Tema: FUNDAMENTOS DE PROGRAMACIÓN EN LENGUAJE C#

Objetivo:

- ✓ Conocer y manejar correctamente los tipos de datos y las diferentes estructuras de programación que hacen parte del lenguaje de Programación "C#" en un proyecto tipo consola.

INTRODUCCIÓN

Una aplicación de consola es aquella que se ejecuta en una ventana que solo permite ingresar línea de comandos, es decir, no tiene elementos gráficos tales como botones o cajas de texto con que el usuario interactúe. Lo más común dentro del desarrollo bajo la plataforma .Net es la creación de aplicaciones tipo Windows o aplicaciones Web, sin embargo la mejor forma de sentar unas bases firmes acerca de la programación en este lenguaje es comenzar construyendo aplicaciones sencillas de tipo consola.

I. TIPOS DE DATOS, DEFINICIÓN DE VARIABLES Y CONSTANTES EN C#

Recibir datos de entrada, realizar un proceso u obtener salidas en un programa escrito en lenguaje C# requiere necesariamente del uso de variables, contantes y operadores.

La variables son datos cuyo valor puede cambiar en cualquier momento durante la ejecución de un programa; en términos computacionales una variable es un espacio de memoria que permite almacenar temporalmente un dato. En el lenguaje C# para poder usar una variable se requiere definirla por medio de un nombre y un tipo de dato de la siguiente forma.

`tipodedato` NombreDeVariable ;

El tipo de dato especifica la naturaleza del valor que se va a almacenar, por ejemplo si es un numero entero, decimal, pequeño, grande o si es una cadena de caracteres (Texto). Los Tipos de datos comúnmente utilizados en leguaje c# son:

Tipo De Dato	Clase .NET	Descripción	Tamaño (En Bytes)	Intervalo o Rango De Valores
`byte`	Byte	Entero sin signo	8	0 a 255
`sbyte`	SByte	Entero con signo	8	-128 a 127
`int`	Int32	Entero con signo	32	-2.147.483.648 a 2.147.483.647
`uint`	UInt32	Entero sin signo	32	0 a 4294967295
`short`	Int16	Entero con signo	16	-32.768 a 32.767
`ushort`	UInt16	Entero sin signo	16	0 a 65535

long	Int64	Entero con signo	64	-922337203685477508 a 922337203685477507
ulong	UInt64	Entero sin signo	64	0 a 18446744073709551615
float	Single	Tipo de punto flotante de precisión simple	32	-3,402823e38 a 3,402823e38
double	Double	Tipo de punto flotante de precisión doble	64	-1,79769313486232e308 a 1,79769313486232e308
char	Char	Un carácter	16	Símbolos o caracteres Unicode utilizados en el texto
bool	Boolean	Tipo Boolean lógico	8	True o false
string	String	Una secuencia de caracteres	En función de los caracteres	Secuencia de Símbolos o caracteres Unicode utilizados en el texto
decimal	Decimal	Tipo preciso fraccionario o integral, puede representar números decimales con 29 dígitos significativos	128	±1.0 × 10 e−28 a ±7.9 × 10 e28

Las constantes son valores que se conocen y no cambian durante la ejecución de un programa. Las constantes se declaran de forma similar a una variable utilizando como la palabra clave "`const`" antes del tipo de dato. Las constantes se deben inicializar en el momento que se declaran. Por ejemplo:

```
const int meses = 12;
const int dias = 365;
const double pi = 3,1416;
```

II. OPERADORES EN LENGUAJE C#

1. Operadores Aritméticos:

Operador	significado
(+)	Suma
(-)	resta
(*)	Multiplicación
(/)	División
(%)	Modulo

2. Operadores De Asignación

Operador	significado
(=)	Asignación simple
(+=)	Asignación suma
(-=)	Asignación Resta
(*=)	Asignación Multiplicación
(/=)	Asignación División

3. Operadores Relacionales o De Comparación

Operador	significado
(==)	Igual que
(!=)	Diferente De
(>)	Mayor que
(>=)	Mayor o Igual que
(<)	Menor que
(<=)	Menor o Igual que
(++)	Incremento
(--)	Decremento

4. Operadores Lógicos

Operador	significado
(&&)	AND , Operador "Y" Lógico
(||)	OR , Operador "O" Lógico
(!)	Not , Operador "Negación"

5. Operador De Concatenación: (+)

Es la operación por la cual dos caracteres se unen para formar una cadena de caracteres (o *string*). También se puede concatenar dos cadenas de caracteres o un carácter con una cadena para formar una cadena de mayor tamaño. Algunos ejemplos serían:

- ✓ 'a' *concatenado 'b'* → *"ab"*
- ✓ "ABCD" *concatenado 'b'* → *"ABCDb"*
- ✓ 'a ' *concatenado "XYZ"* → *"a XYZ"*
- ✓ "ABCD " *concatenado "XYZ"* → *"ABCD XYZ"*

Para realizar una concatenación en Lenguaje C# se usan valores o variable tipo char o string y se unen con el símbolo (+)

Ejemplos:

- ✓ "El valor de a es" + a. ToString ();
- ✓ "Su edad es" + edad. ToString ();

6. Operadores de Conversión:

a) Para convertir el tipo de dato en otro, su precede el tipo que queremos cambiar con el tipo al que queremos convertir, entre paréntesis, de esta forma: variable1 = (tipo) variable2;

Ejemplos:
- ✓ int a = (int) b;
- ✓ d = (double) x;

b) En el lenguaje C# los tipos de datos numéricos cuentan con una función que permite convertir cadenas de texto a un tipo numérico especifico. Esta función es muy útil para capturar los datos de entrada numéricos, ya que estos en la mayoría de los casos ingresan al programa como cadenas de texto (string) y para poder hacer operaciones matemáticas con estos es necesaria su conversión. Esta función es conocida como “Parse” y aplica para todos los tipos de datos numéricos, algunos ejemplo de su uso:

Ejemplo1:
```
Console.WriteLine("Digite Numero 1:");
int n1 = int.Parse(Console.ReadLine());
```

Ejemplo2:
```
Console.WriteLine("Digite nota:");
double nota = double.Parse(Console.ReadLine());
```

c) El Lenguaje C# también cuenta con un conjunto de funciones que realizan conversiones entre los diferentes tipos de datos. Estas funciones se encuentran en la “Librería” Convert de C# y se utiliza como se ve en el siguiente ejemplo:

```
double dNumber = 23.15;
int    iNumber = Convert.ToInt32(dNumber);
```

7. Operador Condicional:

Operador Condicional: Es el único operador de C# que tiene tres operandos. Su sintaxis es esta:

<condición> ? <expresión1> : <expresión2>;

Quiere decir que si la condición es true, se evalúa expresión1, y si es falsa, se evalúa expresión2. No se debe confundir el operador condicional con un "if". Este operador devuelve un valor, mientras que el if es una instrucción usada para la toma de decisiones

Ejemplo:

```
b = (a>0)? a : 0;      // a y b de tipos enteros
```

En este ejemplo, si el valor de la variable a es superior a 0 se asignará a b el valor de a, mientras que en caso contrario el valor que se le asignará será 0. Los caracteres “// “ sirven para realizar comentarios (Texto que no se ejecuta) a las líneas de código en C#.

III. APLICACIONES TIPO CONSOLA CON LENGUAJE C#

Se puede definir una aplicación de consola como aquella que se ejecuta en una ventana tipo MS-DOS, es decir, una ventana en donde el usuario interactúa únicamente con líneas de comandos de texto. Antes de comenzar a desarrollar la aplicación se ha de conocer la clase principal que interactúa con la consola de líneas de comandos es la clase llamada clase "Console".

Mediante esta clase se consigue mostrar información en la pantalla así como capturar la información que introduzca el usuario, cabe destacar que los métodos de la clase Console son de tipo Shared, por lo que no es necesario crear un objeto a partir de la clase para invocar a sus métodos, es posible hacerlo indicando el nombre de la clase seguido de un punto y el nombre del método. Los dos principales métodos de la clase Console son:

1. El método WriteLine()

Este método es el que se usa para mostrar texto en la consola, el método escribe en la pantalla el valor que le pasemos como parámetro. El parámetro que recibe el método puede ser de varios tipos, ya sea una cadena de caracteres, un número entero, una línea en blanco, etc..

2. El método ReadLine()

Este método se usa para recoger la información que el usuario introduce cuando la aplicación así lo requiera. Cuando invocamos al método Console.ReadLine() el sistema queda en espera hasta que el usuario pulsa la tecla Intro. Si se asigna la llamada a Console.ReadLine() a una variable se consigue capturar el dato introducido por el usuario, para después poder operar con él.

Ejercicio 1:

1. Cree un nuevo proyecto en .Net (Archivo – Nuevo Proyecto) seleccione el lenguaje C# y en plantillas seleccione Windows - aplicación de consola.

2. En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        int n1;
        int n2;
        int Suma=0, Resta=0, Multiplicacion=0;

        Console.WriteLine("Digite Numero 1:");
        n1 = int.Parse(Console.ReadLine());
        Console.WriteLine("Digite Numero 2:");
        n2 = int.Parse(Console.ReadLine());

        Suma = n1 + n2;
        Console.WriteLine("La suma de los numeros es: ");
        Console.WriteLine(Suma.ToString());
```

```
        Resta = n1 - n2;

        Console.WriteLine("La resta de los numeros es :" +
        Resta.ToString());

        Multiplicacion = n1 * n2;
        Console.WriteLine("La multiplicacion de los numeros es:" +
        Multiplicacion.ToString());

        Console.ReadLine();
        }
    }
}
```

IV. ESTRUCTURAS CONDICIONALES EN LENGUAJE C#

2.1 Condicional if: Cuando se cumple una determinada condición, se ejecuta una o un conjunto de acciones específicas. Su estructura es:

```
if (condición Logica)

{
   Instrucción 1;
   Instrucción 2;
   …
   Instrucción n;

}
```

✓ **Ejercicio 2.1: (Condicional if)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        int n, r;
        Console.WriteLine("Digite un Numero");
        n = int.Parse(Console.ReadLine());
        r = n % 2;

        if (r == 0)
        {
            Console.WriteLine("El numero Ingresado es par");
        }
        Console.ReadLine();
        }
    }
}
```

2.2 Condicionales (if – else) - (if – elseIf – else): se utilizan para evaluar diferentes opciones que puede tomar una condición. Su estructura es

If – else	If – else if - else
if(condición) **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}**	**if (condición 1)** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else if (condición 2)** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}**

✓ **Ejercicio 3.2. Condicional (if – else)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        int n, r;
        Console.WriteLine("Digite un Numero");
        n = int.Parse(Console.ReadLine());
        r = n % 2;
        if (r != 0)
          {
            Console.WriteLine("El numero Ingresado Es IMPAR");
          }
        else
          {
            Console.WriteLine("El numero Ingresado es PAR");
          }
        Console.ReadLine();
        }
    }
}
```

3.3 Selección de Casos switch: se utiliza cuando se requiere en misma estructura evaluar varios valores que puede tomar una variable. Su estructura es:

```
switch (variable_a_evaluar)
    {
        case valor_1:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;

          }
        case valor_2:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
          }

         case valor_n:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
          }

           default:
            {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
            }
    }
```

✓ **Ejercicio 3.3. (switch)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
    int seleccion = 0;

    Console.WriteLine("Menu:");
    Console.WriteLine("1. Valor Absoluto");
```

```
        Console.WriteLine("2. Raiz Cuadrada");
        Console.WriteLine("3. Seno");
        Console.WriteLine("Por favor Digite una Opción c#");
        seleccion = int.Parse(Console.ReadLine());

        switch (seleccion)
         {
           case 1:
              {
              double Num1 = 0;
              Console.WriteLine("Digite un numero positivo o negativo");
              Num1 = int.Parse(Console.ReadLine());
              Potencia = Math.Abs(Num1);
              Console.WriteLine("El valor absoluto del numero " + Num1 + " es " + Potencia);
              break;
              }
           case 2:
              {
              double Num = 0;
              double Raiz = 0;
              Console.WriteLine("Digite el numero");
              Num = int.Parse(Console.ReadLine());
              Raiz = Math.Sqrt(Num);
              Console.WriteLine("La raiz Cuadrada de " + Num + " es " + Raiz);
              break;
              }
           case 3:
              {
              double Num = 0;
              double Seno = 0;
              Console.WriteLine("Digite el numero");
              Num =int.Parse(Console.ReadLine());
              Seno = Math.Sin(Num);
              Console.WriteLine("El Seno de " + Num + " en radianes   es " + Seno);
              Seno = Math.Sin(Num * (Math.PI / 180));
              Console.WriteLine("El Seno de " + Num + " en grados es " + Seno);
              break;
              }
           default:
              {
              Console.WriteLine("Usted Digitó un opcion InValida");
              break;
              }
         }
        Console.ReadLine();

            }
         }
}
```

V. ESTRUCTURAS REPETITIVAS EN LENGUAJE C#

Se usan cuando se requiere que un procedimiento o un conjunto de instrucciones se ejecuten y se repitan cierta cantidad veces y finalicen únicamente cuando se cumpla una determinada condición.

4.1 La instrucción for: Su estructura es:

```
for (variable_inicio; condición_de_fin; incremento)
{
        Instrucción1;
        Instrucción2;
        …
        Instrucción n;
}
```

4.2 La instrucción while: su estructura es

```
while (condición lógica)
    {
        Instrucción1;
        Instrucción2;
        Incremento;  // opcional, tener muy en cuenta para evitar ciclos infinitos
     }
```

4.3 La instrucción do – while: su estructura es:

```
do
    {
        Instrucción1;
        Instrucción2;
        Incremento;  // opcional, tener muy en cuenta para evitar ciclos infinitos
    }
while (condición logica);
```

✓ **Ejercicio 4.1 (Ciclo Repetitivo for)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string salida = "";
            Console.WriteLine("Los números Impares hasta 50 son: ");

            for (int cont = 1; cont <= 50; cont = cont + 2) {
            salida = salida + " " + cont;
            Console.WriteLine(salida);
            }

             salida = "";
```

```
                Console.WriteLine(" ");
                Console.WriteLine("Los números Pares hasta 50 son: ");


                for (int cont = 2; cont <= 50; cont += 2) {
                  salida = salida + " " + cont;
                 }
                Console.WriteLine(salida.ToString());
                Console.ReadLine();

        }
    }
}
```

✓ **Ejercicio 4.2 (Ciclo Repetitivo while)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
    class Program
    {
        static void Main(string[] args)
        {
          int cantidad = 0, numero = 0, prom = 0;
          Console.WriteLine("Cuantos Numeros Desea Ingresar");
          cantidad = int.Parse(Console.ReadLine());
          int cont = 1;
          int suma = 0;

          while ((cont <= cantidad))
          {
             Console.WriteLine("Digite el numero " + cont);
             numero =  int.Parse(Console.ReadLine());
             suma = suma + numero;
             cont = cont + 1;
          }

          prom = suma / cantidad;

          Console.WriteLine("El suma de todos los numeros Ingresados es: " + suma.ToString());
          Console.WriteLine("El Promedio de los numeros Ingresados es: " + prom.ToString());
          Console.ReadLine();

        }
    }
}
```

✓ **Ejercicio 4.3 (Ciclo Repetitivo do - while)**

En el editor de código escriba las siguientes líneas escritas en lenguaje C#.

```
class Program
{
    static void Main(string[] args)
    {
int seleccion = 0;

do
{
  Console.WriteLine("*****************************");
  Console.WriteLine("Menu: ");
  Console.WriteLine("1. Factorial");
  Console.WriteLine("2. Fibonacci");
  Console.WriteLine("3. Salir");
  Console.WriteLine("*****************************");
  Console.WriteLine("Por favor Digite una Opción");
  seleccion = int.Parse(Console.ReadLine());


  switch (seleccion) {
     case 1:
      {

        int Num = 0;
        int Fact = 0;

        Console.WriteLine("Por favor Digite un numero entero: ");
        Num = int.Parse(Console.ReadLine());
        Fact = 1;

        if (Num > 1) {
           for (int i = 1; i <= Num; i ++) {
              Fact = Fact * i;
           }
        }

        Console.WriteLine("El Factorial de " + Num + " es " + Fact);
        Console.WriteLine(" ");

        break;
      }
     case 2:
      {

        int elementos = 0;
        int a = 0;
        int b = 0;
        int c = 0;
        string serie = null;

        Console.WriteLine("Digite el número de elementos a mostrar     de la serie de fibonaci:");
        elementos = int.Parse(Console.ReadLine());

        a = 0;
```

```
                b = 1;
                serie = a + " - " + b;

                for (int i = 0; i <= elementos; i++) {
                    c = a + b;
                    a = b;
                    b = c;
                    serie = serie + " - " + c;
                }


                Console.WriteLine(serie);
                Console.WriteLine(" ");

                break;
              }
            case 3:

              {
                Console.WriteLine("Muchas Gracias Por usar Este Software");
                Console.WriteLine(" ");
                break;
               }

            default:
                Console.WriteLine("Favor Seleccione Una Opción Correcta");
                Console.WriteLine(" ");
                break;
        }

    } while (seleccion != 3);

    Console.ReadLine();

        }
     }
}
```