Programación III
C#

CARRERA: Análisis de Sistemas
SEMESTRE: IV

INDICE

C# (léase C Sharp), es una evolución que Microsoft realizó de este lenguaje, tomando lo mejor de los lenguajes C y C++, y ha continuado añadiéndole funcionalidades, tomando de otros lenguajes, como java, algo de su sintaxis evolucionada. Lo orientó a objetos para toda su plataforma NET (tanto Framework como Core), y con el tiempo adaptó las facilidades de la creación de código que tenía otro de sus lenguajes más populares, Visual Basic, haciéndolo tan polivalente y fácil de aprender como éste, sin perder ni un ápice de la potencia original de C. En la versión de .NET Core, se ha reconstruido por completo su compilador, haciendo las aplicaciones un 600% más rápidas.

De la misma forma aprenderemos lo siguiente:

- crear un proyecto con C#;
- manipular controles y sus propiedades en tiempo de diseño;
- ejecutar un programa;
- manejar un evento de clic de botón;
- mostrar un cuadro de mensaje;
- colocar texto en una etiqueta en tiempo de ejecución;
- localizar errores de compilación.

Marirene del Socorro Sánchez Rodríguez

Msc. En Telecomunicaciones

Ingeniero de Sistemas

Docente de Escuela de Análisis de Sistemas

Una publicación de

Instituto Universitario de Tecnología para la Informática

Es necesario decir, que C#, No ha perdido la potencia original de C, es decir, se puede acceder a bajo nivel al núcleo de los sistemas operativos, trabajar con punteros a memoria (muchos desarrolladores tienen verdadero pánico a los punteros) e interactuar con elementos físicos de los dispositivos, como tarjetas gráficas o puertos USB, por ejemplo. Además, como hemos comentado con anterioridad, C# es un lenguaje diseñado para su uso en .NET, cuyo objetivo de esta plataforma es crear aplicaciones de forma sencilla. Por tanto, este lenguaje se utiliza para diseñar aplicaciones en esta plataforma.

Se hace hincapié en ello, ya que este lenguaje se diseñó expresamente para la plataforma .NET, por lo que las características de .NET serán las propias de este lenguaje de programación:

Sencillez: C# elimina gran cantidad de elementos que son innecesarios en .NET. Por ejemplo, no se incluyen elementos pocos útiles como macros, herencias múltiples o la necesidad de un operador distinto del punto.

Modernidad: C# Incorpora de forma automática e intuitiva en su lenguaje elementos que se han demostrado con el paso de los años que han sido muy útiles para el desarrollo de aplicaciones.

Seguridad: Incorpora mecanismo para asegurar que los accesos a tipos de datos se lleven a cabo de forma correcta, por lo que se evita que generen errores difíciles de detectar.

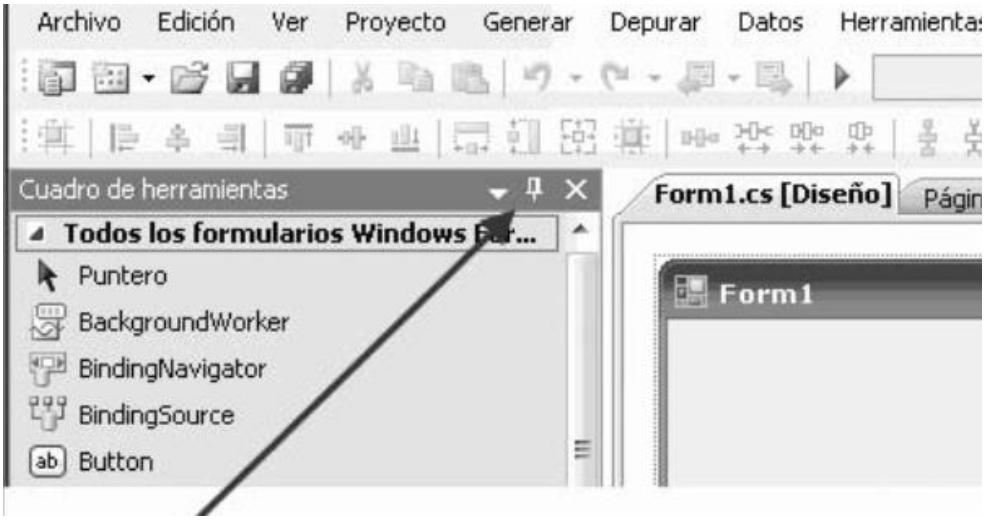
Sistemas de tipos unificados: Todos los datos que obtenemos al programar C# se guardan en una base para que se puedan volver a utilizar posteriormente.

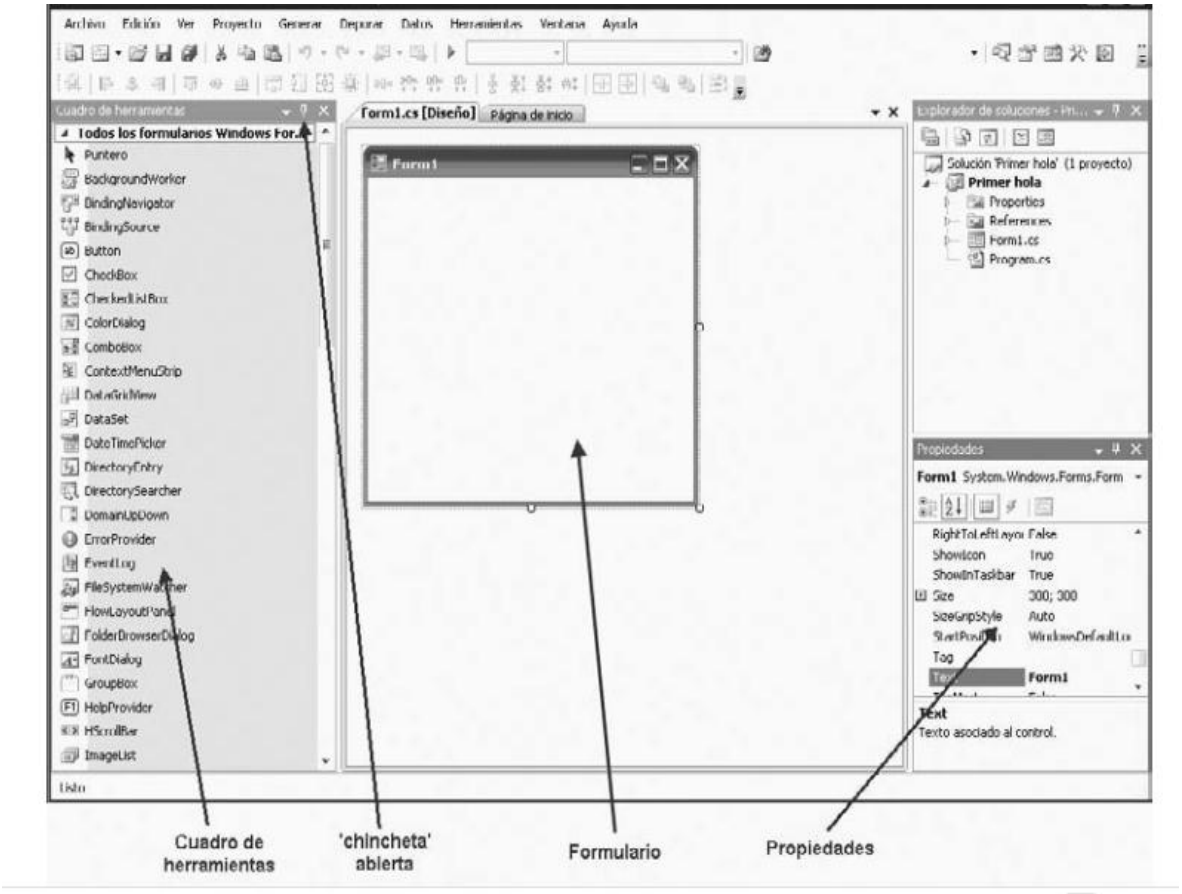
Extensibilidad: puedes agregar tipos de datos básicos, operadores y modificadores cuando se vaya a programar.

Versionable: Dispone de actualización y mejora continua, permitiendo crear versiones de tipo sin tener miedo a que, con la incorporación de nuevos integrantes, provoquen errores complicados de detectar.

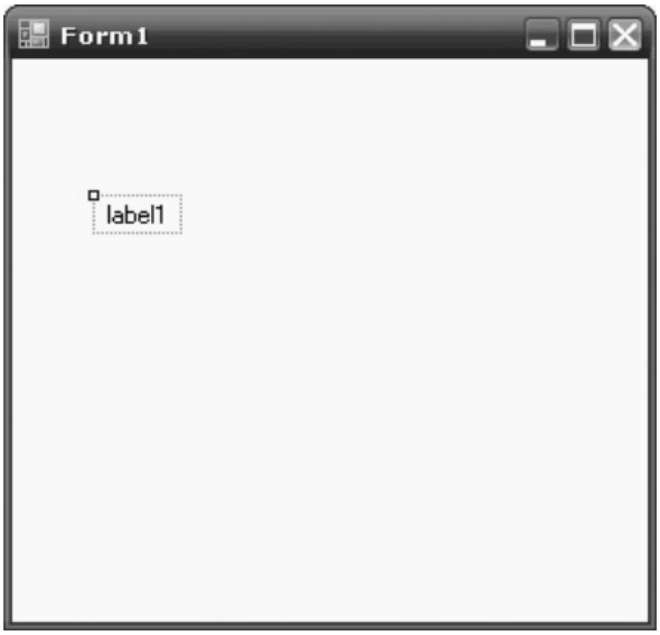
Compatible: C# mantiene una sintaxis muy parecida a C, C ++, Java y muchos otros lenguajes de programación, para facilitar el trabajo del programador.

Eficiente: a pesar de las restricciones que tiene C# en todo el código, se puede saltar estas restricciones utilizando objetos a través de punteros.

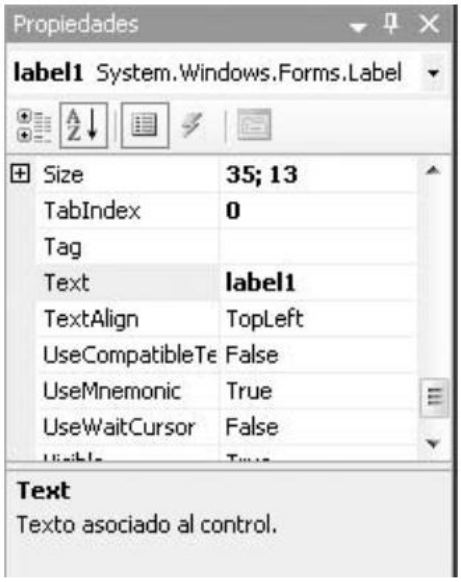




A través de los gráficos que se muestran anteriormente, se puede visualizar el entorno de trabajo en C#, que es un lenguaje de programación basado en C, con la salvedad de ser orientado a objetos y visual, de tal manera que se puedan observar cada uno de los cambios que se realizan en el formulario por parte del programador.



Se agregó una etiqueta al formulario.



Propiedades de la etiqueta.

Como se muestra en los gráficos anteriores, se puede observar que después de ingresar al entorno de visual c#, seleccionar Windows form application y comenzar a trabajar con el programa, se selecciona el formulario para comenzar a programarlo, agregando las diversas herramientas existentes en el cuadro de herramientas y darle vida al formulario.

Desplácese hasta la propiedad **Text** y sustituya el texto **label1** por **Hola Mundo**.

• Ejecutemos ahora el programa, haciendo clic en la flecha que se encuentra en la parte superior del IDE (Figura); se trata del botón Iniciar depuración.

Aparecerá una nueva ventana, como se muestra en la Figura. Es el programa que usted ha creado. Sólo muestra algo de texto, pero es una verdadera ventana en cuanto a que puede moverla, cambiar su tamaño y cerrarla al hacer clic en el icono **X** que se encuentra en la esquina superior derecha.
Experimente con esta ventana, y después ciérrela.
Para guardar su programa y poder usarlo más adelante:
• Vaya al menú **Archivo** y seleccione la opción **Guardar todo** (en adelante, para indicar este tipo de acciones utilizaremos una forma abreviada como ésta: A**rchivo | Guardar todo**).

• A continuación aparecerá la ventana **Guardar proyecto**, como se muestra en la Figura. Asegúrese de que la opción **Crear directorio para la solución** no esté marcada. Deje las demás opciones como están y haga clic en **Guardar**. Cuando vuelva a guardar el proyecto se utilizarán de manera automática las mismas opciones y ya no aparecerá la ventana **Guardar proyecto**.

• Ahora puede utilizar el comando **Archivo | Salir** para cerrar el IDE.

• La próxima vez que utilice C# el nombre de su proyecto aparecerá en la Página de inicio, de manera que podrá abrirlo con un solo clic, sin necesidad de repetir el trabajo que hicimos para configurar el proyecto.

El programa **Primer hola** en ejecución.

PRÁCTICA DE AUTOEVALUACIÓN

2.1 Coloque dos etiquetas en un formulario. Haga las siguientes modificaciones en sus propiedades. Después de realizar cada modificación ejecute el programa, observe el efecto y detenga la ejecución haciendo clic en el botón x en la esquina superior derecha de la ventana.

- Mueva las etiquetas.
- Establezca la propiedad **AutoSize** de una de las etiquetas en **True**.
- Altere sus propiedades **Text** de manera que la información desplegada sean su nombre y edad.
- Modifique sus fuentes utilizando la propiedad **Font**.
- Altere su color de fondo utilizando la propiedad **BackColor**.

2.2 Seleccione el formulario completo. Realice las siguientes tareas, ejecutando el programa después de cada modificación.

- Cambie el tamaño del formulario.
- Altere su propiedad **Text**.
- Altere su propiedad **BackColor**.

El programa que creamos en el ejercicio anterior no es muy representativo, dado que siempre muestra las mismas palabras y el usuario no puede interactuar con él. Enseguida enriqueceremos este programa de manera que aparezca un mensaje de texto cuando el usuario haga clic en un botón. Éste es un ejemplo de cómo usar un *evento*.

Casi todos los eventos son puestos en acción por el usuario, y se generan cuando éste manipula un control de alguna forma en tiempo de ejecución. Cada control tiene varios eventos a los que puede responder, como el clic de un botón del ratón, un doble clic o la colocación del puntero del ratón sobre el control. Otros tipos de eventos no tienen su origen en el usuario; por ejemplo, la notificación de que una página Web ha terminado de descargarse.

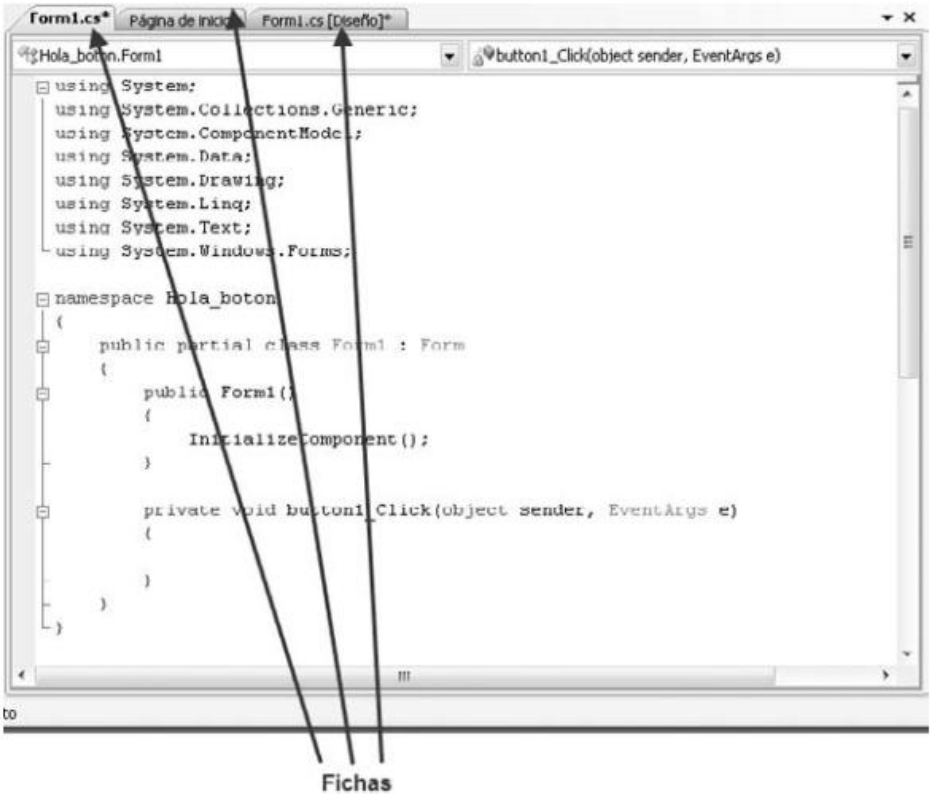
En el programa que crearemos a continuación detectaremos un evento (el clic de un botón); después haremos que se despliegue un mensaje de texto en una etiqueta. He aquí la forma en que crearemos la interfaz de usuario:

- Cree un nuevo proyecto llamado **Botón hola**.
- Coloque una etiqueta y un botón en el formulario. La posición en que lo haga no es importante.
- Escriba **Haga clic aquí** en la propiedad **Text** del botón.
- Modifique la propiedad **Text** de la etiqueta, de forma que aparezca sin contenido.

Ejecute el programa, aunque todavía no está completo. Observe que puede hacer clic en el botón, y que éste cambia su aspecto ligeramente para confirmar que está siendo oprimido; no ocurre nada más. Cierre el formulario.

Veamos ahora cómo detectar el evento del clic. Haga doble clic en el botón dentro del formulario de diseño. A continuación se abrirá un nuevo panel de información, como el que se muestra en la Figura. Observe las fichas de la parte superior:

Form1.cs Página de inicio Form1.cs [Diseño]
Usted puede hacer clic en esas fichas para cambiar de panel. El panel **Form1.cs** muestra un programa de C#. A esto se le conoce como “texto del programa”, o “código” de C#. Modifiquemos este código utilizando el editor del IDE.



Desplácese por el código hasta que encuentre la siguiente sección:
private void button1_Click(object sender, EventArgs e)
{ }

A esta sección de código se le conoce como *método*. El nombre del método es **button1_Click**.
Cuando el usuario haga clic sobre el botón **button1** en tiempo de ejecución, se llevará a cabo, o "ejecutará", cualquier instrucción que coloquemos entre las dos líneas anteriores.

En este programa específico utilizaremos una instrucción para colocar el mensaje **Hola mundo** en el texto de la etiqueta **label1**. La instrucción para realizar esta acción es:
label1.Text = "Hola mundo";

Escriba la instrucción exactamente como se muestra aquí; hágalo entre las líneas **{** y **}**. En los siguientes capítulos explicaremos el significado exacto de líneas como la anterior (que implican una "instrucción de asignación").

El siguiente paso es ejecutar el programa. Para ello hay dos posibilidades:
• Si el programa se ejecuta correctamente, al hacer clic en el botón observará que aparece el mensaje **Hola mundo** en la etiqueta.
La otra posibilidad es que el programa no se ejecute debido a un error. En este caso C# mostrará un mensaje como el de la Figura 2.13. Marque la casilla de verificación de la opción **No volver a mostrar este cuadro de diálogo**, y haga clic en **No** para continuar.
El mensaje no volverá a aparecer; en lo sucesivo la única evidencia del error será un subrayado en el texto del código.

De todas formas hay que corregir el error: revise el código, corrija cualesquiera errores de escritura, y ejecute el programa de nuevo. Más adelante hablaremos sobre los errores con más detalle. He aquí las nuevas características de este programa:

- Puede responder al clic en un botón. Al proceso de colocar código de manera que se lleve a cabo la acción correspondiente cuando ocurra un evento se le conoce como "manejar" el evento.
- Modifica el valor de la propiedad de un control en tiempo de ejecución. Anteriormente sólo hacíamos esto en tiempo de diseño. Ésta es una parte muy importante de la programación, ya que a menudo tenemos que mostrar resultados colocándolos en cierta propiedad de un control.

PRÁCTICA DE AUTOEVALUACIÓN

2.3 Coloque una segunda etiqueta en el formulario. Haga que muestre su nombre cuando se haga clic en el botón.

El Cuadro de Mensajes

Anteriormente utilizamos el control tipo etiqueta para mostrar texto en la pantalla, pero también podemos usar un cuadro de mensajes para lograrlo. Este control no aparece en el cuadro de herramientas, debido a que no ocupa un espacio permanente en un formulario; sólo aparece cuando es requerido. El siguiente es un fragmento de código que muestra el mensaje **Hola mundo** dentro de un cuadro de mensajes al hacer clic en un botón:

```
private void button1_Click (object sender, EventArgs e)
{
MessageBox.Show ("Hola mundo");
}
```

La Figura muestra lo que ocurre al ejecutar el programa y hacer clic en el botón: se despliega el cuadro de mensajes, y debemos hacer clic en el botón **Aceptar** para que desaparezca. Esta característica implica que los cuadros de mensajes se utilizan para mostrar mensajes importantes que el usuario no puede ignorar.

Para utilizar un cuadro de mensajes escriba una línea como la anterior, pero utilice su propio mensaje dentro de las comillas dobles. En este

momento no explicaremos el propósito de la instrucción **Show** ni por qué se requieren los paréntesis. Hablaremos sobre todo esto cuando estudiemos los métodos con más detalle.

PRÁCTICA DE AUTOEVALUACIÓN

2.5 Escriba un programa que tenga un formulario con dos botones. Al hacer clic en un botón deberá aparecer un cuadro con el mensaje `Hola mundo`. Al hacer clic en el otro deberá desplegarse un cuadro con el mensaje `Adiós, mundo cruel`.

Un cuadro de mensajes.

El código siguiente corresponde a un programa de ejemplo llamado Área de rectángulo, mismo que analizaremos con detalle a continuación. Supongamos que las medidas de los lados del rectángulo que nos interesa están representadas en números enteros (**int**). Sólo hay un control en el formulario: un botón con el texto "Calcular" en su propiedad **Text**. Todo el código que agregaremos estará dentro del método **button1_Click**.

```
private void button1_Click(object sender, EventArgs e)
{
int área;
int longitud;
int ancho;
longitud = 20;
ancho = 10;
área = longitud * ancho;
MessageBox.Show("El área es: " + Convert.ToString(área));
}
```

El programa Área de rectángulo.

En los programas en que hemos venido trabajando utilizamos instrucciones de asignación con el propósito de establecer valores iniciales para los cálculos; pero, en la práctica no conoceremos esos valores al escribir el programa, ya que el usuario los introducirá a medida que éste se vaya ejecutando. En esta sección veremos el control **TextBox**, el cual permite que un usuario introduzca datos, y el control **Label** que se utiliza para desplegar información (por ejemplo, los resultados de un cálculo, o instrucciones para el usuario) en un formulario.

Como sabemos, para usar un cuadro de texto todo lo que tenemos que hacer es seleccionarlo en el cuadro de herramientas y colocarlo en un formulario. Estos controles tienen muchas propiedades, pero la principal es **Text**, que nos proporciona la cadena escrita por el usuario. Para acceder a esta propiedad utilizamos la ya conocida notación de "punto", como en el siguiente ejemplo:

```
string s;
s = textBox1.Text;
```

Es bastante común que el programador elimine el contenido de la propiedad **Text** del control en tiempo de diseño (mediante la ventana de propiedades), para que el usuario pueda escribir en un área en blanco. Al igual que en el caso de los cuadros de texto, la principal propiedad del control **Label** (disponible también en el cuadro de herramientas) es **Text**, pues nos permite establecer la cadena que la etiqueta mostrará en pantalla. Podemos acceder a esta propiedad de la siguiente manera:

```
string s = "Alto";
label1.Text = s;
```

Algunas etiquetas se utilizan para mostrar mensajes de ayuda al usuario; por lo general establecemos su propiedad **Text** en tiempo de diseño

mediante la ventana de propiedades. No es necesario que el texto que contienen cambie durante la ejecución del programa. Por otro lado, en el caso de las etiquetas que despliegan resultados hay que establecer su propiedad **Text** en tiempo de ejecución, como se muestra en el ejemplo anterior. El usuario puede sobrescribir los cuadros de texto, pero las etiquetas están protegidas.

En general, las clases tienen métodos y propiedades. Los métodos hacen que los objetos realicen acciones, mientras que las propiedades nos permiten acceder al estado actual de un objeto. He aquí un programa de ejemplo (Dólares y centavos), en el que una cantidad en centavos se convierte a dólares y centavos. Anteriormente en este capítulo vimos cómo usar los operadores / y. En la Figura se muestra este programa en ejecución; en él se utiliza un cuadro de texto y varias etiquetas.

```
private void button1_Click(object sender, EventArgs e)
{
int centavos;
centavos = Convert.ToInt32(textBox1.Text);
dólaresEtiqueta.Text = Convert.ToString(centavos / 100);
centavosEtiqueta.Text = Convert.ToString(centavos % 100);
}
```

El programa Dólares y centavos.

Los principales controles que utilizamos en este programa son:
- un botón para iniciar la conversión;
- un cuadro de texto en donde el usuario introduce una cantidad en centavos;
- dos etiquetas: para mostrar el número de dólares y el número de centavos.

Además hay tres etiquetas debajo del cuadro de texto y las dos etiquetas que muestran el resultado para ayudar al usuario a entender el formulario. Los valores de texto de las etiquetas son:

Introduzca aquí la cantidad de centavos

Dólares

Centavos

Nuestro primer ejemplo es un programa que simula el candado digital de una bóveda de seguridad. En la Figura se muestra la interfaz del mismo. La bóveda se mantiene cerrada hasta que el usuario introduce el código correcto en un cuadro de texto. Al principio este cuadro aparece vacío.

El programa compara el texto introducido con el código correcto. Si son iguales se despliega un mensaje.

Interfaz del programa La bóveda.

```
private void button1_Click(object sender, EventArgs e)
{
string código;
label2.Text = "";
código = textBox1.Text;
if (código == "miguel")
{
label2.Text = "acceso permitido";
}
}
```

La instrucción if prueba el valor de la cadena de texto. Si ésta contiene el valor “miguel”, la instrucción que está entre los corchetes { y } se realiza. Por otro lado, si la cadena es diferente de “miguel” la instrucción entre los corchetes es ignorada y se ejecuta cualquier otra instrucción que esté después.

Tenga en cuenta que la condición a evaluar se encierra entre paréntesis; ésta es una regla gramatical de C#. Observe también que la prueba de igualdad utiliza el operador == (no el operador =).

Algunas veces es necesario especificar dos secuencias de acciones: las que se llevarán a cabo si la condición es verdadera, y las que serán realizadas si es falsa.

El usuario del programa para verificar la capacidad de votar escribe su edad en un cuadro de textoy hace clic en un botón; el programa decide entonces si puede votar o no. La interfaz de este programa se muestra en la Figura. Cuando el usuario hace clic en el botón el programa extrae la información introducida en el cuadro de texto, convierte la cadena en un entero y coloca el número en la variable llamada edad. A continuación se necesita que el programa realice diferentes acciones, dependiendo de si el valor es:

• mayor que 17, o

• igual o menor que 17.

Luego se muestran los resultados de la evaluación en varias etiquetas.

```
private void button1_Click(object sender, EventArgs e)
{
int edad;
edad = Convert.ToInt32(textBox1.Text);
if (edad > 17)
{
etiquetaDecisión.Text = "usted puede votar";
etiquetaComentario.Text = "felicidades";
```

```
}

else
{
etiquetaDecisión.Text = "usted no puede votar";
etiquetaComentario.Text  = "lo siento";
}
etiquetaDespedida.Text = "¡Que gane su candidato!";
}
```

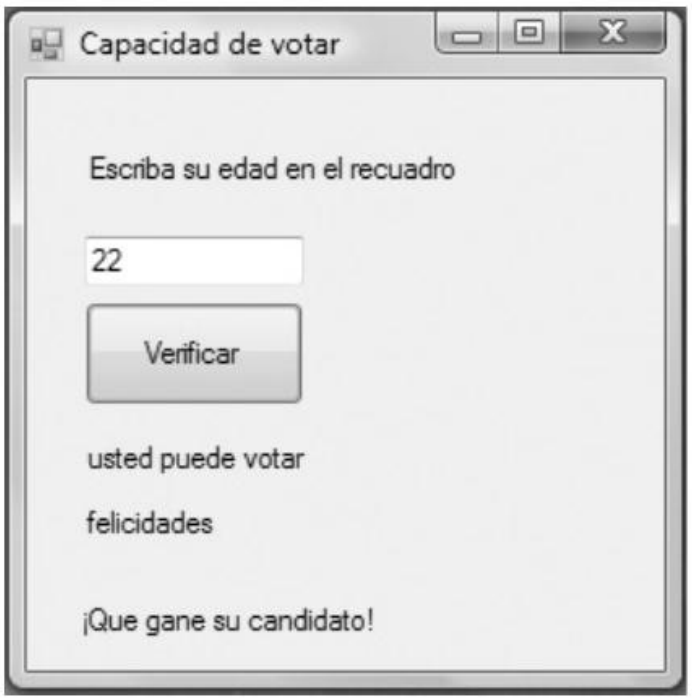
Esta instrucción if consta de tres partes:

• la condición a evaluar, en este caso, si la edad es superior a 17 años;

• la instrucción o secuencia de instrucciones que se ejecutarán si la condición es verdadera; esta parte debe ir encerrada entre corchetes;

• la instrucción o instrucciones a ejecutar si la condición es falsa; esta parte debe ir encerrada entre corchetes.

El nuevo elemento en este código es la palabra clave else, que introduce la segunda parte de la instrucción if. Observe de nuevo cómo la sangría ayuda a enfatizar la intención del programa.



Interfaz del programa para revisar la capacidad de votar.

Esta instrucción constituye otra forma de usar muchas instrucciones if. Usted podrá realizar todo lo que necesite con la ayuda de las instrucciones if, pero switch puede ser una alternativa más eficiente en circunstancias apropiadas. Por ejemplo, suponga que necesitamos una pieza de código para mostrar el día de la semana como una cadena de texto. Imagine que el programa representa el día de la semana como una variable int llamada númeroDía con uno de los valores del 1 al 7 para representar los días de lunes a domingo. Queremos convertir la versión numérica del día en una versión de cadena de texto llamada nombreDía. Para ello podríamos escribir la siguiente serie de instrucciones if:

```
if (númeroDía  == 1)
{
nombreDía  = "Lunes";
}
if (númeroDía  == 2)
{
nombreDía  = "Martes";
```

```
}
if (númeroDía  == 3)
{
nombreDía  = "Miércoles";
}
if (númeroDía  == 4)
{
nombreDía  = "Jueves";
}
if (númeroDía  == 5)
{
nombreDía  = "Viernes";
}
if (númeroDía  == 6)
{
nombreDía  = "Sábado";
}
if (númeroDía  == 7)
{
nombreDía  = "Domingo";
}
```

Aunque la pieza de código que acabamos de proponer es clara y está bien estructurada, hay una alternativa que cumple la misma función utilizando la instrucción switch:

```
switch (númeroDía)
{
case 1:
nombreDía  = "Lunes";
break;
case 2:
nombreDía = "Martes";
```

```
break;
case 3:
nombreDía = "Miércoles";
break;
case 4:
nombreDía = "Jueves";
break;
case 5:
nombreDía = "Viernes";
break;
case 6:
nombreDía = "Sábado";
break;
case 7:
nombreDía = "Domingo";
break;
}
```

La instrucción break transfiere el control al final de la instrucción switch, el cual está marcado con una llave de cierre. Esto nos permite ver con más claridad lo que se va a hacer, en contraste con la serie de instrucciones if equivalente.

Las instrucciones switch como la anterior resultan más comprensibles mediante un diagrama de acción, como el que se muestra en la Figura.

En el ciclo for se agrupan muchos de los ingredientes de los ciclos while en el encabezado de la instrucción. Por ejemplo, veamos el programa anterior para mostrar los números 1 al 10, pero ahora utilizando for:

```
textBox1.Clear();
for (int número = 1; número <= 10; número++)
{
textBox1.AppendText(Convert.ToString(número) + " ");
}
```

Dentro de los paréntesis de la instrucción for hay tres elementos separados por signos de punto y coma:

• una instrucción de inicialización: se lleva a cabo sólo una vez, antes de que inicie el ciclo.

Ejemplo:

```
int número = 1
```

• una condición: se evalúa antes de cualquier ejecución del ciclo. Ejemplo:

```
número <= 10
```

• una instrucción final: ésta se lleva a cabo justo antes del final de cada ciclo. Ejemplo:

```
número++
```

La condición determina si el ciclo for se ejecuta o completa de cualquiera de las siguientes maneras:

• Si la condición es verdadera, se ejecuta el cuerpo del ciclo.

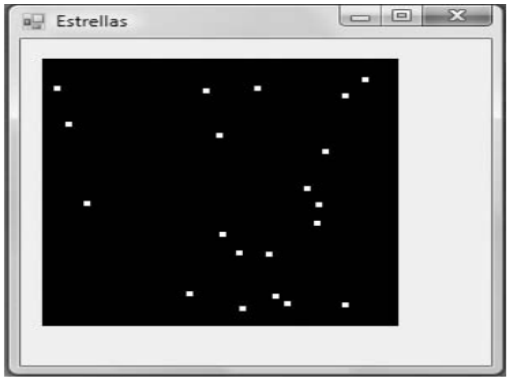
• Si la condición es falsa, el ciclo termina y se ejecutan las instrucciones que van después de la llave de cierre.

Tenga en cuenta que es posible escribir la declaración de una variable dentro de una instrucción for junto con su inicialización, algo que se hace con frecuencia. Esta variable puede utilizarse dentro del cuerpo de la instrucción for.

A continuación presentamos el código de otro programa de ejemplo que utiliza un ciclo for para mostrar veinte círculos pequeños en coordenadas aleatorias dentro de un cuadro de imagen de color negro (Figura).

```
private void pictureBox1_Click(object sender, EventArgs e)
{
Graphics papel;
papel = pictureBox1.CreateGraphics();
papel.Clear(Color.Black);
Random númeroAleatorio = new Random();
SolidBrush miPincel = new SolidBrush(Color.White);
for (int conteo = 0; conteo < 20; conteo++)
{
int x, y, radio;
x = númeroAleatorio.Next(1, 200);
y = númeroAleatorio.Next(1, 200);
radio = 5;
papel.FillEllipse(miPincel, x, y, radio, radio);
}
}
```

Siempre es posible volver a codificar un ciclo for como un ciclo while, y viceversa; sin embargo, por lo general uno de los dos será más conveniente para una aplicación en particular.



Lo primero que haremos es utilizar un ciclo para mostrar los números enteros del 1 al 10 (Figura) en un cuadro de texto, para lo cual usaremos el siguiente código:

```
private void button1_Click(object sender, EventArgs e)
{
int número;
textBox1.Clear();
número = 1;
while (número <= 10)
{
textBox1.AppendText(Convert.ToString(número) + " ");
número++;
}
}
```

La palabra while indica que se requiere la repetición de las instrucciones que están encerradas entre las llaves (lo que se conoce como cuerpo del ciclo). La condición entre paréntesis que va inmediatamente después de la palabra while controla el ciclo. Si la condición se evalúa como verdadera, el ciclo continúa; de lo contrario se da por terminado y el control es transferido a la instrucción que está después de la llave de cierre. En este caso el ciclo continúa mientras cada número sea menor o igual a diez.

Antes de iniciar el ciclo, el valor de número se iguala a 1. Al final del ciclo el valor de número se incrementa una unidad mediante el uso del operador ++ que vimos en un capítulo anterior.

El cuadro de texto se vacía al inicio del programa mediante el método Clear. En cada repetición del ciclo se agrega un número (y un espacio) al cuadro de texto, lo cual se logra con el método AppendText.

Despliegue de números enteros, del 1 al 10.

PRÁCTICAS DE AUTOEVALUACIÓN

8.1 ¿Qué hace el siguiente fragmento de código?

```
int número = 0;
while (número <= 5)
{
    textBox1.AppendText(Convert.ToString(número * número)
                        + " ");
    número++;
}
```

8.2 Escriba un programa que sume (que calcule la suma de) los números del 1 al 100 y la muestre en un cuadro de texto al hacer clic en un botón.

Cuando se utilizan las instrucciones while o for, la evaluación se realiza siempre al principio de la repetición. El ciclo do es una estructura alternativa, en la cual la evaluación se lleva a cabo al final de cada repetición. Esto significa que el ciclo siempre se repite por lo menos una vez. Para ilustrar el uso del ciclo do escribiremos códigos para mostrar los números del 0 al 9 en un cuadro de texto mediante las tres estructuras de ciclo disponibles:

Si usamos while:

```
int conteo;
textBox1.Clear();
conteo = 0;
while (conteo <= 9)
{
textBox1.AppendText(Convert.ToString(conteo) + " ");
conteo++;
}
```

Si empleamos for:

```
textBox1.Clear();
for (int conteo = 0; conteo <=9; conteo++)
{
textBox1.AppendText(Convert.ToString(conteo) + " ");
}
```

Si usamos do (la evaluación se hace al final del ciclo):

```
int conteo;
textBox1.Clear();
conteo = 0;
do
{
textBox1.AppendText(Convert.ToString(conteo) + " ");
conteo++;
}
```

while (conteo < 10);

Veamos ahora un ejemplo que exige la ejecución de un ciclo por lo menos una vez. En los juegos de lotería los números se seleccionan al azar, pero no puede haber dos números iguales. Consideremos una lotería muy pequeña, en la que sólo se seleccionan dos números. Tras obtener el primer número aleatorio necesitamos un segundo número, pero éste no debe ser igual al primero.

En consecuencia, seleccionamos números tantas veces como sea necesario para que sea distinto del primero. He aquí el código resultante de este razonamiento:

```
private void button1_Click(object sender, EventArgs e)
{
Random númeroAleatorio = new Random();
int número1, número2;
número1 = númeroAleatorio.Next(1, 10);
do
{
número2 = númeroAleatorio.Next(1, 10);
}
while (número1 == número2);
label1.Text = "los números son "
+ Convert.ToString(número1) + " y "
+ Convert.ToString(número2);
}
```

Martin Fowler, Addison-Wesley, 1999.

Extreme Programming Explained**, Kent Beck, Addison-Wesley, 2000**.

Este libro complementa nuestros capítulos sobre diseño y prueba. Presenta muchas buenas ideas sobre cómo programar sin estrés.

UML Distilled, Martin Fowler con Kendall Scott, Addison-Wesley, 2000.

UML es la notación dominante para describir programas. En este libro utilizamos esta notación en la medida apropiada. Uno de los libros más simples de todos los que se han escrito sobre UML.

About Face, Alan Cooper, IDG Books, 1995. Alan Cooper es reconocido como la persona que creó el primer IDE de Visual Basic, en donde se podían arrastrar componentes a los formularios de diseño. Este libro contiene sus ideas individuales sobre el diseño de interfaces de usuario. Tal vez no sea una obra que usted lea de principio a fin, pero sus críticas sobre algunas interfaces de usuario y sus sugerencias para mejorar los sistemas harán de su lectura una experiencia reveladora. A diferencia de muchos libros de texto sobre interfaces de usuario, se enfoca en los sistemas Microsoft Windows.

Goto: Superheroes of Software Programming from Fortran to the Internet Age and Beyond, **Steve Lohr, Profile Books, 2002**. Este libro traza la historia de los lenguajes de programación, enfocándose en las contribuciones de los pioneros del software. Cubre temas como Fortran, C, C++, Java, Unix y Visual Basic. Capítulo a capítulo analiza la experiencia de importantes personajes del ámbito, incluye entrevistas con ellos, además de comentarios sobre sus motivaciones y la tecnología que utilizaron.

1. Realizar la carga del lado de un cuadrado, mostrar por pantalla el perímetro del mismo (El perímetro de un cuadrado se calcula multiplicando el valor del lado por cuatro)
2. Escribir un programa en el cual se ingresen cuatro números, calcular e informar la suma de los dos primeros y el producto del tercero y el cuarto.
3. Realizar un programa que lea cuatro valores numéricos e informar su suma y promedio.
4. Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente. Mostrar lo que debe abonar el comprador.
5. Realizar un programa que lea por teclado dos números, si el primero es mayor al segundo informar su suma y diferencia, en caso contrario informar el producto y la división del primero respecto al segundo.
6. Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".
7. Se ingresa por teclado un número positivo de uno o dos dígitos (1..99) mostrar un mensaje indicando si el número tiene uno o dos dígitos. (Tener en cuenta que condición debe cumplirse para tener dos dígitos, un número entero)
8. Se cargan por teclado tres números distintos. Mostrar por pantalla el mayor de ellos.
9. Se ingresa por teclado un valor entero, mostrar una leyenda que indique si el número es positivo, nulo o negativo.
10. Confeccionar un programa que permita cargar un número entero positivo de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2, o 3 cifras. Mostrar un mensaje de error si el número de cifras es mayor.
11. Un postulante a un empleo, realiza un test de capacitación, se obtuvo la siguiente información: cantidad total de preguntas que se le realizaron y la cantidad de preguntas que contestó correctamente. Se pide confeccionar un programa que ingrese los dos datos por teclado e informe el nivel del mismo según el porcentaje de respuestas correctas que ha obtenido, y sabiendo que: Nivel máximo: Porcentaje>=90%, Nivel medio: Porcentaje>=75% y <90%, Nivel regular: Porcentaje>=50% y <75%., Fuera de nivel: Porcentaje<50%.
12. Confeccionar un programa que lea por teclado tres números distintos y nos muestre el mayor La primera estructura condicional es una ESTRUCTURA CONDICIONAL COMPUESTA con una CONDICION COMPUESTA. Podemos leerla de la siguiente forma: Si el contenido de la variable num1 es mayor al contenido de la variable num2 Y si el contenido de la variable num1 es mayor al contenido de la variable num3 entonces la CONDICION COMPUESTA resulta Verdadera.
Si una de las condiciones simples da falso la CONDICION COMPUESTA da Falso y continua por la rama del falso.
Es decir que se mostrará el contenido de num1 si y sólo si num1>num2 y num1>num3.
En caso de ser Falsa la condición, analizamos el contenido de num2 y num3 para ver cual tiene un valor mayor.

En esta segunda estructura condicional no se requieren operadores lógicos al haber una condición simple.

13. Realizar un programa que pida cargar una fecha cualquiera, luego verificar si dicha fecha corresponde a Navidad.
14. Se ingresan tres valores por teclado, si todos son iguales se imprime la suma del primero con el segundo y a este resultado se lo multiplica por el tercero.
15. Se ingresan por teclado tres números, si todos los valores ingresados son menores a 10, imprimir en pantalla la leyenda "Todos los números son menores a diez".
16. Se ingresan por teclado tres números, si al menos uno de los valores ingresados es menor a 10, imprimir en pantalla la leyenda "Alguno de los números es menor a diez".
17. Escribir un programa que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distintos a cero). Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si x > 0 Y y > 0 , 2º Cuadrante: x < 0 Y y > 0, etc.)
18. De un operario se conoce su sueldo y los años de antigüedad. Se pide confeccionar un programa que lea los datos de entrada e informe:
    a) Si el sueldo es inferior a 500 y su antigüedad es igual o superior a 10 años, otorgarle un aumento del 20 %, mostrar el sueldo a pagar.
    b)Si el sueldo es inferior a 500 pero su antigüedad es menor a 10 años, otorgarle un aumento de 5 %.
    c) Si el sueldo es mayor o igual a 500 mostrar el sueldo en pantalla sin cambios.
19. Escribir un programa en el cual: dada una lista de tres valores numéricos distintos se calcule e informe su rango de variación (debe mostrar el mayor y el menor de ellos)
20. Escribir un programa que solicite ingresar 10 notas de alumnos y nos informe cuántos tienen notas mayores o iguales a 7 y cuántos menores.
21. Se ingresan un conjunto de n alturas de personas por teclado. Mostrar la altura promedio de las personas.
22. En una empresa trabajan n empleados cuyos sueldos oscilan entre $100 y $500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuántos empleados cobran entre $100 y $300 y cuántos cobran más de $300. Además el programa deberá informar el importe que gasta la empresa en sueldos al personal.
23. Realizar un programa que imprima 25 términos de la serie 11 - 22 - 33 - 44, etc. (No se ingresan valores por teclado)
24. Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.
25. Realizar un programa que permita cargar dos listas de 15 valores cada una. Informar con un mensaje cual de las dos listas tiene un valor acumulado mayor (mensajes "Lista 1 mayor", "Lista 2 mayor", "Listas iguales")

Tener en cuenta que puede haber dos o más estructuras repetitivas en un algoritmo.

26. Desarrollar un programa que permita cargar n números enteros y luego nos informe cuántos valores fueron pares y cuántos impares. Emplear el operador "%" en la condición de la estructura condicional:

```
If valor Mod 2=0 Then          //Si el if da
verdadero luego es par.
```

27. Confeccionar un programa que lea n pares de datos, cada par de datos corresponde a la medida de la base y la altura de un triángulo. El programa deberá informar:
    a) De cada triángulo la medida de su base, su altura y su superficie.
    b) La cantidad de triángulos cuya superficie es mayor a 12.
28. Desarrollar un programa que solicite la carga de 10 números e imprima la suma de los últimos 5 valores ingresados.
29. Desarrollar un programa que muestre la tabla de multiplicar del 5 (del 5 al 50)
30. Confeccionar un programa que permita ingresar un valor del 1 al 10 y nos muestre la tabla de multiplicar del mismo (los primeros 12 términos) Ejemplo: Si ingreso 3 deberá aparecer en pantalla los valores 3, 6, 9, hasta el 36.
31. Realizar un programa que lea los lados de n triángulos, e informar:
    a) De cada uno de ellos, qué tipo de triángulo es: equilátero (tres lados iguales), isósceles (dos lados iguales), o escaleno (ningún lado igual)
    b) Cantidad de triángulos de cada tipo.
32. Escribir un programa que pida ingresar coordenadas (x,y) que representan puntos en el plano.
    Informar cuántos puntos se han ingresado en el primer, segundo, tercer y cuarto cuadrante. Al comenzar el programa se pide que se ingrese la cantidad de puntos a procesar.
33. Se realiza la carga de 10 valores enteros por teclado. Se desea conocer:
    a) La cantidad de valores ingresados negativos.
    b) La cantidad de valores ingresados positivos.
    c) La cantidad de múltiplos de 15.
    d) El valor acumulado de los números ingresados que son pares.
34. Se cuenta con la siguiente información:
    Las edades de 50 estudiantes del turno mañana.
    Las edades de 60 estudiantes del turno tarde.
    Las edades de 110 estudiantes del turno noche.
    Las edades de cada estudiante deben ingresarse por teclado.

    a) Obtener el promedio de las edades de cada turno (tres promedios)
    b) Imprimir dichos promedios (promedio de cada turno)
    c) Mostrar por pantalla un mensaje que indique cual de los tres turnos tiene un promedio de edades mayor.

LINKS DE MANUALES VARIOS

https://www.youtube.com/watch?v=L-f8u0hwi4Y

https://www.youtube.com/watch?v=axHut2e84fc

https://www.youtube.com/watch?v=j8sxDnr7nPY

https://www.youtube.com/watch?v=8O7PopSscBE&list=PL8qxzfBmzgexdFa0XZZSZZn2Ogx3j-Qd5

https://www.youtube.com/watch?v=--_guc2bYIY&list=PL7tWmaH04EfLjESN9pYsGhkphQkX3HSDP

https://www.youtube.com/watch?v=TqiysLEBZo4