Tema: PROCEDIMIENTOS Y FUNCIONES – LENGUAJE C#

Objetivo:

- ✓ Conocer las diferentes posibilidades, alternativas y aplicaciones de los procedimientos y funciones en el lenguaje de Programación "C#"

I. SUBRUTINAS DE CÓDIGO

Se denomina subrutina a una porción de código que tiene como principal función ejecutar una determinada tarea. Estas tienen un nombre que las identifica, con dicho nombre se les puede llamar y así ejecutar la tarea programada. Las subrutinas tienen un principio y un fin y estas pueden ser de tipo privadas o públicas.

En el siguiente ejemplo se muestra una subrutina que crea C# automáticamente cuando en un formulario insertamos un botón llamado *Button*1 y desde la ventana de eventos seleccionamos el evento click del mismo

```
private void button1_Click(object sender, EventArgs e)
    {

    }
```

En este ejemplo se tiene una subrutina o procedimiento de tipo Privado, esto quiere decir que la podemos utilizar solo en el formulario o módulo donde está declarada.. En este caso lel procedimiento de llama *Button1_click(…)* . Si quisiéramos que este procedimientos se pueda llamar desde otro formulario, lo que tendríamos que hacer es cambiar el alcance del procedimiento, en vez de *private* cambiarlo por *public.* Si hacemos esto, el procedimiento puede ser utilizado desde cualquier parte del proyecto.

II. SUBRUTINAS O PROCEDIMIENTOS PROPIOS

También podemos crear nuestros propios Procedimientos o Subrutinas de código para ejecutar una determinada tarea. Para crear un procedimiento de código se debe escribir la palabra **private void** o **public void** (depende el alcance), seguida del nombre del procedimiento y parámetros de entrada que son opcionales. Se debe tener en cuenta que nunca se puede crear un procedimiento dentro de otro. Un ejemplo para crear un procedimiento seria:

```
private void Borrar()
      {
            label1.Text = "";
            label2.Text = "";
            label3.Text = "";
       }
```

En este ejemplo creamos un procedimiento muy simple que tendrá la función específica de borrar el contenido de 3 controles tipo label. Para que se ejecute el procedimiento que se ha creado solo basta con escribir el nombre del procedimiento en el lugar donde queremos que se ejecute. Por *ejemplo:*

```
private void button1_Click(object sender, EventArgs e)
    {
            Borrar();
    }
```

Al presionar el Button1 C# detectaría el procedimiento llamado “Borrar()”, y automáticamente saltaría al lugar donde creamos el procedimiento y ejecutaría las líneas de código programadas en dicho procedimiento. Si nuestro procedimiento contiene parámetros, debemos pasarlos entre paréntesis. Una vez que se ha terminado de ejecutar las instrucciones que estén dentro del bloque del procedimiento, volvería a la línea siguiente desde donde fue llamado, y ejecutaría todas las líneas restantes de código. En este caso no ejecutaría nada más porque no hemos puesto ninguna otra instrucción debajo del Procedimiento “Borrar()”.

La principal ventaja de crear procedimientos de código es que evita tener que escribir varias veces las mismas instrucciones en un programa, el código se hace mucho más funcional y entendible, se pueden dividir un problema (una rutina), en varios procedimientos y probarlos independientemente, y además la posibilidad de enviar parámetros de entrada.

2.1 – PARÁMETROS DE ENTRADA EN PROCEDIMIENTOS:

Los parámetros de entrada también conocidos como argumentos, Se utilizan para que el procedimiento no ejecute siempre las mismas líneas de código, teniendo la posibilidad de programar las tareas de un procedimiento de manera más dinámica. En ciertas ocasiones no es necesario enviar parámetros, pero en otras es prácticamente fundamental. Los parámetros son datos, casi de cualquier tipo, que se pasan seguido del nombre del procedimiento, si son más de uno deben ir separados por comas (,). Por Ejemplo:

```
DatosPersonales("Pedro Picapiedra", 25, "Bogota D.C")
```

En este ejemplo se pasan tres parámetros: El primero es un nombre y es un dato de tipo string y por ende debe ir entre comillas. El segundo es un número y el tercero también un string. Cuando se va a pasar parámetros a un procedimiento, al momento de crear el procedimiento se debe establecer qué tipo de parámetros va a recibir el mismo. La definición de procedimiento de este *ejemplo seria:*

```
private void DatosPersonales(string nombre, int edad, string ciudad)
     {
        label1.Text = nombre;
        label2.Text = edad.ToString();
        label3.Text = ciudad;
     }
```

En el ejemplo anterior se ha definido un procedimiento con *tres parámetros de entrada*. Cada parámetro se declara similar a como se hace con las variables, es decir, se les debe dar un nombre y definir el tipo de dato que recibirá cuando se llame.

Una vez se ha definido el procedimiento, en el evento click de cualquier Button, se puede llamar al procedimiento y pasarle los parámetros. Por Ejemplo:

```
private void button2_Click(object sender, EventArgs e)
        {
            DatosPersonales("Pedro Picapiedra", 25, "Bogota D.C");
        }
```

Al presionar el botón se llamará al procedimiento que se ha sido creado anteriormente, y como se puede opbservar se le pasan los valores de los parámetros en el orden en que están declarados en el procedimiento. Esto quiere decir que el primer parámetro con el valor "`Pedro Picapiedra`" se almacenará o asignará al parámetro nombre, el valor 25 se le asignará al parámetro edad y el último valor al parámetro ciudad. Después que los parámetros de entrada ya fueron asignados sus valores se pueden utilizar dentro del procedimiento. En el ejemplo anterior se le asigna a un control Label1 el contenido del parámetro nombre, al Label2 el contenido de edad y al Label3 el contenido del parámetro ciudad.

Es muy importante respetar el orden en que se pasan los parámetros en un procedimiento, por ejemplo en el caso anterior si hubiésemos pasado los parámetros de esta forma:

```
DatosPersonales(25, "Pedro Picapiedra", "Bogota D.C")
```

El segundo parámetro "`Pedro Picapiedra`" se almacenaría en el parámetro Edad que es de tipo int y se produciría un error en tiempo de ejecución por no coincidir los tipos de datos, ya que la variable espera un valor numérico de tipo int y se le está pasando una cadena de caracteres de tipo string.

Otra aspecto importante es que cuando se requiera crear, por ejemplo un procedimiento que va a recibir 2 parámetros, cuando se llame, no se puede enviar solo 1 parámetro, se debe obligatoriamente pasarle los 2 parámetros que hubiésemos declarado en el. Un ejemplo que daría un error por no pasar adecuadamente los parámetros sería:

```
private void Sumar(int a, int b)
      {   int c;
          c = a + b;
          label1.Text = a + " + " + b + " = " + c.ToString();
      }

private void button3_Click(object sender, EventArgs e)
        {
            Sumar(300); //Forma Incorrecta
            Sumar(600, 400); // Forma Correcta
        }
```

Se crea un procedimiento llamado sumar que recibirá 2 parámetros de tipo int, si se llama al procedimiento y le pasamos un solo valor, por ejemplo: `Sumar(300),` Esto daría un error de compilación por que el procedimiento espera recibir 2 parámetros. La forma correcta debería ser con dos parámetros de entradas como por ejemplo: `Sumar(600, 400);`

2.2 - PARÁMETROS POR VALOR Y POR REFERENCIA

Los parámetros en las funciones y procedimientos, se pueden enviar de dos maneras: por Valor y por Referencia. La diferencia entre uno y otra forma de paso de parámetros, es que *"Por Valor"* se envía una copia de la variable, de tal manera que si se efectúa un cambio en el procedimiento, solo tendrá efecto dentro del procedimiento o función, una vez que termine y finalice el mismo, la variable original pasará a valer el dato que tenía, es decir este no se modifica.

Por otra lado en los parámetros enviados "*Por Referencia"*, lo que se hace es enviar un puntero de la variable original, entonces, si en la función o procedimiento se cambia el valor de la variable, el cambio seguirá manteniéndose una vez que finalice la ejecución del procedimiento o función.

2.3 - EJEMPLO DE ENVÍO DE PARÁMETROS POR VALOR

Código del ejemplo:

```
private void Sumar(long Valor)
  {
  //Modifica el valor del parametro de entrada
  Valor = Valor + 100;
  MessageBox.Show("Valor Dentro del Procedimiento: " + Valor);
  }

private void button1_Click(object sender, EventArgs e)
  {
  long Valor;
  Valor = 100;
  //Se envía la variable por Valor
  Sumar(Valor);
  //Muestra el valor=100 (no se modificó dentro del procedimiento)
  MessageBox.Show("Valor Fuera del Procedimiento: " + Valor);
  }
```

En el ejemplo anterior hay un procedimiento o subrutina llamada Sumar que recibe como entrada un parámetro enviado por valor (es decir es una copia de la variable original) al presionar el Button1. Al entrar en la subrutina Sumar, el dato se modifica (Valor = Valor + 100). Cuando finaliza el procedimiento Sumar y retorna a la línea siguiente de la llamada a la Subrutina sumar, muestra mediante un "`MessageBox`" el valor de la misma, en este caso es 100, y No 200 que es el cambio que tuvo dentro del procedimiento Sumar (Valor = Valor + 100).

Con esto queda visto que en realidad al enviar el parametro por valor, se envía una copia de la variable original y cualquier cambio que se produzca, será solo en el ámbito del procedimiento o función

2.3 - EJEMPLO DE ENVÍO DE PARÁMETROS POR REFERENCIA

Este ejemplo es igual que el anterior, pero en la Subrutina sumar, la variable Valor se declara por Referencia

```
private void Sumar2(ref long Valor)
        {
        //se Modifica el valor del parámetro de entrada
        Valor = Valor + 100;
        MessageBox.Show("Valor Dentro del Procedimiento: " + Valor);
        }

private void button2_Click(object sender, EventArgs e)
        {
         long Valor;
        Valor = 100;
        //Se envía la variable por Referencia con el valor 100
        Sumar2(ref Valor);
        //Muestra el valor=200, ya que se modificó dentro de Sumar2
        MessageBox.Show("Valor Fuera del Procedimiento: " + Valor);
        }
```

Al probar el código, ahora el `MessageBox` al mostrar el valor de la variable, es de 200, y No de 100 como en el ejemplo por valor. Esto demuestra que al enviar la variable como referencia, si la misma se modifica dentro del procedimiento o función, se está modificando la variable REAL, es decir cualquier cambio que se le haga a la misma, se mantendrá, ya que se está modificando la variable Real , es decir. no es una copia como en el caso por valor

III. - FUNCIONES EN LENGUAJE C#

Las funciones son prácticamente iguales que los procedimientos con respecto a la forma en que se les llama, se les crea o declara en el código, en cómo se le pasa los parámetros etc.. La diferencia fundamental con respecto a los procedimientos o subrutinas es que estos, luego de ejecutar el código que tengan en su interior, al final retornan un valor, y este valor luego lo podemos utilizar para una determinada tarea. En cambio los procedimientos, solo ejecutan el código que contienen y luego Terminan. Un ejemplo de cómo crear o definir una función seria:

```
private long Multiplicar(int Valor1, int Valor2)
        {
            long Mult;
            Mult = Valor1 * Valor2;
            return Mult;
        }
```

Se ha declarado una función llamada Multiplicar cuyo tipo de retorno es long, es decir, que en este caso la función al finalizar su ejecución devolverá un valor de tipo long, el cual se pude utilizar en cualquier otra parte del programa. Las Funciones pueden retornar cualquier tipo de dato, como números, cadenas, fechas, arreglos (vectores y/o matrices). Las funciones pueden o no recibir parámetros de entrada. Para llamar a la función del ejemplo anterior podríamos hacerlo de la siguiente manera, tomando como base un formulario con dos textbox, un boton y un label.

```
  private void button1_Click(object sender, EventArgs e)
     {
       long multiplicacion;
       int num1, num2;
```

```
        num1= int.Parse(textBox1.Text);
        num2 = int.Parse(textBox2.Text);

        multiplicacion = Multiplicar(num1,num2);
        label1.Text = num1 + "x" + num2 + "=" + multiplicacion.ToString();
        MessageBox.Show("La multiplicacion es " + multiplicacion.ToString());
      }
```

En este caso al llamar la función Multiplicar, se realiza la multiplicación de los valores enviados como parámetros de entrada y luego el valor final de la operación retornaria y se le asigna a la variable "multiplicacion" para finalmente mostrar los resultados en un label y en un `MessageBox.`

III. MENUS Y SUBMENUS EN UNA INTERFAZ GRÁFICA DE USUARIO (I.G.U)

Los menús y submenús en una interfaz gráfica de usuario (I.G.U) mejoran considerablemente la usabilidad de un programa, ya que le permite al usuario navegar por diferentes opciones y seleccionar de manera rápida y eficiente la funcionalidad que desea ejecutar o el formulario que dese visualizar.

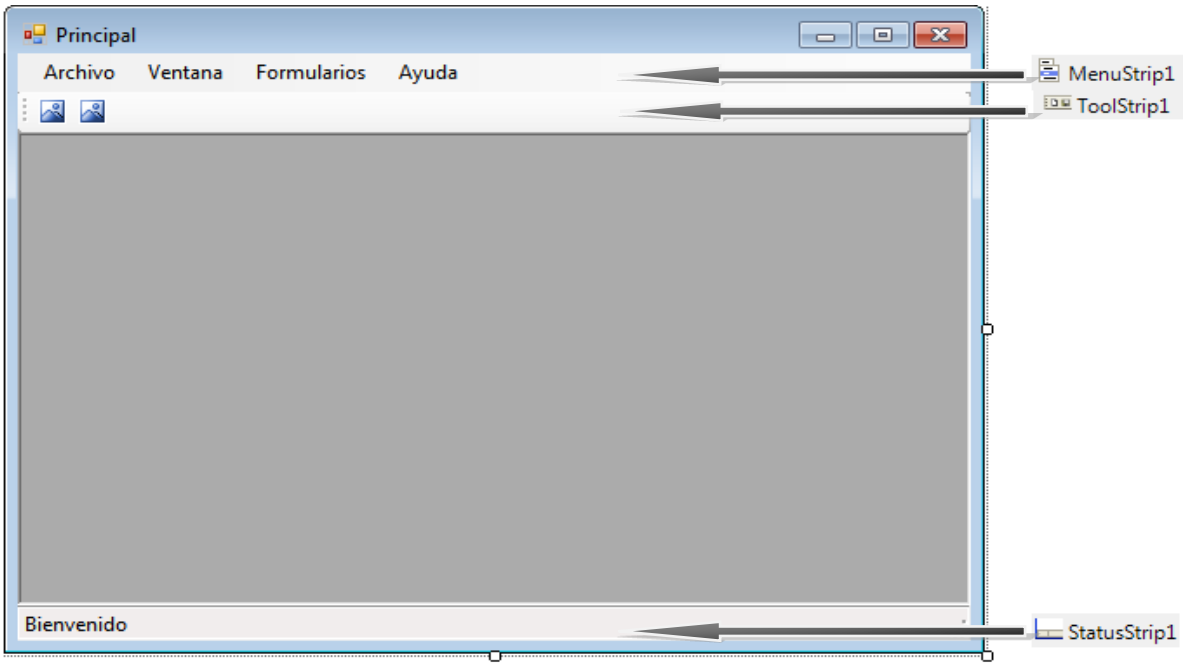
Las Aplicaciones tipo Windows pueden tener varios formularios donde se distribuyen las diferentes tareas o funcionalidades, y una forma muy eficiente de dar acceso a ellos es por medio de un formulario principal (MDI) que contenga un menú con enlaces a los formularios que a su vez se comportarían como formularios hijos.

EJERCICIO:

1. Cree un nuevo proyecto tipo "Windows Forms" C#, agregue un formulario nuevo y colóquele "Principal" como nombre.

2. En tiempo de diseño, diseñe su formulario principal tal y como se muestra a continuación: (Use los controles MenuStrip, ToolStrip y StatusStrip para crear las barras en el formulario)

3. Cambie el valor la propiedad del formulario Principal llamada "IsMdiContainer" de false a true



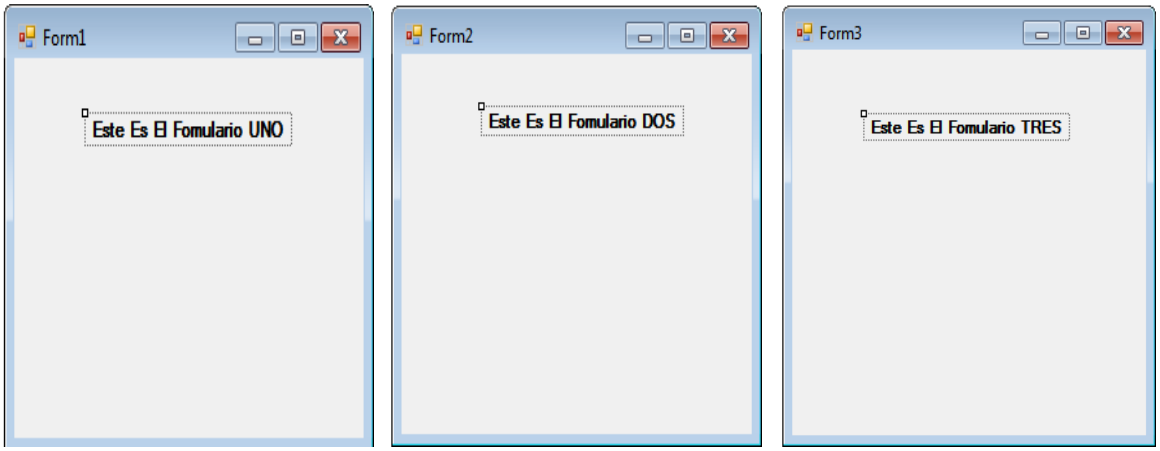
4. Dentro de la Barra de estado “StatusStrip” ubicada en el inferior de la ventana active un label de tipo “toolStripStatusLabel” seleccionando desde las opciones disponibles

5. En la barra de herramientas “ToolStrip” busque imágenes de su preferencia y establézcalas como iconos de las opciones en la barra.

6. Configure el “MenuStrip para que muestre el menú de opciones con la siguiente distribución:

1. Menú Archivo
   - Salir
2. Menú Ventana
   - Cascada
   - Mosaico Vertical
   - Mosaico Horizontal
   - Cerrar Todo
3. Menú Formularios
   - ✓ Fomulario1
   - ✓ Fomulario2
   - ✓ Formulario3
4. Menú Ayuda
   - ✓ Acerca De

7. Agregar al proyecto tres formularios de nombre Form1, Form2, Form3 respectivamente donde por medio de un label indique el nombre del formulario tal y como se muestra a continuación:



8. Agregar un formulario llamado “FormAcercaDe” donde debe mostrar los nombres, los apellidos y la identificación de la persona que está realizando el ejercicio.

9. Presione doble click sobre el formulario principal y escriba el siguiente código:

Nota: Tenga en cuenta que puede acceder al código de cada menú haciendo doble click en ellos.

```
public partial class Principal : Form
    {
        public Principal()
        {
            InitializeComponent();
        }

private void Principal_Load(object sender, EventArgs e)
        {
            toolStripStatusLabel1.Text = "Bienvenido , Hoy es " +
            DateTime.Now.ToShortDateString();
        }

private void cascadaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LayoutMdi(MdiLayout.Cascade);
        }

private void mosaicoVerticalToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            LayoutMdi(MdiLayout.TileVertical);
        }
private void mosaicoHorizontalToolStripMenuItem_Click(object sender,
EventArgs e)
        {
             LayoutMdi(MdiLayout.TileHorizontal);
        }

private void cerrarTodoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            foreach (Form childForm in MdiChildren)
            {
                childForm.Close();
            }
        }

private void salirToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }


private void fomulario1ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form1 Fomulario1 = new Form1();
            Fomulario1.MdiParent = this;
            Fomulario1.Show();
        }
```

```
private void fomulario2ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 Fomulario2 = new Form2();
            Fomulario2.MdiParent = this;
            Fomulario2.Show();
        }

private void fomulario3ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form3 Fomulario3 = new Form3();
            Fomulario3.MdiParent = this;
            Fomulario3.Show();
        }


private void acercaDeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            FormAcercaDe FormularioAcercaDe = new FormAcercaDe();
            FormularioAcercaDe.MdiParent = this;
            FormularioAcercaDe.Show();
        }
}
```

¿COMO CONFIGURAR EL FORMULARIO DE INICIO DEL PROYECTO?

Para establecer cuál es el formulario de inicio, es decir, el primer formulario que aparecerá inmediatamente se ejecute el programa, diríjase al archivo llamado `Program.cs,` y dentro de este ubique la línea de código `"Application.Run(...)"` y dentro de los paréntesis escriba el nombre del formulario que desea establecer como fomulario de inicio.

Por ejemplo para establecer como formulario de inicio un formulario llamado “Principal” la línea de código dentro del archivo `Program.cs` seria;

```
static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Principal());
        }
```

TALLER

Realizar una única aplicación tipo Windows que tenga varios formularios. Debe tener un formulario principal que por medio de un menú proporcione acceso de forma independiente a cada uno de los otros formularios de la solución de los problemas planteados a continuación. Para la solución de los problemas debe ***Aplicar Procedimientos y/o Funciones***

Problema 1: ***(****Usar Un Procedimiento llamado "DefinirSaludo")*

Diseñe un formulario que permita ingresar el nombre del usuario y la hora (formato 24 horas) y salude al usuario dependiendo de la hora ingresada así:

a.) Buenos Días (0-11 horas)
b.) Buenas Tardes (12-18 horas)
c.) Buenas Noches (19-24 horas).

Ejemplo: Nombre: Juan, hora: 19 → BUENAS NOCHES JUAN.

Problema 2: ***(****Usar Procedimiento "GenerarTabla")*

Diseñe un formulario que permita ingresar dos números y mostrar la tabla de multiplicar del primero hasta el valor que indique el segundo.

Problema 3: *(Usar Función "CalcularPromedio" y Procedimiento "DefinirAsignatura ")*

Diseñe un formulario que permita ingresar el nombre de una asignatura, el nombre del estudiante y tres notas en un rango de 1 a 5 (Debe Validar Las Entradas). Se debe permitir calcular el promedio de las notas y determinar si el estudiante aprobada o no la asignatura; sabiendo que una asignatura se pierde con una nota promedio menor a 4.0. El resultado a mostrar debe además contener el nombre del estudiante, el nombre de la asignatura y la nota promedio calculada.

Problema 4: (Usar una Función para cada figura Geométrica)

Diseñe un formulario que permita seleccionar una figura geométrica y calcularle el área o volumen dependiendo de la selección. Las figuras geométricas deben ser: Circulo, triangulo, cuadrado, rombo, rectángulo, Trapecio, cubo, cilindro, cono y esfera.

Problema 5: *(Usar Función "CalcularValorAPagar")*

Un almacén de cadena se encuentra de aniversario y ha programado una serie de ofertas con la finalidad de brindar facilidades a sus clientes y al a vez de incrementar sus ventas. Estas ofertas se basan específicamente en un porcentaje de descuento sobre el total de compra el cual varía de acuerdo al monto:

a.) Por un monto menor o igual a $5000 se hará un descuento del 10%
b) Por un monto mayor de $5000 pero menor o igual a $20.000 se hará un descuento del 15%
c) Por un monto mayor de $20.000 pero menor o igual a $100.000 se hará un descuento del 25%
d) Por un monto mayor a $100.000 se hará un descuento del 40%

Diseñe un formulario donde se ingrese el monto de la compra y se muestre el valor del descuesto y el valor que finalmente pagara el cliente. Se debe agregar el valor del IVA (16%)