Sistemas Operativos 6to

Desarrollo de Aplicaciones con Lenguajes Visuales

Docente / Martin Murdaca

UNIDAD 2
3er trimestre 2020

1. Presentación

Hola a todos, les cuento que esta semana encontrarán publicado junto a la clase un proyecto **Visor de archivos gráficos**, donde utilizaremos varios de los objetos desarrollados en las clases pasadas

También reitero que es importante y **obligatorio ingresar al aula todas las semanas por lo menos una vez,** no sólo para leer o descargar material, sino también para interactuar o consultar dudas con los compañeros. Ante cualquier duda sigamos conectados a través de mi dirección de correo personal martinmurdaca@gmail.com

En esta clase ahondaremos sobre los distintos **tipos de errores** que se producen al momento de programar. Ellos son: errores de sintaxis, de ejecución y lógicos.

Los primeros errores como seguramente ya habrán experimentado, los detecta mayoritariamente el editor de Visual Basic. Si abrimos un For y olvidamos cerrarlo, el editor subrayará de color azul el For por ejemplo y si ubicamos el mouse sobre el subrayado, le indicará el error que se está cometiendo, para que ustedes puedan solucionarlo rápidamente.

El guion se centrará en el segundo tipo de errores, en los de ejecución, en los que el error puede ocasionar que el programa deje de funcionar y se cierre.

Veremos una estructura de control de errores estructurado, la estructura **Try…EndTry** ejemplificando con fragmentos de código las distintas partes que la conforma.

Desarrollo de los contenidos

Manipulación de errores

¿Qué son los errores?

Aún los mejores programadores cometen errores. De hecho, cuanto más largo es el programa, existe mayor probabilidad de cometer errores.

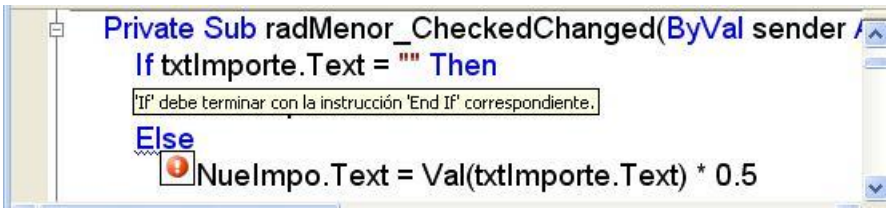
Hay errores de diferentes tipos:

- **Errores de sintaxis**: los que ocurren cuando se escribe mal un comando, o se deja fuera una frase o argumento esperado. Por ejemplo: escribir un If y olvidarse la instrucción de cierre, el EndIf.
- **Errores de ejecución**: los mismos son causados por circunstancias fuera de control del programa. Por ejemplo: invocar el contenido de un DVD y haber olvidado colocarlo.
- **Errores lógicos**: estos son errores que provocan que el programa produzca resultados equivocados. Por ejemplo: intentar mostrar el contenido de una variable antes de desplegar un cuadro de entrada para cargar la misma. Si bien estas sentencias no tendrían errores de sintaxis, el hecho de haber invertido el orden en que deben codificarse provocaría que la variable siempre se muestre en blanco.

En Visual Basic, el primer tipo de errores, es interceptado en la mayoría de los casos al momento de escribir el código por el IDE de Visual Basic. Ejemplo de esto sería tratar de escribir un bloque If sin la sentencia EndIf. Visual Basic le mostraría la ventana de la Figura 1, para que ustedes puedan corregir el error aprovechando la explicación o el mensaje de ayuda de Visual Basic o continuar con el mismo. En este caso todo el código de esa línea quedará con un subrayado azul.

Los otros tipos de errores, errores en tiempo de ejecución y lógicos, debe corregirlos el programador. En este capítulo nos concentraremos en corregir solo, los errores de ejecución.

Figura 1. Error de escritura de código



En Visual Basic los errores en tiempo de ejecución se conocen como *errores interceptables*, esto puede ser, los usuarios olvidan insertar discos en las unidades de disco, los sistemas se quedan sin memoria y los archivos no están donde espera encontrarlos. Si agregan a su aplicación un código eficaz de tratamiento de errores, crearán una aplicación más robusta.

Visual Basic, detecta que ha ocurrido un error y les permite interceptarlo y realizar alguna acción correctiva. Si ustedes no manejan los errores interceptables en sus códigos, cualquier

error en tiempo de ejecución podría ser fatal, esto significa que Visual Basic desplegará un mensaje muy poco significativo de error al usuario y luego finalizará el programa.

Errores y excepciones

Dentro del esquema de gestión de errores encontramos los conceptos de error y excepción. A continuación presentamos la definición de estos conceptos:

- **Error**: es un evento que se produce durante el transcurso del programa provocando una interrupción en su flujo de ejecución. Al producirse un error, este genera un objeto excepción.
- **Excepción**: es un objeto generado por un error con información sobre las características del error producido.

Manipulación estructurada de errores

En este tipo de tratamientos, cada vez que se produce un error, se genera un objeto de la clase Exception conteniendo la información del error ocurrido. La idea es un utilizar una estructura de control que capture este objeto.

Observen en el siguiente video como se ejemplifica el uso de la estructura **Try….Entry** para el caso de una sumatoria sencilla mediante dos cuadros de edición: https://www.youtube.com/watch?v=yA5ugVeSVCw

La estructura Try…EndTry

Esta estructura nos permite escribir un bloque de código sensible a errores y los correspondientes manipuladores de excepciones de acuerdo al error producido.

La sintaxis es la siguiente:

```
Try
        ‘código que pude provocar errores
[Catch [Excepcion [As TipoExcepcionA]]
        ‘respuesta a error A
[Exit Try]
]
[Catch [Excepcion [As TipoExcepcionN]]
        ‘respuesta a error N
[Exit Try]
]
[Finally
        ‘código posterior al control de errores
]
End Try
```

Todo el código escrito debajo de la palabra clave **Try** es el código sensible a errores, es decir el código que puede provocar un error de ejecución.

Desde la palabra clave **Catch** comienza el manipulador de errores donde opcionalmente se puede identificar el tipo de excepción que se está generando.

De lo contrario se crea un manipulador global de errores. Observe el siguiente ejemplo:

```
Private Sub btnIngresar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnIngresar.Click
Dim vnumero As Integer

Try
    vnumero = InputBox("Ingrese un número")
    MessageBox.Show("El numero ingresado es: " & vnumero)
Catch ex As Exception
    MessageBox.Show("Debe ingresar un número")
End Try
End Sub
```

Tanto si se produce o no el error la sentencia Finally nos permite escribir una porción de có-digo, que será ejecutada en ambos casos.

```
Private Sub btnIngresar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnIngresar.Click

Dim vnumero As Integer

Try
    vnumero = InputBox("Ingrese un número")
    MessageBox.Show("El número ingresado es: " & vnumero)
Catch ex As Exception
    MessageBox.Show("Debe ingresar un número")
Finally
    MessageBox. Show (“La rutina de control de errores ha finalizado”)
End Try
End Sub
```

La clase **Excepción** nos proporciona información sobre el error que se produce a través de sus métodos y propiedades:

Message: esta propiedad describe el error acaecido

ToString: este método devuelve una cadena con toda la información detallada del error. Incluye la descripción, el objeto que lo provoca, el código y demás.

Siguiendo con el ejemplo anterior si reemplazamos el mensaje creado personalizado por la propiedad Message así quedaría:

```
Private Sub btnIngresar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnIngresar.Click
Dim vnumero As Integer
Try
    vnumero = InputBox("Ingrese un número")
    MessageBox.Show("El numero ingresado es: " & vnumero)
```

```
Catch ex As Exception
        MessageBox.Show(ex.Message)
Finally
        MessageBox. Show ("La rutina de control de errores ha finalizado")
End Try
```

Finalmente, es importante destacar que si necesitan pueden realizar dentro de su programa una rutina para capturar excepciones de diferente tipo en el mismo controlador de errores, agregando varias líneas con la palabra clave Catch. Observen el siguiente bloque de código…

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim x,y,p As Integer

Try
        y = InputBox("Ingrese un valor")
        x = InputBox("Ingrese otro valor")
        p = y / x
Catch ex As OverflowException
        MessageBox.Show(ex.Message)
Catch ex As Exception
        MessageBox.Show ("Controlador de errores general")
End Try

End Sub
```

Si en la variable x se ingresa el valor 0 el programa arrojara un error de desbordamiento. De lo contrario, por ejemplo si ingresa una letra en lugar de un número se producirá un error, pero ingresará en la segunda sentencia Catch.

TAREA

- Realicen las ejercitaciones presentadas en esta clase.
- Prueben y estudien el funcionamiento del Visor de archivos gráficos.

CHARLA VIRTUAL

A la brevedad estaremos organizando una charla desde Zoom.

Cierre de la clase

En esta clase presentamos una estructura para manipular y controlar los errores de ejecución del programa, que si bien no mejoran el rendimiento de sus proyectos, los vuelven más confiables y seguros a la hora de utilizarlos.

Reitero la importancia de participar en la experiencia de reflexionar y evaluar el trabajo de todos. Seguramente lograremos grandes resultados.

¡¡¡Que pasen todos una excelente semana!!

Martin Murdaca