UNIDAD N° 2

EJERCICIO N° 8

ESCRIBIR UNA SENTENCIA DO

1. Descargar el ejemplo **DoStatement** de: https://frtutneduar.sharepoint.com/:u:/r/sites/PAV2020/Documentos%20compartidos/Unidad%202/Ejercicios/Codigo/DoStatement.zip?csf=1&web=1&e=puuTFp
2. Iniciar Visual Studio 2017
3. Ir al menú Archivo → Abrir → Proyecto / Solución
4. Ubicar la carpeta donde se descargó el ejemplo
5. Seleccionar la solución DoStatement y elegir Abrir.
6. Visualizar el contenido del formulario Principal.cs haciendo doble-click sobre éste en el Explorador de Soluciones.

   El formulario contiene un cuadro de texto llamado *txtNumero* en el cual el se ingresa un número entero. Al presionar el botón Mostrar Pasos se genera la representación octal del número ingresado. El cuadro de texto *txtPasos* muestra cada etapa del cálculo.

7. Visualizar el código del formulario Principal.cs y ubicar el método *showStepsClick.* Este método se ejecuta al presionar el botón Mostrar Pasos.
8. Agregar las siguientes sentencias al método:

```
int amount = int.Parse(txtNumero.Text);
txtPasos.Text = "";
string current = "";
```

   La primera sentencia convierte el texto ingresado en el cuadro de texto *txtNumero* en un entero y lo almacena en la variable *amount*.
   La segunda sentencia borra el contenido del cuadro de texto *txtPasos* asignando una cadena vacía a la propiedad *Text*.

La tercera línea declara una variable string llamada curren y se inicializa con una cadena vacía. Ésta se utilizará para almacenar los dígitos generados en cada iteración para convertir el número en su representación octal.

9. Agregar las siguientes sentencias a continuación de las anteriores:

```
do
{
      int nextDigit = amount % 8;
      amount /= 8;
      int digitCode = '0' + nextDigit;
      char digit = Convert.ToChar(digitCode);
      current = digit + current;
      txtPasos.Text += current + "\n";
}
while (amount != 0);
```

El algoritmo utilizado realiza repetidamente aritmética de enteros para dividir la cantidad variable entre 8 y determinar el resto. El resto después de cada división sucesiva constituye el siguiente dígito en la cadena que se está construyendo. Finalmente, cuando la cantidad se reduce a 0, el ciclo termina. Tener en cuenta que el cuerpo debe correr al menos una vez. Este comportamiento es exactamente lo que se requiere porque incluso el número 0 tiene un dígito octal.
Mirar más de cerca el código, ver que la primera instrucción ejecutada por el bucle *do* es:

```
int nextDigit = cantidad % 8;
```

Esta declaración declara una variable int llamada *nextDigit* y la inicializa en el resto después de dividir el valor en la cantidad por 8. Este será un número en algún lugar entre 0 y 7.

La siguiente declaración en el bucle do es:

```
amount /= 8;
```

Esta es una declaración de asignación compuesta y es equivalente a escribir *amount = amount / 8*;. Si el valor de la cantidad es 999, el valor de la cantidad después de que se ejecuta esta declaración es 124.

La siguiente declaración es:

```
int digitCode = '0' + nextDigit;
```

Esta declaración requiere una pequeña explicación. Los caracteres tienen un código único de acuerdo con el conjunto de caracteres utilizado por el sistema operativo. En los juegos de caracteres utilizados con frecuencia por el sistema operativo Microsoft Windows, el código para el carácter "0" tiene un valor entero 48. El código para el carácter "1" es 49, el código para el carácter "2" es 50, y así sucesivamente al código para el carácter "9", tiene el valor entero 57. Con C #, puede tratar un carácter como un entero y realizar una aritmética en él, pero cuando lo hace, C# usa el código del carácter como valor. Entonces, la expresión '0' + *nextDigit* en realidad da como resultado un valor en algún lugar entre 48 y 55 (recordar que *nextDigit* estará entre 0 y 7), correspondiente al código para el dígito octal equivalente.

La cuarta declaración en el bucle do es:

```
char digit = Convert.ToChar (digitCode);
```

Esta declaración declara una variable *char* llamada *digit* y la inicializa con el resultado de llamar al método *ToChar(digitCode)* de la clase *Convert*. Este método toma un entero, que corresponde al código del carácter, y retorna el carácter. Por ejemplo, si *digitCode* tiene el valor 54, el método retorna el carácter “6”.

Para resumir, las primeras cuatro declaraciones en el bucle *do* determinan el carácter que representa el dígito octal menos significativo (más a la derecha) correspondiente al número que escribió el usuario. La siguiente tarea es anteponer este dígito a la cadena que se generará, de esta forma:

```
current = digit + current;
```

La siguiente declaración en el bucle do es:

```
txtPasos.Text += current + "\n";
```

Esta declaración agrega al cuadro de texto *txtPasos* la cadena que contiene los dígitos producidos hasta ahora para la representación octal del número. También agrega un carácter de nueva línea para que cada etapa de la conversión aparezca en una línea separada en el cuadro de texto.

Finalmente, se evalúa la condición en la cláusula *while* al final del ciclo:

```
while (amount != 0);
```

Como el valor de la cantidad aún no es 0, el ciclo realiza otra iteración.

10. **Opcional**: Se puede seguir el ejercicio de la página 127 del libro para visualizar los valores parciales utilizando el depurador de Visual Studio.