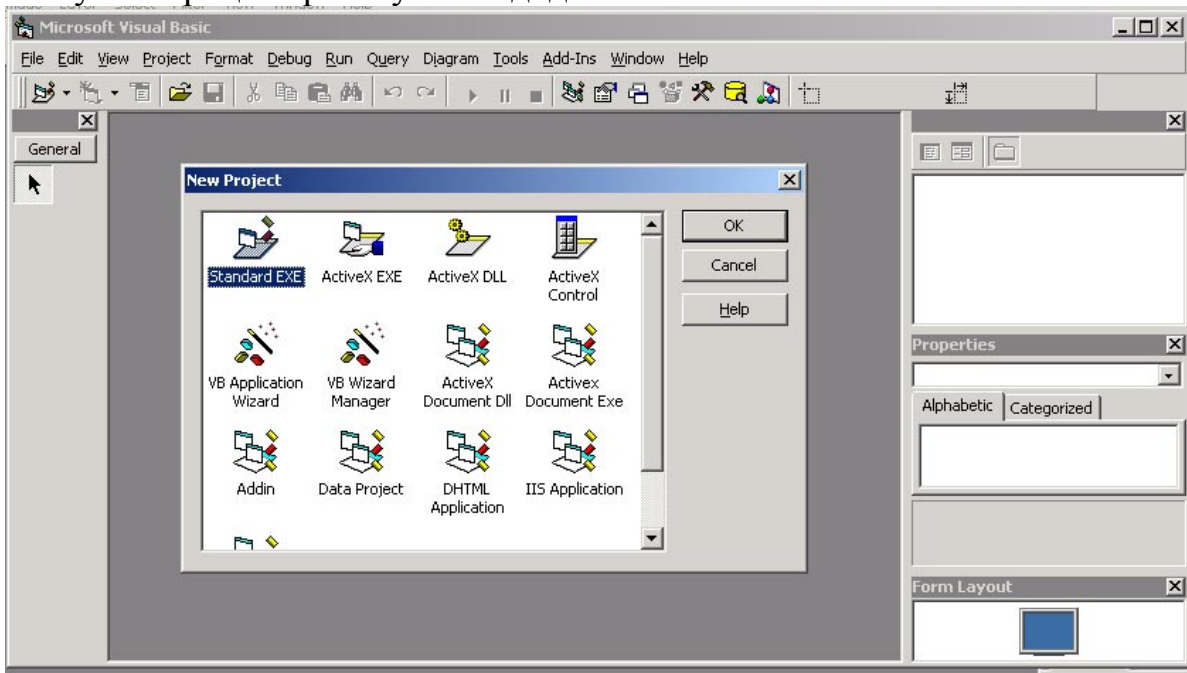


Рис. 1.3. Вікно New Project

2. Двічі натисніть значок Standard EXE зі списку вікна New Project ,що приведе до завершення роботи менеджера проектів Project Wizard і переходу в інтегроване середовище розробки (IDE, Integrated Development Environment), скомпонованого для розробки обраного типу проекту. IDE - це одна з основних складових Visual Basic, у ній конструюються й програмуються компоненти розробляємих додатків. IDE складається з декількох елементів. У цей момент нас цікавлять тільки два:

- Вікно Project1 - Form1(Form) (рис. 1.4.). Зовні воно схоже на макет діалогового вікна.

- Ліворуч від вікна Project1 - Form1(Form) розташована панель інструментів Toolbox (елементи керування), котра призначена для створення елементів керування у формі. Панель інструментів нагадує панель інструментів будь-якого графічного редактора, наприклад, Paint. Досить натиснути на кнопку конструйованого елемента і потім зафіксувати курсор у деякому місці форми й при натиснутій кнопці миші протягти його, позначивши під елемент поле.

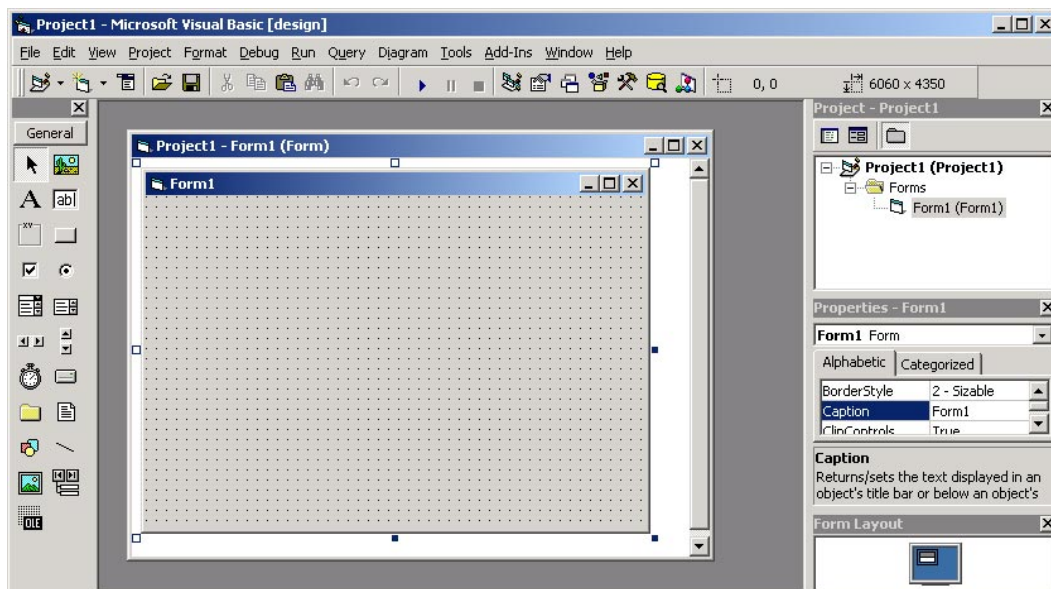





Рис. 1.4. Вікно Project1 - Form1(Form)

3. Зупинимося детальнішої на додатку, що ми збираємося створити. У діалоговому вікні повинні розташовуватися три поля введення куди вводиться перший доданок, другий доданок і сума. Обчислення, як звичайно, проводяться по натисканню кнопки, після набору в поля введення вихідних даних. Подивившись на спливаючі підказки елементів керування, визначимо, що поле введення створюється елементом (TextBox) , а командна кнопка — (CommandButton) . Допоміжні написи робляться за допомогою елемента — (label) .

4. За допомогою згаданих елементів керування сконструюємо діалогове вікно з розрахунку ПДВ (рис. 1.5.).

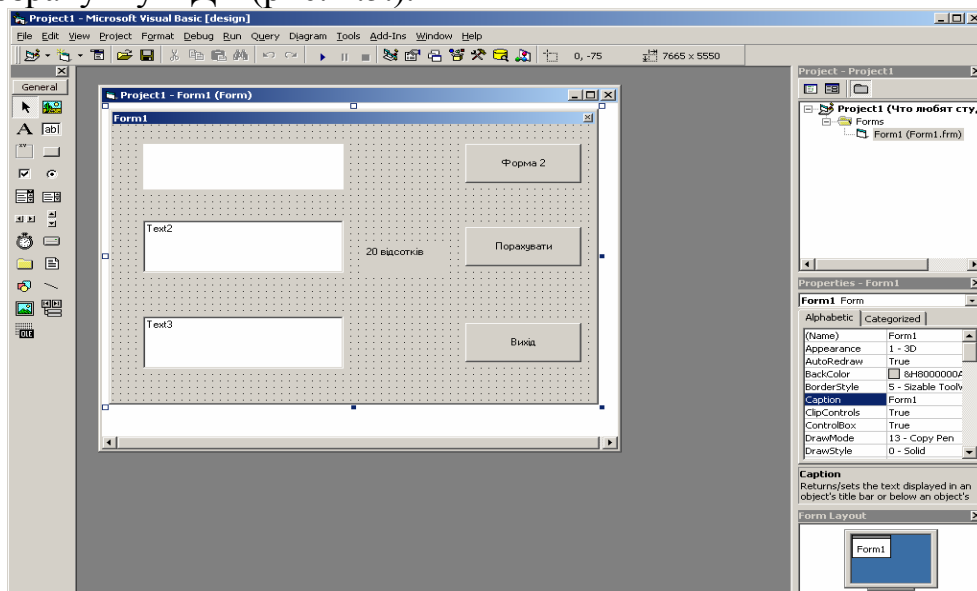


Рис. 1.5. Діалогове вікно з розрахунку ПДВ

5. Діалогове вікно з розрахунку ПДВ створено. Необхідно, щоб у ньому при введеній сумі в перше поле введення по натисканню кнопки обчислювалися 20% і 80% від вихідної суми, і ці результати виводилися в другому й третім полі введення.



Розрахунки повинні вироблятися по натисканню кнопки. Двічі натисніть кнопку. На екрані відобразиться вікно модуля Project1 - Form1(Code) для введення програми обробки події - натискання кнопки (рис. 1.6.). При подвійному натисканні на кнопку автоматично в модулі форми створюються заголовки процедури і її завершальна інструкція:

Лістинг 1.1. Перша й остання інструкції процедури обробки події натискання кнопки

```
Private Sub Command1_Click()  
End Sub
```

Залишається тільки ввести відсутній код між цими двома інструкціями.

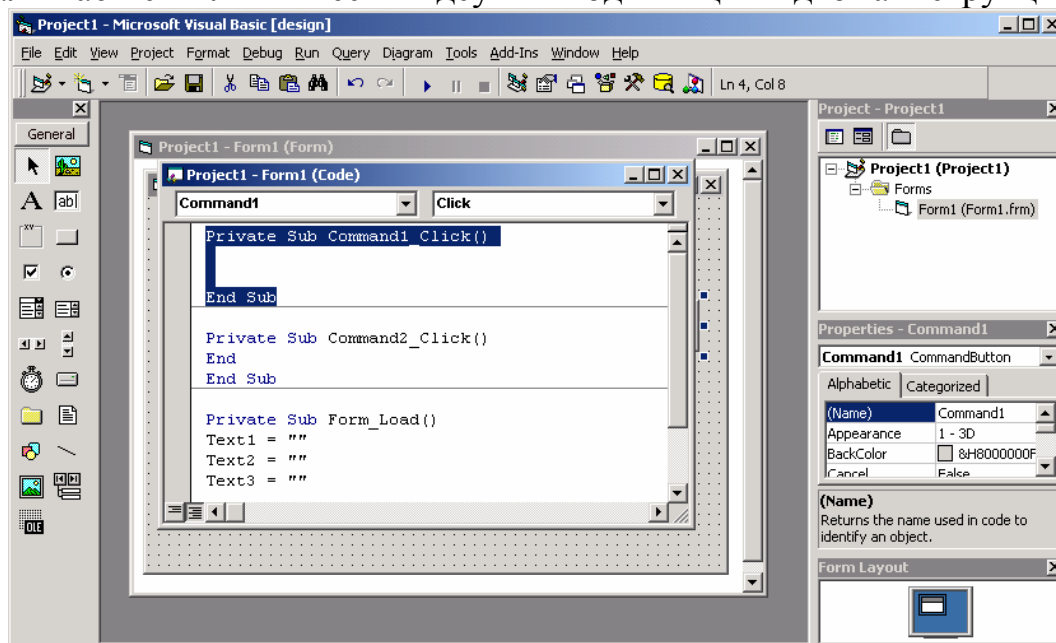


Рис. 1.6. Вікно Project1 - Form1 (Code)

6. З рис. 1.5, видно, що поля введення мають імена Text1, Text2 і Text3, це відображається на палітрі властивостей. Використовуючи це спостереження, складемо процедуру, що обчислює відповідні відсотки при натисканні кнопки:

Лістинг 1.2. Процедура розрахунку ПДВ, що виконується при натисканні кнопки Command2:

```
Private Sub Command2_Click()  
Сума = Text1  
Двадцятьвідсотків = Сума * 0.2  
Вісімдесятвідсотків = Сума * 0.8  
Text2 = Двадцятьвідсотків  
Text3 = Вісімдесятвідсотків  
End Sub
```

7. Судячи з рис. 1.5., при ініціалізації діалогового вікна в полях введення будуть відображатися рядки Text1, Text2 і Text3, що зажадає зайвих дій при роботі з додатком. Для того, щоб при ініціалізації вікна ці поля були порожні, у програму необхідно додати процедуру, що при завантаженні вікна, очищає поля введення (хоча також можна змінити властивість об'єкта caption). Для цього активізуйте вікно Project 1 - Form1 (Form) і двічі натисніть на самій формі, не по кнопці. В результаті в модулі форми з'являться інструкції початку

й кінця процедури, що обробляє подію, яка виникає при завантаженні форми в пам'ять:

Лістинг 1.3. Перша й остання інструкції процедури обробки події завантаження форми в пам'ять

```
Private Sub Form_Load()  
End Sub
```

Щоб при завантаженні форми в пам'ять відбувалося очищення полів, тобто в них вводилися порожні рядки символів, процедура, що здійснює ці дії, повинна мати наступний вигляд:

Лістинг 1.4. Процедура обробки події завантаження форми в пам'ять

```
Private Sub Form_Load()  
Text1 = ""  
Text2 = ""  
Text3 = ""  
End Sub
```

**8.** Як видно з рис. 1.1., на формі розташовані Label, що служать користувачеві підказкою, яку інформацію містить у собі кожне з вікон введення. Для того щоб призначити назви Label і призначити формі назву «Розрахунок процентів», необхідно в процедуру обробки події завантаження форми в пам'ять при розрахунку суми додати відповідий код.

Лістинг 1.4. Процедура обробки події завантаження форми в пам'ять при розрахунку суми

```
Private Sub Form_Load()  
Caption = "Розрахунок процентів"  
Label1.Caption = "20 відсотків"  
Text1 = ""  
Text2 = ""  
Text3 = ""  
Command2.Caption = "Обчислити"  
End Sub
```

**9.** Для того, щоб при натисканні на кнопку «Форма2» відкривалася відповідна форма можна використовувати властивість форми видимість (Visible), для відображення форми 2 використовувати Form2.Visible = True, для приховання форми - Form2.Visible = False. При цьому форма як і раніше залишається в оперативній пам'яті.

Лістинг 1.5. Процедура обробки події при натисканні кнопки

```
Private Sub Command1_Click()  
Form2.Visible = True  
End Sub
```

**10.** За допомогою згаданих вище елементів управління сконструюємо діалогове вікно за розрахунками суми двох чисел (Форма 2). У діалоговому вікні повинні розташовуватися три поля введення куди вводиться перше число, друге число і шуканий результат. Обчислення проводяться по натисканню кнопки, після набору в поля введення вихідних даних. Вихід з форми здійснюється також при натисканні відповідної кнопки. (рис. 1.2.)

**11.** Необхідно, щоб при ініціалізації вікна поля введення були порожні, тобто в нашу програму необхідно додати процедуру, що при завантаженні вікна в пам'ять, очищає поля введення.

**12.** Для того щоб призначити назви Label і призначити формі назву «Сума двох чисел», необхідно в процедуру обробки події завантаження форми в пам'ять при розрахунку суми додати відповідний код.

Лістинг 1.6. Процедура обробки події завантаження форми в пам'ять при розрахунку суми

```
Private Sub Form_Load()  
Caption = «Сума двох чисел»  
Label1.Caption = «Перший доданок»  
Label2.Caption = «Другий доданок»  
Label3.Caption = «Сума»  
Text1 = ""  
Text2 = ""  
Text3 = ""  
Command1.Caption = "ОК"  
Command2.Caption = "Вихід"  
End Sub
```

**13.** При введенні першого числа в перше поле й при введенні другого числа в друге поле введення, по натисканню кнопки повинна обчислюватися сума цих чисел, і результати виводилися в третім полі введення.

Код процедури обробки події матиме нижче приведений вигляд.

Лістинг 1.5. Процедура розрахунку суми, що виконується при натисканні кнопки:

```
Private Sub Command1_Click()  
Dim Перший_доданок As Double  
Dim Другий_доданок As Double  
Dim Сума As Double  
Перший_доданок = CDb1 (Text1.Text)  
Другий_доданок = CDb1 (Text2.Text)  
Сума = Перший_доданок + Другий_доданок  
Text3.Text = CStr (Сума)  
End Sub
```

**14.** Скриття форми відбувається при натисканні кнопки, для цього необхідно ввести код процедури обробки події. Для скриття форми використовується властивість форми Visible

Лістинг 1.7. Процедура обробки події натискання кнопки при скриванні форми

```
Private Sub Command2_Click()  
Form2.Visible = False  
End Sub
```

**15.** Закриття додатка здійснюється за рахунок використання функції End.

Лістинг 1.8. Процедура обробки події натискання кнопки при закритті додатка

```
Private Sub Command3_Click()
```

```
End
```

```
End Sub
```

15. Запустіть програму:

```
Run/Start
```

16. Введіть дані в поля вводу і переконаєтеся в правильності роботи програми.

17. Запишіть проект на диск і створіть виконуючий файл. Проект запишіть на диск за допомогою команди File, Save Project As по ім'ю Робота1. В результаті на диску будуть створені три файли form1.frm, form2.frm і Робота1.vbp. Для створення здійсненого файлу Робота1, виконайте команду File, Make Робота1.exe.

## Лабораторна робота № 2

**Тема:** Розробка процедур, що запобігають появі помилок. Робота з конструкціями керування

**Ціль:** Вивчити оператори If, Select і навчитися створювати програми з використанням компонента MsgBox

### Теоретичні відомості щодо ключових питань завдання

#### Конструкції керування

Існує кілька різновидів даної структури. Якщо при виконанні якої-небудь умови необхідний виконувати один оператор, то потрібно використовувати конструкцію наступного виду:

*If умова Then оператор*

У тому випадку, коли результатом перевірки умови є значення True (Істина), то виконується *оператор*, що перебуває після службового слова Then. З іншого боку, якщо після перевірки умови було отримане значення False (Неправда), то виконається наступний оператор. Всі описані параметри даної структури повинні бути зазначені в одному рядку.

Якщо при виконанні умови потрібно виконати не один, а декілька операторів, то варто використовувати таку конструкцію:

*If умова Then*

*Оператори*

*End If*

У випадку істинності умови, що перевіряється, будуть виконані *оператори*, розташовані після ключового слова Then.

З іншого боку, якщо умова є помилковою, то виконується наступний після даної конструкції оператор. У тому випадку, коли в блоці *оператори* перебуває тільки один оператор, то дана структура однаково повинна закінчуватися службовим словосполученням End If.

При необхідності виконання того або іншого оператора (або блоку операторів) у залежності від результату перевірки певної умови, у мові Visual Basic варто використовувати таку конструкцію:

*If умова Then*

*оператори1*

*Else*

```

оператори2
End If
або
If умова Then
оператори1
Else: оператори2
End If

```

Якщо результатом перевірки *умови* є значення True, то буде виконаний блок *оператори1*, що перебуває після ключового слова Then. З іншого боку, якщо перевірка *умови* дала результат False, то буде виконаний блок *оператори2*, розташований після службового слова Else.

У другому з наведених варіантів в якості блоку *оператори2* може використовуватися як один оператор (тоді він записується після знака «:» у тому ж рядку, що й службове слово **Else**), так і декілька (при цьому кожний оператор, починаючи із другого, записується в окремому рядку).

У тому випадку, коли певна дія (або набір дій) потрібно виконувати після перевірки не одного, а декількох умов, мовою Visual Basic варто використовувати таку керуючу структуру:

```

If умова1 Then
оператори1
ElseIf умова2 Then
оператори2
[Else
оператори]
End If

```

Якщо *умова1*, що перебуває після ключового слова If, істинно, то виконується блок *оператори1*, розташований після Then. Якщо умова помилкова, то здійснюється перевірка *умови2*, що перебуває після службового слова ElseIf, у випадку його істинності виконується блок *оператори2* і т.д. Якщо жодне із цих умов не є істиною, тобто результатом всіх перевірок є значення False, то виконується блок оператори, розташований після ключового слова Else (даний блок є необов'язковим).

На додаток до наведеного вище структури If...Then варто також розглянути функцію If, що повертає одне із двох значень, в залежності від умови, що перевіряється. Синтаксис даної функції має такий вид:

```

If (умова, значення1, значення2)

```

У тому випадку, коли результатом перевірки *умови* є значення True, функція повертає *значення1*, а коли перевірка дає значення False, то повертається результат, - *значення2*. Наприклад:

```

Dim int As Integer, str As String
int = 6
str = If(int Mod 2 = 0, "Парне", "Непарне")

```

Якщо число *int* ділиться на 2 без залишку, то рядку *str* буде привласнене значення "Парне", у протилежному випадку - "Непарне".

Коли існує кілька операторів (або блоків операторів), які необхідно виконувати у випадку істинності тієї або іншої умови, то запис конструкції **If ...Then** виявиться досить громіздким. Тому в подібних випадках варто використовувати структуру **Select Case**, що поліпшує читаємість програми. Її конструкція виглядає наступним чином:

```
Select Case змінна
Case значення 1
Оператори 1
Case значення 2
Оператори 2
[Case Else
оператори]
End Select
```

### *ASCII-код*

Кожному символу привласнюється свій ASCII-Код. Наприклад рядкова латинська буква *a* має код 97, а заголовна латинська *A* - 65. Усього кодів 255: 0...31 - це керуючі коди, такі, як табуляція, повернення каретки й т.п.; 32...127 - це символи букв, цифр і знаків у тому числі препинання; 128...255 - це символи псевдографіки, іноземні алфавіти (розширена таблиця символів). При натисканні клавіш на клавіатурі генерується ASCII-Код того, що намальовано на кнопках.

Щоб довідатися конкретний ASCII-Код наявного в нас символу, використовується функція `Asc ("символ")`, щоб навпаки, довідатися, який символ відповідає конкретному коду, є зворотна функція `Chr (номер коду)`

Запобігання помилок введення й контроль значень, що вводяться

При складанні додатків важливо передбачити, щоб програма аналізувала можливі помилки, що виникають при її виконанні з вини користувача, і інформувала його про це, підказуючи користувачеві, що конкретно він зробив неправильно. При цьому можливо два підходи.

◆**Запобігаючий** появи помилки. Програмно аналізуються данні, що вводяться або обчислюються і у випадку, якщо вони можуть призвести до помилки, забезпечується, щоб програма інформувала користувача в необхідності коректного завдання даних.

◆**Оброблюючий** помилки. У випадку появи помилки, вона перехоплюється, обробляється і створюється програмний відгук на виниклу помилку.

Запобігання помилки досягається в програмі перевіркою коректності введених даних. Дані спочатку вводяться, а потім проводиться перевірка коректності їхнього введення.

Контроль значень, що вводяться, можна провадити безпосередньо на етапі введення, відсіюючи значення, що вводяться неправильно. Цього можна досягти, додавши в програму код обробки події `KeyPress`.

Наприклад, якщо по постановці завдання чисельником може бути тільки ціле ненегативне число, то в код програми можна додати наступну процедуру, що забезпечує введення тільки цифр.

```

Privat Sub Text1_KeyPress(KeyAscii As Integer)
    Dim Цілечисло As String
    Цілечисло = "0123456789"
    If KeyAscii > 26 Then
        If InStr(Цілечисло, Chr(KeyAscii)) = 0 Then
            KeyAscii = 0
        End If
    End If
End Sub

```

У процедурі в першу чергу перевіряється, чи був код натиснутої клавіші більше 26. Це найбільший керуючий код Visual Basic. Керуючий код вказує на те, що натиснуто не звичайну клавішу клавіатури, а керуючу. Така, як <Esc>. <Del>? <BackSpace> і т.д. Після цього, процедура перевіряє, чи була натиснута одна із цифрових клавіш. Якщо не була натиснута цифрова клавіша, то параметру KeyAscii привласнюється нульове значення, що призводить до ігнорування натискання клавіші. Таким чином, даний код забезпечує введення із клавіатури в поля введення тільки цифр і редагування введеного числа стандартними засобами за допомогою керуючих клавіш.

#### Відображення діалогового вікна MsgBox

Щоб програма вела діалог з користувачем існують діалогові вікна різного змісту. Виводяться вони на екран функцією MsgBox. Входять у неї такі аргументи:

MsgBox Prompt [, Buttons] [, Title] [, Helpfile, Context]

MsgBox ("повідомлення", число й тип кнопок, "заголовок вікна", "ім'я файлу довідки якщо є", номер розділу довідки, якщо є ім'я файлу)

Два останніх аргументи необов'язкові. Вони використовуються, якщо в тебе створений файл допомоги (Help).

"повідомлення" - це повідомлення користувачеві (максимальна довжина приблизно 1024 символу)

"заголовок вікна" - це те, що виводиться у верхній смужці (заголовку) вікна

Число виходить зі складання трьох чисел або констант (кому як зручніше):

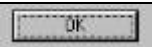





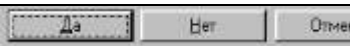



- кількість і тип кнопок
- вид повідомлення (який значок буде у вікні)
- яка кнопка є основною (за замовчуванням - перша)

Основна кнопка повинна бути оголошена змінна. В неї записується число, що позначає, яку ж кнопку нажав користувач, щоб ми могли змусити нашу програму як-небудь зреагувати. А числа (замість них теж можна писати константи) ці відповідають наступним кнопкам:

Таблиця 1

Константа	Число	Яка кнопка натиснута
vbOK	1	ОК
vbCancel	2	Скасування (Cancel)
vbAbort	3	Перервати (Abort)
vbRetry	4	Повторить (Retry)
vbIgnore	5	Пропустить (Ignore)
vbYes	6	Так (Yes)
vbNo	7	Немає (No)

Таблиця 2

Тип кнопок			Вид повідомлення			Основна кнопка		
константа	або число	виведені кнопки	константа	або число	значок сообще- ния	константа	або число	номер основ- ний кнопки
vbOKOnly	0		VbCritical	16		VbDefaultButton1	0	перша
VbOKCancel	1		VbQuestion	32		VbDefaultButton2	256	друга
VbAbortRetryIgnore	2		VbExclamation	48		VbDefaultButton3	516	третя
VbYesNoCancel	3		VbInformation	64		VbDefaultButton4	768	четверта
VbYesNo	4					VbApplicationModal (на рівні додатка)	0	модальне вікно
VbRetryCancel	5					VbSystemModal (на рівні системи)	4096	(не згорнеться, поки на кнопку не натиснеш)

Таким чином, програма виводу вікна підтвердження при натисканні на кнопку закриття додатка може виглядати в такий спосіб:

```
Dim ret
```

```
ret = MsgBox("Чи впевнені ви, що хочете закрити додаток?", vbYesNo, _  
"Підтвердіть закриття")
```

```
If ret = 6 Then End
```

### Завдання

Скласти програму виконуючу розподіл чисел з перевіркою коректності вхідних даних використовуючи підходи запобігання й обробки помилок.

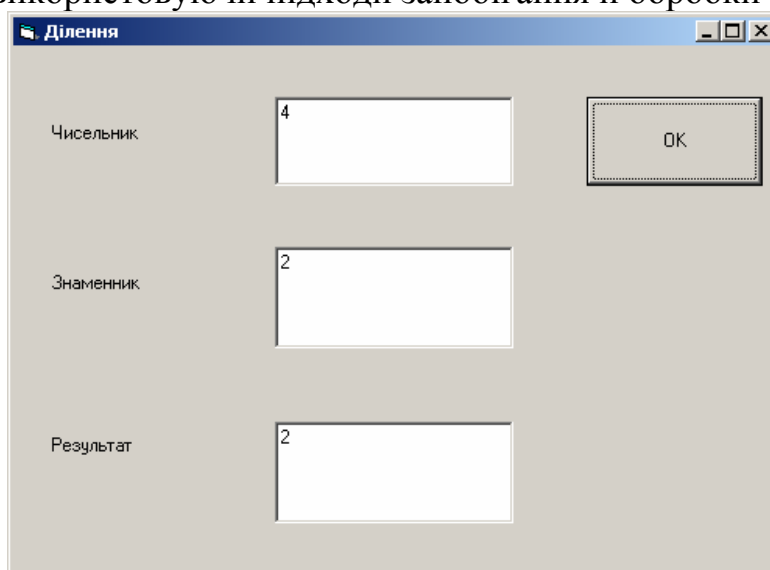


Рис. 2.1. Діалогове вікно з розрахунку остатку від ділення числа



## Хід роботи

1. Створюємо діалогове вікно форми (мал.1) .

2. Пишемо програмний код у вікні кодів.

У процедурі завантаження форми змінюємо заголовок форми. В тілі процедури пишемо:

```
Form1.Caption = "Ділення"
```

Змінюємо записи в керуючих елементах label і TextBox. Для введення будь-якої інформації в поле Text, можна використовувати властивість .Text. В Label аналогічна властивість .Caption.

Привласнимо TextBox значення порожнього рядка:

Лістинг 1.1 Процедура обробки події завантаження форми в пам'ять

```
Text1.Text = ""
```

```
Text2.Text = ""
```

```
Text3.Text = ""
```

Лейблам привласнимо відповідні значення:

```
Label1.Caption = "Чисельник"
```

```
Label2.Caption = "Знаменник"
```

```
Label3.Caption = "Результат:"
```

Для кнопки привласнимо значення Command1.Caption = "ОК"

3. Для кнопки пишемо підпрограму (процедуру) обробки дії.

В поле (General) вибираємо Command1. У нас з'явилася нова процедура для кнопки "ОК". Вона дуже схожа на процедуру форми, тільки подія тут інше - натискання кнопки мишею (Click). Пишемо рядок для обробки операції розподілу.

Лістинг 1.2 Процедура обробки події натискання кнопки

```
Private Sub Command1_Click ()
```

```
Dim Чисельник As Double
```

```
Dim Знаменник As Double
```

```
Dim Результат As Double
```

```
Чисельник = CDb1 (Text1.Text)
```

```
Знаменник = CDb1 (Text2.Text)
```

```
Результат = Чисельник / Знаменник
```

```
Text 3.Text = CStr (Результат)
```

```
End Sub
```

Незважаючи на те, що розглянута ситуація дуже проста, вже вона може містити в себе безліч помилок. Наприклад, якщо користувач забуде ввести в поле чисельника або знаменника значення, при натисканні кнопки відбувається аварійне переривання виконання програми з мало зрозумілим повідомленням про невідповідність типів.

Дане повідомлення про помилку пов'язане з однієї з наступних двох інструкцій у програмі:

```
Чисельник = CDb1(Text1.Text)
```

```
Знаменник = CDb1(Text2.Text)
```

де параметром функції CDb1 повинні бути строка, трансформована в число. Але порожній рядок не може бути перетворений в число, і тому через

функцію CDb1 відбувається помилка. Помилка в невідповідності типів виникне також, якщо в одне з полів користувач введе число з десяткової коми, а установками системи передбачається десяткова крапка і навпаки.

Щоб уникнути помилки необхідно передбачити в програмі попередню перевірку: чи перетворюються дані, що вводяться, у числа і якщо вводиться інформація, що, не перетвориться в числа, то видається повідомлення про поле, у якому некоректно введені дані, на цьому полі встановлюється фокус.

4. Для кнопки пишемо підпрограму (процедуру) обробки дії, що передбачає перевірку коректності введення даних.

Програма процедури буде виглядати в такий спосіб:

Лістинг 1.3 Процедура обробки події натискання кнопки

```
Private Sub Command1_Click ()
    Dim Чисельник As Double
    Dim Знаменник As Double
    Dim Результат As Double
    If IsNumeric (Text1.Text) = False Then
        MsgBox «Помилка в чисельнику», VbInformation «Ділення»
        Text1.SetFocus
        Exit Sub
    End If
    If IsNumeric (Text2.Text) = False Then
        MsgBox «Помилка в знаменнику», VbInformation «Ділення»
        Text1.SetFocus
        Exit Sub
    End If
    Чисельник = CDb1 (Text1.Text)
    Знаменник = CDb1 (Text2.Text)
    Результат = Чисельник / Знаменник
    Text 3.Text = CStr (Результат)
End Sub
```

5. Наведеною вище програмою передбачені не всі можливі помилки які допускаються користувачем. Якщо в знаменник ввести 0, то відбудеться аварійна зупинка програми. Щоб уникнути цієї помилки, будемо так само перевіряти, чи не є введені в поле дані нулем. Це можна зробити додавши перед розрахунковим блоком у процедурі наступну додаткову перевірку:

Лістинг 1.4 Розподіл чисел з перевіркою, чи не дорівнює знаменник нулю

```
If CDb1 (Text2.Text) = 0 Then
    MsgBox «Знаменник не може бути нулем», _
        VbInformation, «Ділення»
    Text2.SetFocus
    Exit Sub
End If
```

6. Щоб забезпечити введення із клавіатури в поле введення тільки цифри й забезпечити редагування числа стандартними методами за допомогою керуючих клавіш, необхідно ввести наступний код у тіло процедури обробки події:

```

Dim Знак As String
Число = «0123456789,+»
Знак = «+-»
If KeyAscii > 26 Then
    If InStr (Число, Chr (KeyAscii)) = 0 Then
        KeyAscii = 0
    End If
    If InStr (Text 1.Text, ",") > 0 And Chr (KeyAscii) = "," Then
        KeyAscii = 0
    End If
    If Text1.SelStart > 0 And Instr (Знак, Chr (KeyAscii) ) > 0 Then
        KeyAscii = 0
    End If
End If
End Sub

```

Логіка процедури:

- ♦ перевіряється, чи не є символ, що набирається, цифрою, знаком, або комою. Якщо не є, то набір символу ігнорується;
- ♦ якщо набирається символ, що є цифрою, то він вводиться в поле введення;
- ♦ якщо набирається символ, що є комою, то перевіряється, чи є вона першою вводимою комою. Якщо раніше коми не вводилися, вона вводиться в поле введення. У протилежному випадку введення коми ігнорується;
- ♦ якщо набирається символ, що є знаком, то перевіряється, у якій позиції числа він вводиться. Якщо в першій позиції, то знак вводиться. У протилежному випадку введення знака блокується.

7. Введіть дані в поля введення й переконайтеся в правильності роботи програми.

8. Запишіть проект на диск і створіть виконуючий файл ЛР№2.exe.

### **Лабораторна робота №3**

**Тема:** *Робота зі строковими змінними. Малювання в Visual Basic. Поняття координатної системи. Застосування методу Circle*

**Ціль:** *Вивчити роботу зі строковими значеннями. Навчитися малювати фігури програмним методом*

#### **Теоретичні відомості щодо ключових питань завдання**

##### *Додавання строк*

Оскільки інформація в полях введення Visual Basic завжди зберігається в текстовому форматі, робота зі строками представляється тут більше важливою, чим у звичайному BASIC. Для того щоб скласти дві строки разом (т.зв. конкатенація), можна використовувати оператори & або +. Наприклад:

```

Titles = "Queen "
Name$ = "Elizabeth "
Numeral$ = "I"
Title$ & Name$ & Numeral$ = "Queen Elizabeth I"
Title$ 6 Name$ & Numeral$ & Numeral$ = "Queen Elizabeth II"

```

Оператор & поєднує строки в тім порядку, як вони представлені. Тому, на відміну від підсумовування чисел, для злиття строк важливий їхній порядок проходження. За допомогою & можна скласти скільки завгодно строкових значень. Нижче наведений приклад з використанням уже оголошених вище змінних:

CurrentQueen\$ = Title\$ & Name\$ & Numeral\$ & Numeral\$

Основною відмінністю оператора & від + є можливість використання & для об'єднання строкових даних із другими їхніми типами. Наприклад, вираження Z=A% & B\$ поєднує цілочисельну і строкову змінні, змінюючи тип їхніх значень на variant.

### *Формат ASCII/ANSI*

У комп'ютера немає окремої ділянки пам'яті для зберігання тексту, а ще одного - для чисел. Усе, що надходить в пам'ять комп'ютера, перетворюється в числовий формат (реально - у двійкове подання). У програмі позначається тільки, дана ділянка оперативної пам'яті яка містить закодований текст. Звичайно формат для перетворення текстової інформації в цифрову називається ASCII (American Standard Code for Information Interchange). Даний формат привласнює кожному символу відповідне число в діапазоні від 0 до 255, хоча Windows не може вивести на екран всі 255 символів і використовує більше обмежений їх набір ANSI (American National Standards Institute). Керуючі символи, а також спеціальні типи табуляції або переводу строки мають номери до 32. Значенням функції Chr(n) є символ, що відповідає числу пі з формату ASCII.

Оператор Print Chr(n) або виводить на екран символ, що відповідає коду ASCII, шрифтом який використовується, або має місце особливий ефект у залежності від типу керуючого символу. Наприклад, оператор Print Chr(227) виводить грецьку букву "пі" на екрані, якщо перед цим значення FontType було встановлено як MS LineDraw за допомогою вікна Properties або через Code. Наступний приклад використовує код символу лапок ("), 34, для виводу на екран фрази, зазначені в лапки по обидва боки: Print Chr (34). Print "Quoth the raven, nevermore."

### *Print Chr (34)*

Функція Chr повертає строку, збережену в типі variant. Аналогічна більше стара функція Chr повертає безпосередньо строкове значення. На екрані буде виведене "Quoth the raven, nevermore."

*Примітка:* Попередній результат можна одержати за допомогою оператора Print ""Quoth the raven, nevermore.""; тому що Visual Basic, на відміну від багатьох версій BASIC, сприймає два символи лапок як один і виводить його на екран у вираженнях з оператором Print або при роботі зі строковими значеннями.

### *Перехід на новий рядок*

У ранніх версіях Visual Basic одним з основних способів використання функції Chr є формування керуючих символів для переходу до нового рядка в програмах. Перехід до нового рядка використовується при роботі із багатостроковими полями введення або при додаванні інформації в

інформаційній панелі. Як і в старих друкарських машинках, для переходу на новий рядок необхідно проробити дві операції: переклад каретки (carriage return) для повернення до першого символу рядка, а потім переклад рядка для переходу на наступний рядок. При використанні функції Chr перехід на новий рядок виглядає так: vbCrLf = Chr (13) + Chr (10)

Але тепер є можливість використовувати вбудовану константу vbCrLf.

Наприклад, необхідно розірвати рядок в інформаційній панелі або в багатостроковому полі введення. Швидше всього це реалізувати з використанням vbCrLf:

```
TextString$ = "Visual Basic For Windows" + vbCrLf
TextString$ = TextString$ + "Osborne McGraw-Hill" + vbCrLf
TextString$ = TextString$ + "Berkeley, CA"
Text1.Text = TextString$
```

Креслення фігур програмними методами. Метод Circle.

Для креслення на об'єкті окружності, еліпса або дуги застосовується метод Circle. Його синтаксис: Circle [Step] (x, y), радіус, [color, start, end, aspect],

де

об'єкт - це об'єкт на якому креслиться фігура. Якщо об'єкт не заданий, то використовується об'єкт на якому в цей момент установлений фокус;

step - необов'язкове ключове слово, встановлює, що центр окружності задається не по абсолютних координатах x, y, а щодо поточних координат, обумовлених значеннями властивостей Current і Current;

x, y - координати центру окружності (одиниці виміру визначаються властивістю ScaleMode);

радіус - радіус окружності, еліпса або дуги (одиниці виміру визначаються властивістю ScaleMode);

color - визначає колір окружності, еліпса або дуги. Може задаватися як RGB - у форматі шістнадцятиричного числа VB (&HFF0033), або кольором, обумовленим ключовими словами Visual Basic (vbRed). Можна використовувати функції RGB або QBColor. Якщо не заданий, колір установлюється той, що задано властивістю .ForeColor;

start, end - при створенні дуги задають початкове і кінцеве значення кута дуги в радіанах (від -2π до 2π);

aspect - коефіцієнт стиску еліпса. За замовчуванням дорівнює 1,0, тобто стиску немає, виводиться окружність.

*Приклад:*

```
Private Sub Form_Load()
```

```
Picture1.ScaleMode = vbPixels - задаємо одиниці виміру пікселі
```

```
Picture1.AutoRedraw = True - встановлюємо видимість малювання
```

```
Picture1.DrawWidth = 3 - ширину лінії встановлюємо в 3 пікселя
```

```
Picture1.ForeColor = &HFF00FF - ставимо колір у властивості ForeColor у
```

фіолетовий

```
Picture1.Circle (200, 120), 70 - центр окружності перебуває в крапці X=200, Y=120, діаметр - 70 пікселей
```

```
End Sub
```

## Завдання:

Написати додаток, що складається з п'яти форм, виклики різних форм здійснюється з першої форми за допомогою керуючих елементів, а саме перемикачів і командних кнопок. Всі інші форми мають кнопки закриття форми й закриття самого додатка. Зовнішній вигляд першої форми наведений на рисунку 1.

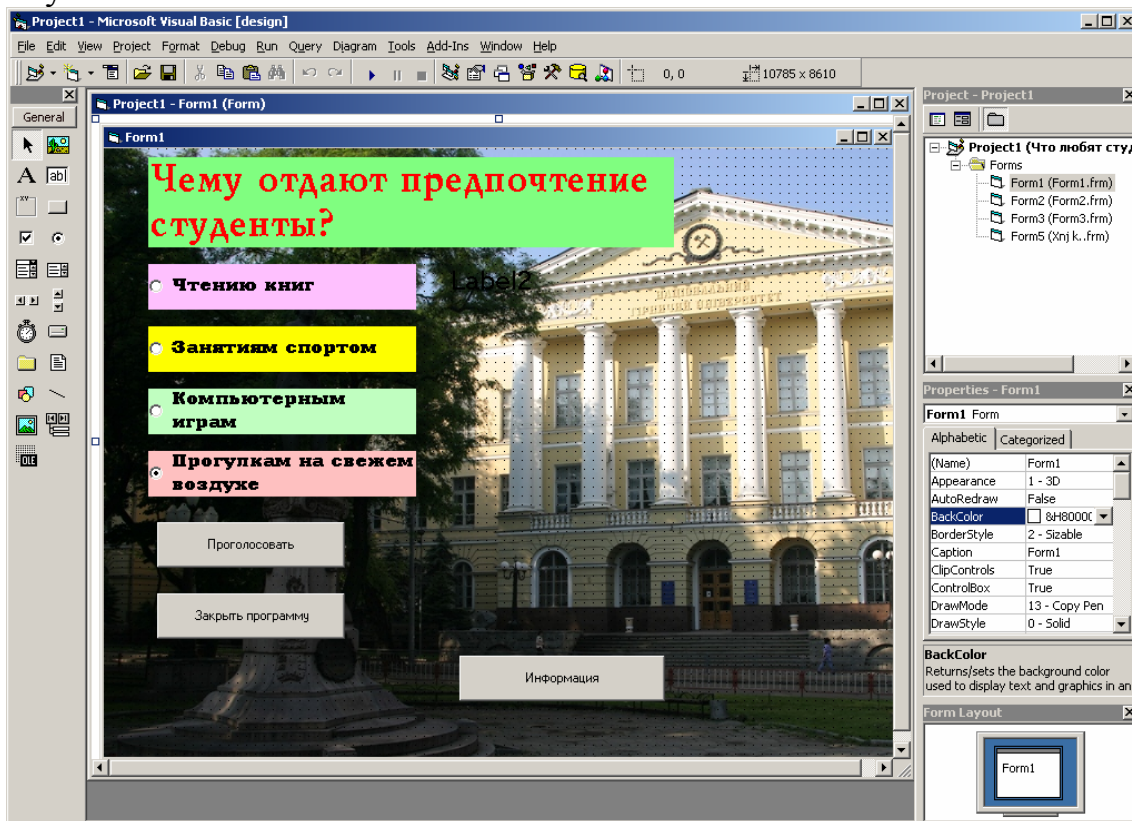


Рис. 3.1. Форма 1. «Статистичні дані»

У формі 1:

- при виборі одного з перемикачів (Option Button) і натисканні на кнопку «Проголосувати» відкривається відповідна форма;
- при натисканні на кнопку «Закрити програму» відбувається аварійне завершення програми, за допомогою оператора End;
- при натисканні на кнопку «Інформація» у написі 2 (Label2) відображається інформація про індивідуальне завдання, групу і студента, що виконує роботу, а також напис на кнопці (властивість кнопки Caption) змінюється на напис «Приховати інформацію». При повторному натисканні на кнопку дані в керуючому елементі Label2 зникають, і напис на кнопці приймає первісне значення «Інформація».

Інші форми містять дві обов'язкових кнопки - закриття форми, що здійснюється за рахунок установки властивості даної форми `Visible=False`, і кнопки закриття додатка, при натисканні на яку відбувається відкриття модального діалогового вікна уточнюючого подальші дії.

Перед завантаженням форми 2 відбувається запит про ім'я користувача.

Форма 3 містить відображення в текстовому полі віршика. Для переходу на іншу строку використовується константа Visual Basic `vbCrLf`. Для з'єднання

текстових даних використовується як символ амперсанда, так і знак плюс (&, +).

Форма 4 містить дві обов'язкові кнопки.

Форма 5 містить чотири кнопки, дві з яких працюють із графічними об'єктами що створюються в додатку. При натисканні на кнопку «Створити діаграму» на формі кресляться сегменти кола з різним заливанням. При натисканні ж на кнопку «Убрати діаграму» форма очищується від всіх графічних елементів за допомогою оператора CLS.

#### **Хід роботи:**

1. Створюємо новий Exe-Проект.

Відкриється вікно форми Проект n - Form1 (Form).

2. Створюємо командні кнопки, лейбли і текстбокси (рис. 3.1.)

3. Змінюємо властивості форми і її компонентів таким чином, щоб зовнішній вигляд Form1 збігався з видом форми 1 наведеної на рисунку 3.1.

4. Створюємо форми 2, 3, 4, 5 з відповідними компонентами.

5. Створюємо вікно кодів для Form1. Для кожного компонента пишемо підпрограму (процедуру) обробки відповідної дії.

Для того, щоб при виборі одного з перемикачів (Option Button) і натисканні на кнопку «Проголосувати» відкривалася відповідна форма можна використовувати властивість форми видимість (Visible), наприклад для відображення форми 1 необхідно вказати Form1.Visible = True, для приховання форми використовувати Form1.Visible = False. При цьому форма як і раніше залишається в оперативній пам'яті. Це здійснюється за рахунок використання керуючої структури If...Then...

*Алгоритм написання:* якщо обрано певний перемикач, то відбувається відображення відповідної форми, властивість вимикача активна (Value) може бути також True або False.

Для того, щоб при натисканні на кнопку «Інформація» у написі2 (Label2) відображалася інформація про індивідуальне завдання, групу і студента, який виконує завдання, а також напис на кнопці (властивість кнопки Caption) мінялася на напис «Приховати інформацію», а при повторному натисканні на кнопку дані в керуючому елементі Label2 віддалялися й напис на кнопці приймав первісне значення «Інформація», необхідно використовувати наступні оператори:

```
If Label2.Caption = "" Then
```

```
Label2.FontSize = 14
```

```
Label2.Caption = "Індивідуальна робота 2 студенти групи АМГ ФІО"
```

```
Command3.Caption = "Приховати інформацію"
```

```
Else
```

```
Label2.Caption = ""
```

```
Command3.Caption = "Інформація "
```

```
End If
```

Прозорість тіла в написі (Label) досягається за рахунок установки властивості BackStyle=0 (transparency).

6. Form2. Визвати запит імені користувача перед завантаженням другої форми можна в такий спосіб:

```
Dim n
n = InputBox("Як ваше ім'я?")
Label1.FontSize = 20 - установка розміру шрифту 20 пунктів
Label1.Caption = "Поздоровляємо вас шановні аматори книг!" _
& " І Вас - вельмишановний і самий головний аматор літератури по імені " &
n _ & "."
```

Закриття форми, здійснюється за рахунок установки властивості даної форми Visible=False.

Для того, щоб при натисканні кнопки закриття додатка, відбувалося відкриття модального діалогового вікна уточнюючи подальші дії, використовується конструкція наступного типу:

```
MsgBox Prompt [, Buttons] [, Title].
```

Вивід вікна підтвердження при натисканні на кнопку закриття додатка може виглядати в такий спосіб:

```
Dim ret
ret = MsgBox("Чи впевнені ви що хочете закрити додаток?", vbYesNo, _
"Підтвердити закриття")
If ret = 6 Then End
```

7. Form3. Для відображення в текстовому полі вірша, необхідно використовувати нижче наведену конструкцію:

```
Dim F As String
F = " В неволі, в самоті немає!" + vbCrLf
F = F + " Нема з ким серце поєднать ..." + vbCrLf
F = F + " То сам собі оце шукаю..." + vbCrLf
F = F + " Когось-то, з ним щоб розмовлять."
Text1.Text = F
```

8. Form5. Для того, щоб при натисканні на кнопку «Створити діаграму» на формі рисувалися сегменти кола з різним заливанням, необхідно написати нижченаведений програмний код:

```
Const pi = 3.141593
Dim i As Integer; Dim ast, aend As Double
FillColor = vbBlue
ForeColor = vbBlack
DrawWidth = 10
aend = -0.001
For i = 1 To 6
FillStyle = i + 1
ast = aend
aend = -(2 * pi / 6) * i
Circle (150, 100), 70, , ast, aend
Next
```

9. Введіть дані в поля вводу й переконайтеся в правильності роботи програми.



10. Запишіть проект на диск і створіть виконуючий файл, під ім'ям Робота3.

#### **Питання для підготовки до захисту лабораторної роботи**

1. Структура програми на Visual Basic?
2. Типи перемінних?
3. Способи представлення чисел?
4. Представлення оператор присвоювання?
5. Різновиди конструкції керування?
6. Що таке ASCII-код?
7. Код обробки події контролюючої введення значень?
8. Діалогове вікна MsgBox? Призначення, основні аргументи?
9. Формат ASCII/ANSI? Призначення формату?
10. Оператори додавання рядків?
11. Метод Circle? Синтаксис методу?

#### **Вимоги до оформлення**

Роботи оформлюються відповідно вимогам зазначеним в п.3 і зберігаються в електронному вигляді на кафедрі.

#### **Оцінювання лабораторної роботи**

Поточний контроль занять лабораторного модуля здійснюється для кожної лабораторної роботи після її виконання і полягає в оцінюванні практичних навичок при виконанні завдань за темою роботи.

#### **Критерії оцінок**

**"Відмінно"** – виставляється, якщо при відповіді на питання студент виявив знання та уміння на продуктивно-синтетичному рівні (глибоке розуміння щодо навчального об'єкту, здатність здійснювати синтез, генерувати нові уявлення, переносити раніше засвоєні знання на нетипові ситуації), виконав завдання передбачене програмою за зазначеними вимогами.

**"Добре"** - виставляється, якщо при відповіді на питання студент виявив знання та уміння на понятійно-аналітичному рівні (чітке уявлення та поняття щодо навчального об'єкту, здатність здійснювати смислове виділення, пояснення, аналіз, перенесення раніше засвоєних знань на типові ситуації), виконав завдання передбачене програмою за зазначеними вимогами з незначними помилками.

**"Задовільно"** – виставляється, якщо при відповіді на питання студент виявив знання та уміння відповісти за програмним матеріалом на ознайомчо-орієнтованому рівні (орієнтоване уявлення щодо понять, які вивчаються, здатність відтворювати формулювання визначень, законів тощо, уміння вирішувати типові завдання шляхом підставлення числових даних), при виконанні завдання допустив неprincipові помилки.

**"Незадовільно"** – виставляється, якщо при відповіді на питання студент проявив володіння матеріалом на рівні окремих фрагментів, допустив принципові помилки при виконанні завдання на рівні нижче ознайомчо-орієнтованого

Результати модульного контролю з опанування четвертого модулю визначаються за результатами поточного контролю.

Результати модульного контролю проставляються у відомість у балах національної шкали (5, 4, 3, 2) незалежно від запланованої форми підсумкового контролю. За відсутністю студента на контрольному заході у відомість проставляється 2 бали (матеріал не опановано). Результати модульного контролю заносяться у відомість після закінчення кожної чверті.

Перздача модуля здійснюється за розкладом та процедурою, що визначаються кафедрою, протягом двох тижнів наступної чверті навчального року.

Перша перздача модуля здійснюється викладачу, який викладав матеріал модуля. Друга перздача здійснюється комісії у складі трьох осіб: викладача, який викладав матеріал модуля; завідуючого кафедрою; лектора або іншого викладача.

## Література

1. Малачівський П.С. Програмування в середовищі Visual Basic: Навчальний посібник. – Львів:Видавництво «Бескид Біт», 2004. – с. 6-132
2. Microsoft Visual Basic 6.0 для професиналов. Шаг за шагом: Практич. посіб./Пер.с англ. – М.: Издательство ЭКОМ, 2004. – с. 15-207
3. Visual Basic 5. Библия разработчика.: Пер. с англ. – К.:Диалектика, 1997. – с.8-231
4. <http://vbzero.narod.ru/part1.htm>
5. <http://www.biblioteka.net.ru/index.php?book=basic>

## Зміст

Передмова.....	3
Лабораторна робота №1 .....	4
Лабораторна робота №2.....	12
Лабораторна робота №3.....	19
Література.....	27
Зміст.....	27