Тема. Оператори циклу.

Навчальні питання:

1. Основні поняття. Види циклів.

2. Загальний підхід до побудови циклів.

1. Основні поняття. Види циклів.

У цій лекції ви дізнаєтесь, як багаторазово виконувати блок операторів за допомогою циклів. Для виконання операторів задане число раз ви будете використовувати цикл **For ... Next**, а для виконання операторів до тих пір, поки не виконається умовний вираз - цикл **Do**.

Ви також дізнаєтеся, як за допомогою оператора конкатенації рядків (&) виводити в об'єкті текстового поля більше одного рядка тексту.

Цикли For ... Next

За допомогою циклу **For ... Next** ви можете виконувати групу операторів програми в процедурі події або модулі коду задане число раз.

Цей підхід корисний в тому випадку, якщо ви виконуєте кілька пов'язаних обчислень, працюєте з елементами на екрані або обробляєте кілька фрагментів даних, введених користувачем.

Цикл For ... Next є насправді просто коротким способом записати довгий список операторів програми. Так як кожна група операторів в такому списку буде робити одне і те ж, Visual Basic дозволяє визначити одну таку групу операторів і виконати її стільки разів, скільки ви захочете.

Синтаксис циклу For ... Next

For змінна = початкове_значення To кінцеве_ значення

оператори, що повторюються

Next [змінна]

У цьому описі синтаксису **For, To** і **Next** - це обов'язкові ключові слова, оператор присвоєння **(=)** також обов'язковий.

Ви повинні замінити слово змінна ім'ям числової змінної, яка зберігає поточне значення лічильника циклів (змінну після Next вказувати необов'язково), **а початкове_значення** і **кінцеве_значення** **замінити** числовими значеннями, що представляють початкову і кінцеву точки циклу. (Зауважте, що ви повинні оголосити змінну до того, як станете використовувати її в операторі For ... Next.)

Рядок або рядки між операторами For і Next - це оператори, які повторюються при кожному виконанні циклу.

Наприклад, наступний цикл **For...Next** відтворює у вигляді послідовності чотири звукових сигнала:

```
Dim i As Integer
For i = 1 To 4
Beep()
Next i
```

Це еквівалентно:

```
Beep()
Beep()
Beep()
Beep()
```

2. Загальний підхід до побудови циклів.

Змінна, яка використовується в циклі - це **i**, єдина буква, яка, за угодою, визначає перший лічильник циклу For ... Next і оголошується як змінна типу Integer.

При кожному виконанні цикла змінна лічильника збільшується на одиницю. (При першому проході через цикл змінна містить значення 1 - значення, задане параметром початкове_значення; при останньому проході вона містить значення 4 - кінцеве_значення.)

Відображення змінної-лічильника в елементі управління TextBox

Змінна-лічильник аналогічна будь-який іншій змінній процедури події.

Її значення може бути присвоєно властивості, використано в обчисленнях або відображено в програмі. Одним з практичних застосувань змінної-лічильника є її відображення в елементі управління TextBox. Засобом при відображенні більш ніж одного рядка полягає в тому, що ви просто встановлюєте властивість Multiline елемента управління TextBox на значення True, а властивість ScrollBars - на значення Vertical. При використанні цих простих налаштувань об'єкт однострочного текстового поля стає об'єктом многострочного текстового поля зі смугами прокрутки для полегшення доступу до нього.

Алгоритм реалізації відображення інформації за допомогою циклу For ... Next.

1. Запустіть Visual Studio і створіть новий проект Windows Application (Додаток Windows) на Visual Basic. У Windows Forms Designer (Конструкторі Windows Forms) з'явиться порожня форма. Перш за все ви додасте в форму елемента управління Button.
2. Двічі клацніть мишею у вікні області елементів на елементі управління Button. Visual Studio помістить об'єкт кнопки у верхньому лівому куті форми. У випадку з елементом управління Button, а також і з багатьма іншими, подвійне клацання мишею дозволяє швидко створити в формі об'єкт зі стандартними розмірами. Тепер можна перетягнути цей об'єкт кнопки в потрібне місце і налаштувати його властивості.
3. Перетягніть об'єкт кнопки вправо і помістіть його в центрі верхньої частини форми.
4. Відкрийте вікно Properties (Властивості), а потім встановіть властивість Text кнопки на значення Цикл.
5. В області елементів клацніть двічі на елементі управління TextBox. Visual Studio створить невелике текстове поле.

Алгоритм реалізації відображення інформації за допомогою циклу For ... Next.

6. Встановіть властивість Multiline цього об'єкта текстового поля на значення True, а властивість ScrollBars на значення Vertical. Ці настройки готують текстове поле до відображення більш ніж одного рядка тексту.
7. Зробіть властивість Text об'єкта текстового поля порожнім (empty).
8. Перемістіть текстове поле під кнопку і збільште його так, щоб воно зайняло більшу частину форми.
9. Клацніть двічі на кнопці Цикл форми. У редакторі коду з'явиться процедура події Button1_Click.
10. Введіть у цій процедурі наступні оператори програми:

```
Dim i As Integer
Dim Wrap As String
Wrap = Chr (13) & Chr (10)
For i = 1 To 10
TextBox1.Text = TextBox1.Text & "Рядок" & i & Wrap
Next i
```

Алгоритм реалізації відображення інформації за допомогою циклу For ... Next.

Ця процедура події оголошує дві змінні - одну типу Integer (i), а другу - типу String (Wrap), а потім привласнює другій змінній строкове значення, що представляє символ переходу рядка.

У термінах програмування символ переходу рядка є еквівалентом натискання на клавішу (Enter) на клавіатурі. Щоб зробити його менш громіздким,створимо для цього символу спеціальну змінну, яка складається з елементів "повернення каретки" (return) і "прокрутка рядка" (linefeed).

Після оголошення змінної і присвоєння, використовуємо цикл For ... Next для десятикратного відображення в об'єкті текстового поля рядка "Рядок X", де X - це поточне значення змінної-лічильника (іншими словами, з "Рядок 1" до "Рядок 10"). Символи конкатенації рядків (&) об'єднують в текстовому полі частини кожного рядка воєдино.

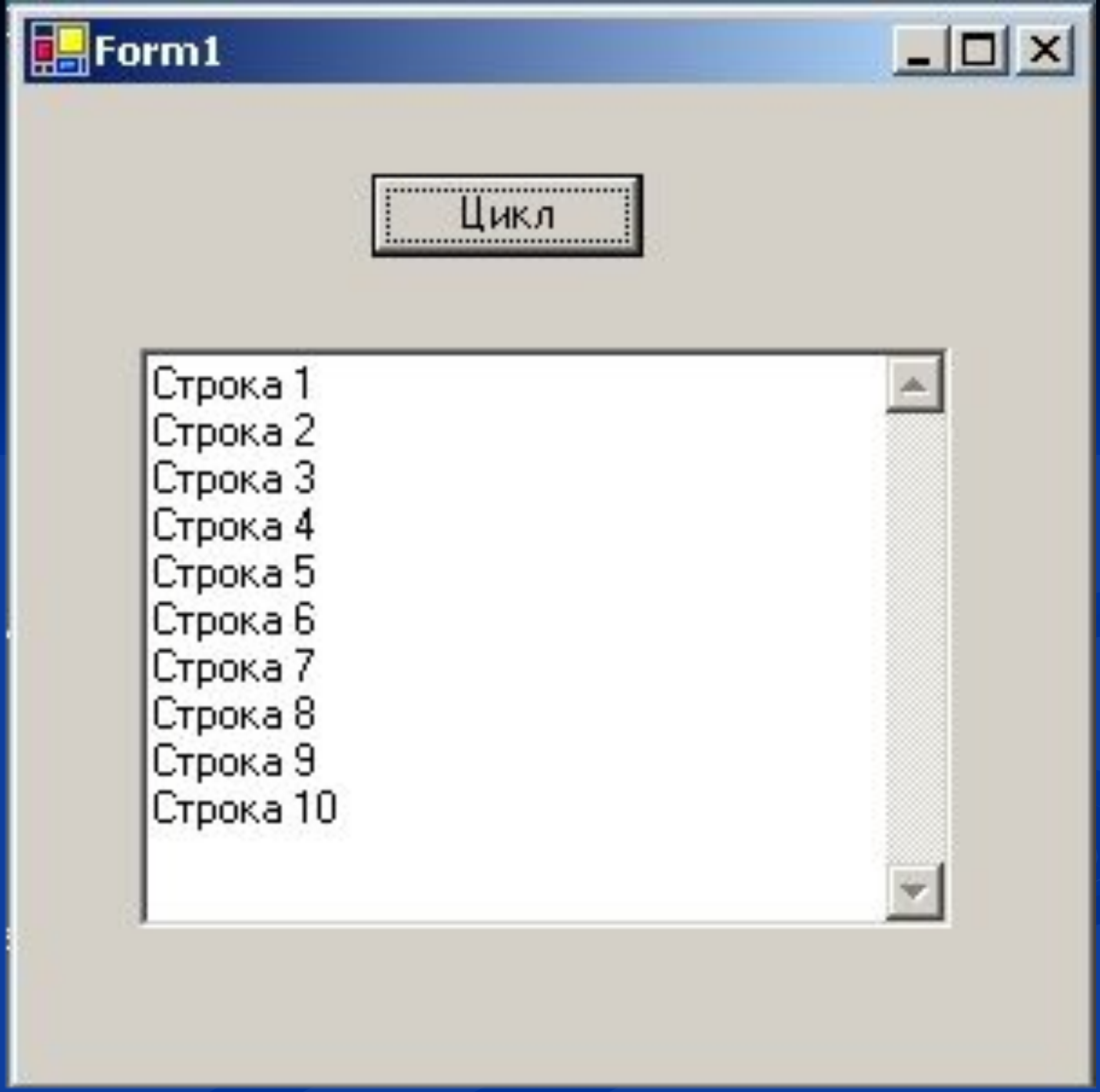
На початку в об'єкт додається все значення текстового поля, яке зберігається у властивості Text, так, що попередні рядки при додаванні нових будуть збережені. Потім для відображення нового рядка і перекладу курсору вліво і на наступний рядок об'єднуються рядок "Рядок", поточний номер рядка і символ повернення каретки (Wrap). Оператор Next завершує цикл.

Зверніть увагу, що Visual Studio автоматично додає оператор Next в кінець циклу, коли ви вводите For в його початку.

Алгоритм реалізації відображення інформації за допомогою циклу For ... Next

11. Щоб зберегти зміни, клацніть на кнопці Save All (Зберегти все) на стандартній панелі інструментів. Тепер запустіть цю програму.
12. Клацніть на кнопці Start (Почати) стандартної панелі інструментів.
13. Клацніть на кнопці Цикл.

Цикл For ... Next відображає в текстовому полі 10 рядків, як показано нижче.


Form1
Цикл
Строка 1
Строка 2
Строка 3
Строка 4
Строка 5
Строка 6
Строка 7
Строка 8
Строка 9
Строка 10

Створення складних циклів For ... Next

Змінна-лічильник в циклі For ... Next може стати потужним інструментом вашої програми.
При деякій уяві ви можете використовувати її для створення в ваших циклах декількох корисних послідовностей чисел. Щоб створити цикл з шаблоном лічильника, відмінним від 1, 2, 3, 4 і т.д., ви можете вказати в циклі різні початкові значення, а потім використовувати ключове слово Step для збільшення лічильника на різні інтервали.

Створення складних циклів For ... Next

```
Dim i As Integer
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 5 To 25 Step 5
TextBox1.Text = TextBox1.Text & "Строка " & i & Wrap
Next i
```

Результат:

Строка 5
Строка 10
Строка 15
Строка 20
Строка 25

Якщо ви оголосите **i** як змінну з плаваючою точкою одинарної або подвійної точності, ви зможете вказати в циклі десяткові значення. Наприклад, цикл For ... Next

```
Dim i As Single
Dim Wrap As String
Wrap = Chr (13) & Chr (10)
For i = 1 To 2.5 Step 0.5
TextBox1.Text = TextBox1.Text & "Рядок" & i & Wrap
Next i
```

Результат:

Строка 1
Строка 1.5
Строка 2
Строка 2.5

Оператор Exit For

Більшість циклів For ... Next виконуються до кінця без будь-яких проблем, але іноді буває потрібно зупинити роботу циклу For ... Next "достроково" при виконанні деякої умови. Таку можливість надає використання оператора Exit For - спеціального оператора для дострокового завершення виконання циклу For ... Next і передачі управління на перший оператор, що стоїть після цього циклу.

Наприклад, наступний цикл For ... Next запитує у користувача 10 імен і відображає їх одне за іншим в текстовому полі доти, поки користувач не введе слово "Готово":

```
Dim i As Integer
Dim InpName As String
For i = 1 To 10
InpName = InputBox ( "Введіть ваше ім'я або наберіть Готово для виходу.")
If InpName = "Готово" Then Exit For
TextBox1.Text = InpName
Next i
```

Якщо користувач вводить "Готово", то оператор Exit For завершує цикл, і виконання триває з оператора, що стоїть після Next.

Цикл Do

В якості альтернативи циклу For ... Next можна написати цикл Do, який виконує групу операторів до тих пір, поки деяка умова не стане True. Цикл Do цінний тим, що найчастіше ви не зможете заздалегідь дізнатися, скільки разів цикл повинен повторюватися. Наприклад, ви можете дозволити користувачеві вводити імена в базу даних до тих пір, поки користувач не введе в поле введення слово "Готово". В цьому випадку можна використовувати цикл Do, що повторюється до тих пір, поки не буде введено слово "Готово". Цикл Do має кілька форматів, що залежать від того, де і як обчислюється умова циклу. Частіше за інших зустрічається такий синтаксис:

Do While умова
блок виконуваних операторів
Loop

Наприклад, наступний цикл Do буде запитувати у користувача дані і відображати їх в текстовому полі доти, поки не буде введено слово "Готово":

```
Dim InpName As String
Do While InpName <> "Готово"
InpName = InputBox ( "Введіть ваше ім'я або наберіть Готово для виходу.")
If InpName <> "Готово" Then TextBox1.Text = InpName
Loop
```

Умовний оператор в цьому циклі має вигляд InpName <> "Готово", і компілятор Visual Basic транслює його в значення "повторювати цикл до тих пір, поки змінна InpName ні буде вміщати слово Готово". Це призводить до одного цікавого моменту, що стосується циклів Do: якщо умова на початку циклу не дорівнює True при першому виконанні оператора Do, то цикл Do ніколи не виконується. Таким чином, якщо строкова змінна InpName містить значення "Готово" до початку циклу (можливо, в результаті більш раннього присвоєння значення в процедурі події), Visual Basic пропустить весь цикл і продовжить виконання з рядка, розташованого за ключовим словом Loop.

Якщо ви хочете, щоб в програмі цикл завжди виконувався хоча б один раз, помістіть перевірку умови в кінці циклу. Наприклад, цикл

```
Dim InpName As String
Do
InpName = InputBox ( "Введіть ваше ім'я або наберіть Готово для виходу.")
If InpName <> "Готово" Then TextBox1.Text = InpName
Loop While InpName <> "Готово“
```

в основному, такий же, як і попередній цикл Do, але тут умова циклу перевіряється після того, як з функції InputBox було отримано ім'я. Перевага цього варіанту в тому, що він оновлює змінну InpName до того, як буде проведена перевірка умови циклу, і, таким чином, попереднє значення "Готово" не призведе до пропуску виконання циклу. Перевірка умови циклу в кінці гарантує, що ваш цикл виконається хоча б один раз, але часто це буде призводити до того, що вам буде потрібно додавати додаткові оператори для обробки даних.

У наступному прикладі показано, як можна використовувати цикл Do для перетворення температур за шкалою Фаренгейта в температури за шкалою Цельсія.

Ця проста програма запитує введення від користувача за допомогою функції InputBox, перетворює температуру і відображає результат у вікні повідомлення.

Перетворення температур за допомогою циклу Do

1. У меню File (Файл) вкажіть на New (Створити), а потім клацніть на Project (Проект). З'явиться діалогове вікно New Project (Створити проект).
2. У папці c: \ vbnet03sbs \ гл.7 створіть новий проект Windows Application (Додаток Windows) на Visual Basic з ім'ям My Celsius Conversion. Буде створено новий проект і в Windows Forms Designer (Конструкторі Windows Forms) з'явиться порожня форма. На цей раз ви помістіть весь код програми в процедурі події Form1_Load, так що під час запуску програми Visual Basic попросить вас ввести температуру за Фаренгейтом. Для запиту даних використовується функція InputBox, а для відображення перетворених даних - функція MsgBox.
3. Зробіть подвійне клацання мишею на формі. У редакторі коду з'явиться процедура події Form1_Load.
4. Введіть в процедурі події Form1_Load наступні оператори програми:

```
Dim FTemp, Celsius As Single
Dim strFTemp As String
Dim Prompt As String = "Введите температуру по
Фаренгейту."
Do
strFTemp = InputBox(Prompt, "Пересчет температуры")
If strFTemp <> "" Then
FTemp = CSng(strFTemp)
Celsius = Int((FTemp + 40) * 5 / 9 - 40)
MsgBox(Celsius, , "Температура по Цельсию")
End If
Loop While strFTemp <> ""
End
```

Цей код виконує необхідні обчислення. Перший рядок оголошує дві змінні одинарної точності - FTemp і Celsius - які зберігають температури за Фаренгейтом і Цельсієм відповідно. Другий рядок оголошує строкову змінну з ім'ям strFTemp, яка зберігає строкову версію температури за Фаренгейтом. Третій рядок оголошується строкову змінну з ім'ям Prompt, яка буде використовуватися функцією InputBox, і привласнює їй початкове значення. Цикл Do запитує у користувача температуру за Фаренгейтом, перетворює її в шкалу Цельсія, а потім відображає на екрані за допомогою функції MsgBox.

Значення, введене користувачем, зберігається в змінної strFTemp. Функція InputBox завжди повертає значення строкового типу, навіть якщо користувач вводить число. Так як ми хочемо виконати з введенням значенням математичні обчислення, змінна strFTemp повинна бути перетворена в число. Для перетворення рядка в число одинарної точності використовується функція CSng. CSng - це одна з кількох функцій, призначених для перетворення рядка в інші типи даних. Потім перетворене значення одинарної точності зберігається в змінної FTemp.

Цикл виконується до тих пір, поки користувач не клацне на кнопці Cancel (Скасувати), або до тих пір, поки не натисне на (Enter) або НЕ клацне на OK при відсутності в поле введення будь-якого значення. Клацання на кнопці Cancel (Скасувати) або введення порожнього значення повертають порожній рядок (""). Цикл перевіряє, чи не порожний рядок, за допомогою перевірки умови While, що стоїть в кінці циклу. оператор програми

Celsius = Int ((FTemp + 40) * 5/9 - 40)

виробляє перетворення значень шкали Фаренгейта в шкалу Цельсія. Цей оператор реалізує стандартну формулу перетворення, але при цьому використовує функцію Int, щоб повернути в змінну Celsius значення, яке не містить десяткових знаків. (Все, що знаходиться праворуч від десяткового дробу, відкидається.) Це відрізання приносить в жертву точність, але допомагає вам уникнути появи довгих значень типу 21.11111, яке є еквівалентом за Цельсієм для температури в 70 градусів за Фаренгейтом.