Práctica A: Crear y utilizar procedimientos

****

Objetivos

En este laboratorio, aprenderemos a:

- Crear procedimientos **Function** en un módulo.
- Pasar parámetros a un procedimiento.
- Invocar procedimientos que se encuentren en un módulo.

Nota Este laboratorio se centra en los conceptos de este módulo y, en consecuencia, podría no cumplir las recomendaciones de seguridad de Microsoft.

Requisitos previos

Para trabajar en este laboratorio, es necesario:

- Tener experiencia en el uso de variables y constantes en Visual Basic .NET.
- Estar familiarizado con la creación e invocación de procedimientos en Visual Basic .NET.
- Tener experiencia en el uso de controles y formularios en Visual Basic .NET.

Escenario

En los laboratorios de los Módulos 2, 4, 5, 6 y 12, crearemos una aplicación de estimación de pagos de un crédito. Crearemos toda la aplicación por fases, siguiendo cada fase con el código creado en el laboratorio anterior. Al principio de cada laboratorio, podemos seguir con nuestros propios archivos o empezar con los archivos que se nos proporcionan.

En el Laboratorio 4.1, *Crear y utilizar procedimientos*, crearemos procedimientos para el cálculo de un crédito.

Importante En estos laboratorios, se utiliza el método **Form.Close** para cerrar la aplicación de estimación de pagos de un crédito. En una aplicación con sólo un formulario, **Form.Close** cierra la aplicación y no es necesario invocar **Application.Exit**. En una aplicación con varios formularios, se debería invocar **Application.Exit** para cerrarla. Además, debido a que los eventos **Form.Closed** y **Form.Closing** no se ejecutan cuando se invoca a **Application.Exit**, se debería invocar **Form.Close** para cada formulario abierto antes de invocar **Application.Exit** si hay código en los eventos **Closed** y **Closing** que debe ejecutarse.

Archivos de solución

Los archivos de solución para este laboratorio se encuentran en las carpetas PracticaA\Ex01\Solution y PracticaA\Ex02\Solution dentro del archivo labs04.zip.

Tiempo estimado para realizar este laboratorio: 60 minutos

Ejercicio 1
Crear funciones en un módulo

En este ejercicio, crearemos funciones de utilidad en un módulo. Estos procedimientos calcularán el pago mensual y la cantidad total pagada durante el periodo del crédito.

↙ Abrir el proyecto de aplicación de crédito

- Abrir el proyecto de aplicación de crédito del Laboratorio 2.1, Crear el interfaz de usuario, en el Módulo 2, "Trabajar con formularios y controles". Si no hemos finalizado el Laboratorio 2.1, podemos utilizar el proyecto LoanApplication.sln de la carpeta PracticaA\Ex01\Starter dentro del archivo labs04.zip.

↙ Añadir un nuevo módulo al proyecto

1. En el menú **Proyecto**, hacer clic en **Agregar nuevo elemento**.
2. En el cuadro de diálogo **Agregar nuevo elemento**, hacer clic en **Módulo** en el panel **Plantillas**.
3. Cambiar el nombre del módulo por **Utility.vb** y hacer clic en **Abrir**.
4. Declarar una constante de nivel de módulo que represente el número de meses de un año. Nuestro código debería ser similar al siguiente:

```
Private Const conversionPeriod As Integer = 12
```

↙ Calcular el pago mensual

1. Añadir la siguiente función pública a la definición del módulo.

Nombre función	Parámetros	Valor devuelto

MonthlyPayment	`loanAmount As Double,`
	`rate As Double,`
	`loanLength As Integer`
		Double

2. Completar el cuerpo de la función:
   a. Declarar una variable **Double** con el nombre *interestRate* para guardar el tipo de interés.
   b. Declarar una variable **Double** con el nombre *monthRate* para guardar el tipo de interés mensual.
   c. Declarar una variable **Integer** con el nombre *numberOfPayments* para guardar el número de pagos.
   d. Calcular el tipo de interés utilizando la siguiente línea de código:

```
interestRate = rate / 100
```

   e. Calcular el tipo de interés mensual y guardar este valor como *monthRate*, como se muestra en el siguiente código:

```
monthRate = interestRate / conversionPeriod
```

f. Calcular el número de pagos que se realizarán, y guardar este valor como *numberOfPayments*, como se muestra en el siguiente código:

```
numberOfPayments = loanLength * conversionPeriod
```

g. Utilizar la función **Pmt** de Visual Basic para calcular el importe del pago mensual, y establecer esa cantidad como valor devuelto de la función, como se muestra en el siguiente código:

```
MonthlyPayment = Pmt(monthRate, numberOfPayments, _
    -loanAmount)
```

↙ Calcular el pago total

1. Añadir la siguiente función pública a la definición del módulo.

Nombre función	Parámetros	Valor devuelto

TotalPaid	`loanAmount As Double,`
	`rate As Double,`
	`loanLength As Integer`
		Double

2. Completar el cuerpo de la función:

   a. Declarar una variable **Integer** con el nombre *numberOfPayments* para guardar el número de pagos.

   b. Calcular el número de pagos que se realizarán, y guardar este valor como *numberOfPayments*, como se muestra en el siguiente código:

```
numberOfPayments = loanLength * conversionPeriod
```

   c. Invocar la función **MonthlyPayment** para calcular el pago mensual. Multiplicar el valor devuelto desde la función **MonthlyPayment** por el número total de pagos. Establecerlo como valor devuelto de la función **TotalPaid**.

   Nuestro código debería ser similar al siguiente:

```
TotalPaid = MonthlyPayment(loanAmount, rate, _
    loanLength) * numberOfPayments
```

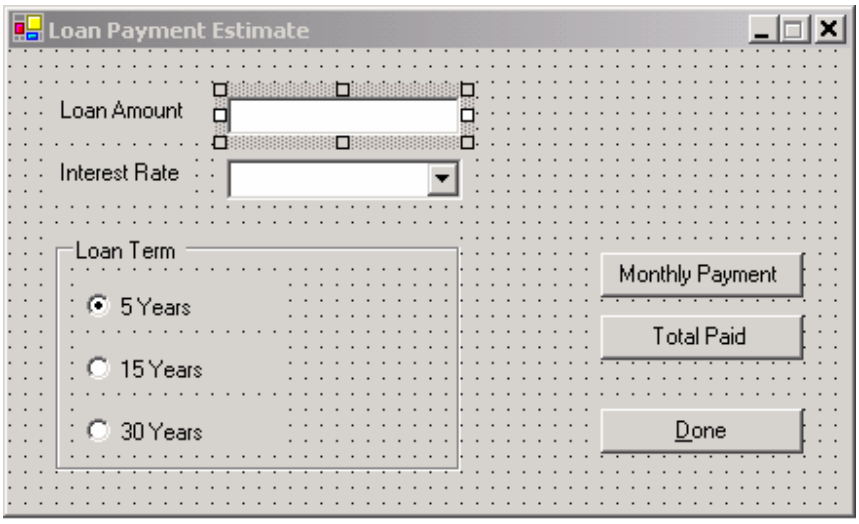
↙ Crear el proyecto

- Crear el proyecto y corregir errores.

Ejercicio 2
Trabajar con el formulario Main

En este ejercicio, añadiremos dos botones al formulario Main. Estos botones ejecutarán el código que calcula los pagos mensuales y el pago total del importe del crédito.

La siguiente ilustración muestra la ubicación de los nuevos botones que añadiremos al formulario en el primer procedimiento.



↙ Añadir controles al formulario Main

1. Abrir el archivo **Main.vb** en vista de Diseño. Si no hemos completado el Ejercicio 1, utilizar el proyecto de la carpeta PracticaA\Ex02\Starter dentro del archivo labs04.zip.
2. Añadir dos controles **Button** al formulario.
3. Establecer las propiedades para los controles como se muestra en la siguiente tabla:

Nombre actual	Propiedad	Nuevo valor
Button1	**Name**	**MonthlyPaymentButton**
	Text	**Monthly Payment**
Button2	**Name**	**TotalPaidButton**
	Text	**Total Paid**

↙ Almacenar el periodo del crédito

1. En la clase **MainForm**, declarar una variable **Integer** privada a nivel de clase con el nombre *loanTerm*. Inicializarla a 5.

   Nuestro código debería ser similar al siguiente:

```
Public Class MainForm
   Inherits System.Windows.Forms.Form

   Private loanTerm As Integer = 5

End Class
```

2. Para cada uno de los tres botones de opción, añadir un controlador de eventos **CheckedChanged**.
3. En cada controlador de eventos, establecer *loanTerm* con el valor apropiado asociado al botón de opción. Nuestro código debería ser similar al siguiente:

```
Private Sub Length5RadioButton_CheckedChanged(...)
   loanTerm = 5
End Sub

Private Sub Length15RadioButton_CheckedChanged(...)
   loanTerm = 15
End Sub

Private Sub Length30RadioButton_CheckedChanged(...)
   loanTerm = 30
End Sub
```

↙ Añadir un controlador de eventos para MonthlyPaymentButton

- En el controlador de eventos **MonthlyPaymentButton_Click**, realizar los siguientes pasos:
   a. Invocar la función **MonthlyPayment** creada en el Ejercicio 1.
   b. Utilizar la función de conversión de tipos de datos **CDbl** para convertir los tipos de datos importe del crédito y tipo de crédito antes de pasar estos valores a la función.
   c. Utilizar un cuadro de mensaje para mostrar el pago devuelto desde la función.

   Nuestro código debería ser similar al siguiente:

```
MessageBox.Show(MonthlyPayment(CDbl(LoanTextBox.Text), _
   CDbl(RateComboBox.Text), loanTerm)
```

↙ Añadir un controlador de eventos para TotalPaidButton

- En el controlador de eventos **TotalPaidButton_Click**, realizar los siguientes pasos:
   a. Invocar la función **TotalPaid** creada en el Ejercicio 1.
   b. Utilizar la función de conversión de tipos de datos **CDbl** para convertir los tipos de datos importe del crédito y tipo de crédito antes de pasar estos valores a la función.

c. Utilizar un cuadro de mensaje para mostrar el pago devuelto desde la función.

Nuestro código debería ser similar al siguiente:

```
(MessageBox.Show(TotalPaid(CDbl(LoanTextBox.Text), _
   CDbl(RateComboBox.Text), loanTerm)
```

↙ Probar la aplicación

- Generar, ejecutar y probar la aplicación introduciendo importes de créditos y escogiendo tipos de interés y periodos para el crédito. Observar que los valores de moneda no están formateados.

↙ Formateo de los valores de moneda

1. En las instrucciones del cuadro de mensaje que invocan la función **MonthlyPayment** y la función **TotalPaid**, utilizar la función **FormatCurrency** para formatear el importe mensual de pago del crédito y el total pagado antes de mostrar estos valores.

   Nuestro código debería ser similar al siguiente:

   ```
   MessageBox.Show(FormatCurrency(MonthlyPayment(CDbl _
      (LoanTextBox.Text), CDbl(RateComboBox.Text), _
      loanTerm)))

   MessageBox.Show(FormatCurrency(TotalPaid(CDbl _
      (LoanTextBox.Text), CDbl(RateComboBox.Text), _
      loanTerm)))
   ```

2. Probar de nuevo la aplicación y comprobar que los valores de moneda están formateados correctamente.