LECTURA 7.1
ENVÍO DE PARÁMETROS A UN MÉTODO: POR VALOR, POR REFERENCIA Y PARÁMETROS DE SALIDA EN C#

La recursividad basa su funcionamiento en el uso correcto del ámbito de las variables y para entenderla mejor es fundamental tener presente la manera de controlar las variables globales y locales.

Regularmente los algoritmos recursivos se apoyan en el envío de parámetros durante las llamadas recursivas de sus métodos, por eso, es importante resaltar que existen dos maneras de enviar parámetros ó argumentos a un método: por valor y por referencia.

7.6.1 Parámetros por valor

Cuando se envía un parámetro por valor, se hace una copia del valor de la variable enviada en el parámetro recibido, el cual actúa como una variable local, que lo recibe, lo manipula dentro del método y que desaparece y pierde su valor al terminar la ejecución de ese segmento de código.

Para ilustrarlo mejor, consideremos el siguiente ejemplo: Se declara una variable global denominada x y se inicializa con el valor 5 (x=5). Dentro del programa principal se declara y se inicializa una variable local llamada (y) con el valor 13 (y=13). Cuando se hace la llamada a un `METODO(y)` y se envía la variable (y), se hace por valor, es decir, se envía una copia del valor de la variable (13) que lo recibe otra variable local (a). En ese momento, se transfiere el control de la ejecución del programa hacia el método, activando su variable local (a) y desactivando momentáneamente la variable local del programa principal (y) (la variable x permanece activa ya que se trata de una variable global). Dentro del método, se modifica el valor de su variable local (a), se imprime (16) y se duplica el valor de la variable global (x) y se imprime (10). Cuando el método termina, el sistema desactiva su variable local (a), regresa el control de la ejecución del programa al lugar donde se hizo la llamada, activa y recupera el valor de su propia variable local (y=13) y continua con su operación. Al imprimir los valores,

nos percatamos que la variable x modificó su valor por tratarse de una variable global que puede ser accedida en cualquier parte del programa, sin embargo, la variable y mantiene su valor original (no fue alterado), ya que, por tratarse de una variable local, fue desactivada al llamar al `Metodo()` y reactivada con su valor original al retornar.

El siguiente pseudocódigo ilustra este ejemplo:

```
x = 5  // Declaración de variable global

PROGRAMA_PRINCIPAL
1.- y = 13  // Declaración de variable local
2.- IMPRIMIR x  // Despliega 5
3.- METODO(y) // Llamada del método y le envía la variable
“y” por valor
4.- IMPRIMIR x  // Despliega 10
5.- IMPRIMIR y  // Despliega 13
6.- FIN

// Declaración del método con el parámetro local “a”
METODO(a)
7.- a = a + 3
8.- IMPRIMIR a  // Despliega 16
9.- x = x * 2
10.- RETURN
```

Es importante mencionar que las variables que reciben los parámetros de un método son variables locales e incluso pueden tener el mismo nombre que la variable origen, sin embargo se trata de copias de los valores originales, por lo tanto, son variables diferentes.

Durante las llamadas recursivas, se hacen copias de los valores de las variables locales de cada método (incluyendo sus parámetros), razón por la cual, mantiene sus valores solamente en la etapa que esté ejecutando y que, cuando se termina y regresa el control a la ejecución de la llamada anterior, recupera los valores que tenía en dicha etapa.

7.6.1.1 Prog. 7.1.- Proyecto de consola en C#: Envío de parámetros por valor

Para enviar un parámetro por valor en C#, basta con escribirlo entre los paréntesis de la llamada del método, además la variable que lo recibe debe tener el tipo de dato correspondiente. A continuación se muestra el código del Prog. 7.1 con una aplicación de consola en C# que ilustra el ejemplo anterior con el envío de parámetros por valor.

Prog. 7.1. Envío de parámetros por valor (Program.cs).

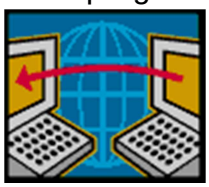
```
class Program
    {
        static int x = 5;  // Variable global

        static void Main(string[] args)
        {
            int y = 13;  // Variable local

            Console.WriteLine("\nx=" + x.ToString());

            Metodo(y);  // Llamada al método y envío por valor

            Console.WriteLine("\nx=" + x.ToString());

            Console.WriteLine("\ny=" + y.ToString());

            Console.ReadKey();
        }

        static void Metodo(int a)  // El parámetro "a" recibe el valor de "y"
        {
            a = a + 3;
            Console.WriteLine("\na=" + a.ToString());

            x = x * 2;
        }
    }
```

Al ejecutar el programa anterior se produce la salida mostrada en la Fig. 7.2.

```
x=5

a=16

x=10

y=13
```

Fig. 7.2. Salida del programa de envío de parámetros por valor.

El programa completo puede descargarse del siguiente sitio web:



`http://www.itnuevolaredo.edu.mx/takeyas/libroED/Prog7-1.rar`

7.6.2 Parámetros por referencia

Cuando se envía un argumento por referencia, no se hace una copia de su valor y quien lo recibe, no contiene directamente el dato, sino una referencia a él (Fig. 7.3).

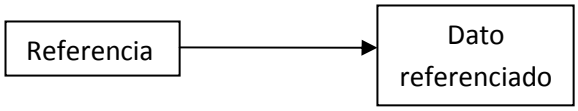


Fig. 7.3. Referencia a un dato.

El envío de parámetros por referencia permite a los miembros de una clase cambiar el valor del parámetro y mantener vigente dicho cambio. Cuando una variable es enviada por referencia, el método recibe la referencia de la variable original y esto implica que los cambios realizados a esa variable dentro del método, afecta la variable original.

Para ilustrarlo mejor, consideremos el ejemplo de la sección anterior, pero ahora enviando la variable (`y`) por referencia: Se declara una variable global denominada x y se inicializa con el valor 5 (x=5). Dentro del programa principal se declara y se inicializa la variable (`y`) con el valor 13 (y=13), la cual se considera variable local al ser declarada dentro de un método. Cuando se hace la llamada del `METODO(referencia y)` y se envía la variable y, se hace por referencia, es decir, se envía la referencia (no el valor) de la variable y, que lo recibe la variable local a. Dentro del método, se modifica el valor de la variable a, se imprime (16) y se duplica el valor de la variable global x (10) y se imprime. Cuando el método termina, el sistema desactiva sus variables locales (a), regresa el control de la ejecución del programa al

lugar donde se hizo la llamada, activa y recupera los valores de sus propias variables locales (y=13) y continua con su operación. Al imprimir los valores, nos percatamos que la variable x modificó su valor por tratarse de una variable global que puede ser accedida en cualquier parte del programa y la variable y, aunque se trata de una variable local, también modificó su valor, ya que siendo la variable a una referencia de la variable y, entonces al modificar a, también se modifica y.

El siguiente pseudocódigo ilustra este ejemplo:

```
 x = 5  // Declaración de variable global

 PROGRAMA_PRINCIPAL
 1.- y = 13  // Declaración de variable local
 2.- IMPRIMIR x  // Despliega 5
 3.- METODO(referencia y) // Llamada del método y le envía la
variable “y” por referencia
 4.- IMPRIMIR x  // Despliega 10
 5.- IMPRIMIR y  // Despliega 16
 6.- FIN

 // Declaración del método con el parámetro local “a” por
referencia
 METODO(referencia a)
 7.- a = a + 3
 8.- IMPRIMIR a  // Despliega 16
 9.- x = x * 2
 10.- RETURN
```

7.6.2.1 Prog. 7.2.- Proyecto de consola en C#: Envío de parámetros por referencia

Para enviar un parámetro por referencia en C#, se debe anteponer la palabra *ref* al nombre de la variable enviada (colocada entre los paréntesis de la llamada del método), además la variable que lo recibe también debe tener la palabra *ref*, seguido del tipo de dato correspondiente. A continuación se muestra el código del Prog. 7.2

con una aplicación de consola en C# que ilustra el ejemplo anterior con el envío de parámetros por referencia.

Prog. 7.2. Envío de parámetros por referencia (Program.cs).

```
class Program
    {
        static int x = 5;  // Variable global

        static void Main(string[] args)
        {
            int y = 13;  // Variable local

            Console.WriteLine("\nx=" + x.ToString());

            Metodo(ref y);  // Llamada al método y envío por referencia

            Console.WriteLine("\nx=" + x.ToString());

            Console.WriteLine("\ny=" + y.ToString());

            Console.ReadKey();
        }

        static void Metodo(ref int a)  // El parámetro "a" recibe la referencia de "y"
        {
            a = a + 3;
            Console.WriteLine("\na=" + a.ToString());

            x = x * 2;
        }
    }
```

Al ejecutar el programa anterior se produce la salida mostrada en la Fig. 7.4.

```
x=5

a=16

x=10

y=16
```

Fig. 7.4. Salida del programa de envío de parámetros por referencia.

El programa completo puede descargarse del siguiente sitio web:

`http://www.itnuevolaredo.edu.mx/takeyas/libroED/Prog7-2.rar`

7.6.2.2 Parámetros de salida en C#

Un parámetro de salida de C# (*out*) es muy similar a un parámetro por referencia (*ref*), excepto que los parámetros por referencia deben ser inicializados antes de enviarse, sin embargo, el método debe asignarle un valor antes de devolverlo. Para utilizar un parámetro de salida, basta con anteponer la palabra *out* tanto en el parámetro enviado como en su declaración en el método.

Este tipo de parámetros son útiles cuando se requiere que una función devuelva más de un valor, ya que por definición, existe una restricción de que una función solamente devuelve un valor.

7.6.2.2.1 Prog. 7.3.- Proyecto de consola en C#: Parámetros de salida

Para ilustrar mejor el uso de un parámetro de salida, tomaremos como ejemplo el programa anterior (Prog. 7.2) al que se le agrega una variable booleana como parámetro de salida para determinar si el parámetro enviado por referencia es par ó impar. Este pseudocódigo muestra este ejemplo:

```
 x = 5  // Declaración de variable global

 PROGRAMA_PRINCIPAL
 1.- y = 13  // Declaración de variable local
 2.- IMPRIMIR x  // Despliega 5
 3.-  METODO(referencia y, salida esImpar) // Llamada del
método y le envía la variable “y” por referencia y el
parámetro de salida “esImpar”
 4.- IMPRIMIR x  // Despliega 10
 5.- IMPRIMIR y  // Despliega 16
```

```
6.- SI esImpar == verdadero ENTONCES
      6.1.1. IMPRIMIR “y es un número impar”
    SINO
      6.2.1. IMPRIMIR “y es un número par”
7.- {FIN DE LA CONDICIONAL DEL PASO 6}
8.- FIN


// Declaración del método con el parámetro local “a” por
referencia y el parámetro de salida “Impar”
METODO(referencia a, salida Impar)
9.- a = a + 3
10.- IMPRIMIR a  // Despliega 16
11.- x = x * 2
12.- SI a MOD 2 ≠ 0 ENTONCES
       12.1.1. Impar = verdadero
      SINO
       12.2.1. Impar = falso
13.- {FIN DE LA CONDICIONAL DEL PASO 12}
14.- RETURN
```

El Prog. 7.3 contiene el ejemplo completo en modo consola de la aplicación de un parámetro de salida en C#

Prog. 7.3. Parámetros de salida (Program.cs).

```
class Program
{
    static int x = 5;  // Variable global

    static void Main(string[] args)
    {
        int y = 13;  // Variable local
        bool esImpar;

        Console.WriteLine("\nx=" + x.ToString());

        Metodo(ref y, out esImpar);  // Llamada al método y envío por referencia

        Console.WriteLine("\nx=" + x.ToString());

        Console.WriteLine("\ny=" + y.ToString());

        if (esImpar)
```

```
                Console.WriteLine("\ny es un número impar");
            else
                Console.WriteLine("\ny es un número par");

            Console.ReadKey();
        }

        // El parámetro "a" recibe la referencia de "y" y el parámetro de salida
        // para determinar si el parámetro enviado es Impar
        static void Metodo(ref int a, out bool Impar)
        {
            a = a + 3;
            Console.WriteLine("\na=" + a.ToString());

            x = x * 2;

            if (a % 2 != 0)
                Impar = true;
            else
                Impar = false;
        }
    }
```

Al ejecutar el programa anterior se produce la salida mostrada en la Fig. 7.5.

```
x=5
a=16
x=10
y=16
y es un número par
```

Fig. 7.5. Salida del programa de uso de parámetro de salida.

El programa completo puede descargarse del siguiente sitio web:

`http://www.itnuevolaredo.edu.mx/takeyas/libroED/Prog7-3.rar`