Лекція №6 ПОіП «Обробка помилок»

(Модуль 1 -)

План лекції.

«Відладка і обробка помилок»

Засоби відладки.

Процес пошуку і усунення помилок називається **відладкою**. Після зупинки можна проаналізувати достаток програми: значення її змінних, вміст стека викликів процедур і вміст пам'яті.

Якщо в рядку виникли проблеми, спочатку необхідно перевірити, як функціонує цей рядок, а потім визначити, що ж конкретно виконується неправильно. Встановивши курсор в цьому рядку, натискуйте **клавішу [F9],** внаслідок чого рядок буде виділений коричневим кольором, тобто стане точкою останову.

Точка останову — це виділений рядок програми, досягши якої автоматично зупиняється виконання програми. Точку останову можна помістити в будь-якому рядку коду.

Натискайте клавішу [F5], щоб продовжити роботу додатку . Робота додатка буде припинена по досягненню точки останову і відкриється вікно редактора коди. Сам рядок виділяється жовтим кольором, а поряд із нею з'являється жовта стрілка. Помістивши курсор над змінною у вікні редактора коду, ви побачите її значення, яке використовується в даний момент при виконанні коду.

Перевірити, що станеться далі, можна, натискує клавішу [**F8] – покрокове виконання програми**. При натисненні вказаної клавіші ви виконуєте лише наступний рядок коду, тому робота програми буде зупинена на цьому рядку. Перш за все зупините програму за допомогою команди **Stop Debugging** з меню **Debug** (або натискуйте [ctrl+alt+break]).

Покрокове виконання програми

Як це часто буває, ми чекали знайти одну помилку, а зіткнулися із іншою.

Подальші рядки програми ми виконаємо із зупинками на кожній з них. Щоб виконати один рядок коде, потрібно натискати клавішу Step Over[Shift+F8] «крок з обходом» . Програма зробить один «крок» і зупиниться, а відладчик виділить жовтим наступний рядок.

Існує ще один спосіб покрокового виконання коди, яка називається Step Into«Крок із заходом», - клавіша [F8].

Вікна Locals і Watch

Залишаючись в режимі відладки, виберіть з меню команду Debug | Windows|Locals.B нижній частині IDE відкриється вікно Locals. У цьому вікні виводяться поточні значення **всіх локальних змінних**. Тепер нам зручно стежити за зміною змінних при покроковому виконанні програми.

Вікно Watch подібно окну **Locals,** але дозволяє переглядати значення необхідних локальних змінних і складних виразів. Щоб додати змінну у вікно, потрібно клацнути на ній правою кнопкою і вибрати **Add Watch.**

Ви можете також обчислити значення виразів, що містять будь-які змінні цього обробника подій, ввівши відповідний рядок у вікно **Command.** Вікно **Command** розташовується внизу IDE. Якщо він не видно на екрані, виберіть з меню команду **View | Other Windows | Command Window.**

Приклади:

? (3/4) - 0.75

? a / b

Для здобуття випадкового числа з діапазону від 0 до 1 введіть такий рядок:

? Rnd()

Щоб повернути інтерфейсу первинний вигляд –

Windows - Reset Windows Layout

Типи помилок

У програмах, написаних на різних мовах, зустрічаються однотипні помилки. Їх можна розділити на три групи:

- помилки часу розробки
- помилки часу виконання
- логічні помилки.

Помилки часу розробки знаходити і виправляти найлегше, бо середа розробки Visual Studio .NET сама повідомить, в якому рядку ви допустили помилку і яку частину рядка вона не розуміє.

Помилкою часу розробки називається невідповідність фрагмента коди правилам мови програмування, на якій він написаний. Програму, що містить хоч би одну помилку часу розробки, не можна відкомпілювати і запустити. Хвилясті лінії, що підкреслюють слово,вказують на помилку часу розробки. Щоб взнати, в чому саме полягає помилка, підвести курсор до коду з помилкою або переглянути у вікні Error List. У цьому вікні виводиться інформація про помилки в програмному коді.

Помилки часу виконання важче знаходити, оскільки в цьому випадку Visual допомогти не зможе — він нічого не знає про помилку до тих пір, поки вона не станеться. Такі помилки виявляються тоді, коли програма намагається виконати неприпустиму операцію, наприклад звернутися до неіснуючих даних або ресурсів, на використання яких у неї немає дозволу і можуть викликати аварійне завершення роботи.

Приклади:
Спроба відкрити неіснуючий файл.
Спроба увійти на сервер з неправильним ім'ям або паролем.
Спроба звернутися до папки, не маючи на це права доступу.
Запит даних з перейменованої таблиці бази даних.
Відкриття файлу на сервері, який на час профілактики відключений.
Спроба доступу до неіснуючого URL в Internet.
Спроба ділення числа на нуль.
Введення символьних даних замість очікуваних числових (або навпаки).
Якщо в програмі **є логічна помилка**, це може ніяк не виявитися у момент її виконання. Надалі результатом такої помилки може стати неадекватна поведінка програми або видача неправильних даних.

```
Private Sub Buttonl_C1ick(ByVa1 sender As System.Object, _
ByVal e As System.EventArgs) Handles Buttonl.Click
Dim i As Integer
 i = 1
Do While i > 0
i =i+ 1
 Loop
End Sub
```

При цьому в умові циклу вказано, що він повинен виконуватися, поки змінна 1 буде більше нуля. Проблема очевидна: значення змінної завжди буде більше нуля, тому цикл ніколи не завершиться. Такий цикл не має кінця. Хоча насправді цикл не може продовжуватися нескінченно — після 2 мільярдів ітерацій **змінна переповниться і в програмі станеться помилка часу виконання.**
Логічні помилки, як правило, не приводять до аварійного завершення роботи програми, найчастіше код просто робить не те, що передбачалося.

Перехоплення помилок

слово «**bug**» **(**що означає помилку в програмі) використовується рідко. Прийнято називати такі помилки **виключеннями**. Їх можна сприймати як ситуації, що не виникають при нормальному виконанні коди. Проте термін «debugging» (відладка) до цих пір широко поширений.
Код, який перехоплення помилок, може бути дуже неупорядкованим, і саме із-за цієї проблеми компанія Microsoft запропонувала **метод**, званий **структурною обробкою помилок**. Він більше личить для обробки виключень або помилок, що виникають при виконанні програми.
У основі методу лежить наступне твердження: при виникненні виняткової ситуації виконання програми не переривається з повідомленням про помилку, а викликається та частина коду, яка обробляє відповідне виключення.

Як запобігти виникненню виключення. Єдиним виходом є додавання **структурного обробника виключень**. Єдине місце, де виключення може мати місце, це обробник події кнопки із знаком рівності, при натисненні якої і виконуються обчислення. Саме

тут потрібно помістити обробник виключень. Для цього використовується наступний код:-

Оператор Try... Catch... Finally забезпечує спосіб обробити деякі або всі можливі помилки, які можуть статися в даному блоці коди, при виконанні коду.

```
Try
{ блок операторов}
 Catch [ exception [ As type ] ] [ When expression ]
 { блок обработчика}
[ Exit Try ]
...
Finally
   { блок очистки операторов}
End Try
```

Цей код намагається виконати обчислення, які описані в блоці операторів. Коли все відбувається успішно, реалізується частина коде з блоку очищення. Якщо при виконанні операторів в першому блоці виникають помилки, активізується рядок Catch Exception і викликається код з блоку обробника.

Оператори в блоці Catch Exception відповідають за обробку виключень. Вони не дозволяють зробити щось істотне з помилками, що з'явилися в результаті обчислень. З їх допомогою можна лише вивести повідомлення про помилку і надати користувачеві можливість змінити введені значення. Існують також помилки інших типів, які можуть оброблятися витонченіше.

немає єдиного універсального методу для обробки всіх виключень, просто потрібно заздалегідь передбачати всі можливі виключення і обробляти кожне з них, використовуючи якнайкращий метод. Обробники помилок в математичних застосуваннях повинні інформувати користувача про виникаючі помилки і припиняти обчислення, навіть не відображуючи результат.

Велику частину часу обробник помилок неактивний і ніяк не взаємодіє з програмою. При виникненні помилки (найчастіше - всього це переповнювання) виконується блок Try...Catch...End Try.

Приклади:

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
    Dim s As String
    s = "питання"
    Button4.Text = s.Substring(10, 1)
End Sub
```

Ця підпрограма намагається вивести на кнопці Button4 одинадцятий символ рядка "питання". Оскільки рядок містить всього шість символів, то програма генерує виключення.

Щоб обробити це виключення, необхідно укласти помилковий код всередину обробника помилок.

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        Dim s As String
        s = "рядок"
        Try
            Button4.Text = s.Substring(10, 1)
```

```
        Catch
            Button4.Text = "error"
        End Try
 End Sub
```

Інструкція Substring поміщена всередину блоку Try...Catch...End Try. Даний блок є базовою структурою мови VB .NET, призначеною для обробки помилок. Якщо будь-яка інструкція, виконана після інструкції Try, згенерує виключення, управління буде передано коду, розташованому після інструкції Catch. За відсутності виключення блок Catch просто пропускається.

System.ArgumentOutOfRangeException

VB .NET розділяє всіх можливі типи помилок на групи. У відповідь на виключення створюється екземпляр об'єкту, що успадковує клас Exception. У повідомленні про помилку сказано, що в даному випадку згенерував об'єкт класу System. ArgumentOutOfRangeException.

Приведені вище обробники виключень не розрізняють тип помилки. Будь-яке виключення, що згенеровано в блоці Try, передає управління в блок Catch, де завжди виконуються одні і ті ж дії. Проте можна написати обробник виключень, що розрізняє тип помилки.

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button5.Click
        Try
            Button5.Text = ListBox1.SelectedItem.ToString
        Catch oEX As System.NullReferenceException
            MsgBox("виберіть із списку")
        Catch oEX As Exception
            MsgBox("other error: " & oEX.Message)
        End Try
    End Sub
```

Ця підпрограма намагається прочитати з елементу управління Listbox, виділений елемент списку і вивести його текст на кнопці. Якщо жоден елемент не виділений, згенерує виключення System.NullReference-Exception, і ми використовуємо цю інформацію для того, щоб попросити користувача вибрати що-небудь із списку. Якщо станеться якесь інше виключення, код виведе повідомлення про помилку.

У списку виключень спочатку обробляється конкретне виключення, а потім — універсальний об'єкт, відповідний будь-якому іншому виключенню. Ви також повинні стежити за тим, щоб виключення оброблялися в правильному порядку. Якщо спочатку виконати універсальну частину блоку Catch, то дії, пов'язані з конкретними виключеннями, взагалі не будуть виконані.

Змінна oEX використовується в кожній гілці блоку Catch, завдяки тому, що інструкція Catch грає роль неявного її оголошення. Ця змінна доступна лише усередині блоку Catch.

Блок Finally

інколи перед виходом з блоку обробки виключень необхідно виконати код, поновлюючий нормальну роботу програми, – це робиться за допомогою **Finally**.

Приклад з When

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        Dim x As Integer = 5
        Dim y As Integer = 4
        Try
            x =x/ y
            MsgBox(x)  ' вызывает  "Divide by Zero" error.
        Catch ex As Exception When y = 0   ' Catch the error.
            MsgBox(ex.ToString) ' Show friendly error message.
        Finally
            Beep()   ' Beep after error processing.
        End Try
    End Sub
```

Будь-яку математичну функцію можна реалізувати одним рядком коду. В деяких випадках потрібно буде включити код виявлення помилок або скористатися методами перевірки даних. Наприклад, функція Sqrt() для извлечения квадратного кореня застосовна лише до аргументу >0. При введенні значення <0 програма повинна повідомити про це:

```
  If Label1.Text < 0 Then
            MsgBox("Can't calculate the square root of a negativeumber")
            ' (Не можна витягувати квадратний корінь з негативного числа)
        Else
            Label1.Text = sqrt(Val(Label1.Text))
  End If
```

Функція Log() обчислює логарифм, але лише позитивних чисел. Якщо додати в код функцію обчислення логарифма і ввести негативне число, то результатом її виконання буде нечислове значення NAN. Це значення тотожне невизначеності, воно говорить про те, що результат не є допустимим числом. Звичайно, відображення значения NAN — не дуже хороший метод обробки математичних помилок. Краще виконувати перевірку даних і виводити повідомлення про допустимі значення, которые можуть бути введені

```
If  Val(1b1Disp1ay.Text)< 0 Then
      MsgBox("Can t calculate the logarithm of a negative umber")
Else
IblDisplay Text = Math.Log(1b1Display.Text)
End If
```

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim a, b, з, x As Integer
        Dim n As Integer
        Dim в As Integer
        'Dim в As Double
        Dim гілка As String
        Dim кнопка As Integer
        a = 2 : b = 4 : з = 6
1:      x = InputBox("Введіть значення аргументу ", "Вікно введення" -6)
        'Визначуваний номер вітки
 If x <= a Then n = 1 Else If x > a And x <= b Then n = 2 Else If x > b And x <= з Then
n = 3 Else n = 4
```

```
        'залежно від номера вітки обчислюваний в
        Try
            Select Case n
                Case 1
                    гілка = "перша"
                    в = Math.Sqrt(x)
                Case 2
                    гілка = "другу"

                    в = Math.Sqrt(Math.Cos(x))
                Case 3
                    гілка = "третю"
                    в = Math.Sqrt(Math.Tan(x))
                Case Else
                    гілка = "четверту"
                    в = Math.Log(x) / Math.Log(10)
            End Select
            'виводимо на екран результат
            кнопка = MsgBox("Ти потрапив в " & гілка & " гілку y=" & в & Chr(13)_
             & Chr(13)& "Працюємо далі? - ПОВТОР" & Chr(13)& _
            "Припініті введення? - ВІДМІНА", vbQuestion + vbRetryCancel _
            "Відповідь: ")
            'блок обробника виключень
        Catch exc As Exception
         кнопка = MsgBox("ПОМІЛКА!!!" & "Ти потрапив в " & гілка & "  гілку" & " Але_
ПОМИЛКА x=" & x & " не належить ОДЗ" & Chr(13) & "Працюємо далі? - ПОВТОР" & Chr(13)_ &
"Припинити введення? - ОТМЕНА", vbCritical + vbRetryCancel, "Відповідь: ")
        Finally
            Beep()
        End Try
        'Реакція на натискання кнопок
        If кнопка = vbRetry Then GoTo 1
    End Sub


Dim message, title, defaultValue As String
        Dim myValue As Object
        ' Set prompt.
        message = "Enter a value between 1 and 3"
        ' Set title.
        title = "InputBox Demo"
        defaultValue = "1"   ' Set default value.

        ' Display message, title, and default value.
        myValue = InputBox(message, title, defaultValue)
        ' If user has clicked Cancel, set myValue to defaultValue
        If myValue Is "" Then MsgBox("Nothing") 'myValue = defaultValue
```