DIPLOMADO: Fundamentos de Programación Web
MODULO: Lógica de programación
INSTRUCTOR: Tec. Darwin Flores

Resultados de aprendizaje:

Que el estudiante:

- Se familiarice con el entorno de desarrollo Visual Estudio
- Determine las reglas de sintaxis de las instrucciones usadas.
- Desarrolle un diagrama de flujo
- Programe problemas de algoritmos de manera secuencial en lenguaje de programación c#.
- Identifique los diferentes operadores matemáticos usados para definir expresiones
- Aplique la jerarquía de operadores de una manera correcta.
- Utilizar las instrucciones de selección “if” e “if..else” para elegir una de varias acciones alternativas.
- Conocer la sintaxis de C# para las instrucciones condicionales simples, dobles y múltiples, así como también la utilidad en la programación.
- Aprender a utilizar la estructura y sintaxis del switch-case para la evaluación de condiciones múltiples.

Contenido

VISUAL C#

Visual Studio .NET es un entorno de programación repleto de herramientas que contiene toda la funcionalidad necesaria para la creación de proyectos de C# grandes o pequeños. Es posible crear, incluso proyectos que combinan de forma homogénea módulos de lenguajes diferentes.

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos; no obstante, perfectamente pueden crearse aplicaciones utilizando el paradigma de la programación estructurada. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C.

QUE SON LAS VARIABLES Y CONSTANTES EN C#

Si estas iniciando en el mundo de la programación de seguro que el termino variable y constante fue de los primeros que viste y si te quedaste con alguna duda no te preocupes porque en este documento te dejare en claro que son las variables y constantes en C#.

Variable

Es un espacio reservado en la memoria de la computadora, este espacio este asociado a un nombre simbólico también conocido como identificador.

En este espacio podemos almacenar un valor, el cual puede ser modificado en cualquier momento durante la ejecución del programa que contiene la variable.

Para crear o declarar una variable en lenguajes como C, Java, C++ y C#, se realiza de la siguiente forma:

1. Se coloca el tipo de datos que esta variable almacenara.
2. Se coloca un identificador, esto sería el nombre de la variable
3. y por ultimo y opcional se le asigna un valor.

Un ejemplo concreto sería el siguiente y aplica para lenguajes de programación ya mencionados

```
int a = 56; // int es el tipo de datos, a es el nombre o identificador
char caracter = '@'; // char es el tipo de datos, caracter es el identificador
string myNombre = "Juancito";
```

Los Identificadores o nombres de variables

Los identificadores deben ser nombres que hagan referencia al uso que se le dará a la variable, por ejemplo, si la variable será usada para el apellido, lo recomendado es ponerle «apellido», y evitar usar identificadores como «x», «c», «a», porque en este caso no identifican correctamente el valor de esta variable.

El identificador de **una variable debe ir en minúscula**, (esto es un estándar, no obligatorio, pero es buena práctica)

También puede iniciar con una letra o underscore (_), nunca números o caracteres especiales como la eñe, letras con acentos entre otros.

Ejemplo de creación de variable

```
int a = 56;
int _a = 45;
int 45a = 90; // el compilador no lo permite
int 45; // no es permitido
int $%^& = 5; //no es permitido
```

No se permite caracteres especiales fuera del inglés (acentos españoles, franceses, etc.)

```
string contraseña =  "12345";// Erroneo
string contrasena = "12345"; //Correcto
```

Si el identificador está compuesto por dos palabras o mas usar el método **camelcase** , **pascalcase** o «_»

```
String primeraPalabra ; //Camel Case (jorobas de camello)
String primera_palabra; //uso de underscore(_)
int estaVariableTieneMuchasPalabras; // mas de 2 palabras
String primera pablabra; // no se permiten espacios
```

Constante

Es similar a la variable, tiene un tipo de datos, un identificador, pero a diferencia de la variable, una vez asignado un valor este no cambiara nunca durante la ejecución del programa, y se debe asignar un valor en su declaración.

Para su declaración se hace uso de la palabra reservada const. la cual le dice al compilador que esto sera una constante.

```
const double PI = 3.14;
const char SUBFIJO = '_'
```

También las constantes deben escribirse en mayúsculas (no obligatorio, pero permite establece diferencia entre variables y constante en otras partes del código).

QUE SON LOS OPERADORES EN C#

Todos los lenguajes de programación poseen mecanismos para realizar operaciones sobre los datos y c# no es la excepción, por eso en esta entrada te explicare que son los operadores en c#.

*En C#, un operador es un elemento de programa que se aplica a uno o varios operandos en una expresión o instrucción. Los operadores que toman un operando, como el operador de incremento ++ o new, se conocen como operadores unarios. Los operadores que toman dos operandos, como los operadores aritméticos (+, -, *, /) se conocen como operadores binarios. Un operador, el operador condicional (? :), toma tres operandos y es el único operador ternario de C#.*

Tenemos 2 palabras destacadas **operadores** y **operandos**

los **operadores** son los elementos que usaremos para realizar ciertas operaciones (comparación, sumas, resta, incrementos, etc.)

los **operandos** son los elementos sobres los cuales son aplicados los operadores (números, variables, etc.)

En este Post trataremos los siguientes tipos de operadores en c#

- Operadores aritméticos
- Operadores de asignación
- Operadores lógicos
- Operadores relacionales u operadores de comparación
- Operadores de incremento y decremento

Operadores Aritméticos

Son símbolos aritméticos básicos: suma (+), resta (-), multiplicación (*), división (/) y Modular (%) este último es el residuo de una división entera.

Estos toman valores numéricos sean variables o valores literales como su operando y dan como resultado un solo valor numérico, este resultado lo puedo almacenar en una variable si deseo utilizarlo más adelante.

```
// esto es una suma da como resultado 13
int suma = 5 + 8; // se realiza la suma y se almacena en la variable
double division = 8/9;
long multiplicacion = 45 * 45; // long es tipo entero(int) pero soporta
//cantidades mas grandes
int resta = 5-1;
```

el operador Modular (%) retorna el residuo de una división entera, es decir cuando dividimos dos números y hay una parte que no puede ser dividida por lo tanto esta parte sobra, en una división entera el resultado no tiene decimales.

```
int resultado;

int numero1 = 7;
int numero2 = 5;

resultado = numero1 % numero2; // el resultado sera 2,
//porque 7 entre 5 es igual a 1, y sobran 2

numero1 = 18;
numero2 = 3;

resultado = numero1 % numero2; // el resultado sera 3,
//porque 17 entre 3 es igual a 5, y sobran 3
```

el resultado de modular siempre sera la parte que no se puede dividir, sería el sobrante o residuo de la división.

Operadores de Asignación

Los operadores de asignación se utilizan para asignar un nuevo valor a una variable, propiedad o evento. El **operando** izquierdo de una **asignación** debe ser una **variable**, un acceso a propiedad, un acceso a indizador o un acceso a evento.

Este asigna el valor del operando de la derecha al operando de la izquierda.

```
int numero;
string nombre;

numero = 45;
// se le asigna el valor 45(derecha) a la variable numero(izquierda)
nombre = "Jorge";
// se le asigna jorge a la variable nombre
numero = numero + 13;
// se le suma 13 al valor contenido en numero (45)
//y luego se le asigna el resultado a numero
```

Existen sobrecargas del operador de asignación (=), estas acortan ciertas operaciones

```
int numero1 = 4;
int numero2 = 5;

numero1 += 5; // esto equivale a numero1 = numero1 + 5
numero1 -= numero2; // esto equivale a numero1 = numero1 - numero2
numero1 *= 2; //esto equivale a numero1 = numero1 - numero2
numero2 /= numero1; //esto equivale a numero1 = numero1  / numero2
numero1 %= 2; //esto equivale a numero1 = numero1 % numero2
```

Operadores relacionales

son símbolos usado para comparar dos valores, si la expresión evaluada es correcta, entonces estos operadores dan como resultado verdadero, en caso contrario falso

Operador	Significado	Ejemplo
==	Igual que	a == b
!=	Distinto que	a != b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor o igual que	a <= b
>=	Mayor o igual que	a >= b

```
bool resultado;

resultado = 5 > 1; // esto da como resultado verdadero (true)
resultado = 5 <= 5; // esto da como resultado verdadero(true)
//debido a que aunque 5 no es menor que 5, pero si es igual a 5

String valor1 = "juan";
String valor2 = "pepe";

resultado = valor1 != valor2; // esto es verdadero,
porque juan es diferente a pepe
resultado = "pepe" == valor2; // esto es como resultado verdadero,
//ambos valores son iguales

resultado = valor1 == "Juan"; // esto es falso (false), debido a que juan y Juan
//son valores diferentes, C# distingue entre minusculas y mayusculas
```

Operadores lógicos

Estos trabajan usando la tabla de verdad,

El Operador &&(AND) que quiere decir «y» expresa que si ambas expresiones son verdaderas entonces el retorna el valor verdadero.

El operador lógico && (AND)

P	Q	P && Q
V	V	V
V	F	F
F	V	F
F	F	F

El operador || (OR) que quiere decir «o» expresa que al menos una de las dos expresiones debe ser verdadera y retornara verdadero

El operador lógico || (OR)

P	Q	P || Q
V	V	V
V	F	V
F	V	V
F	F	F

El operador! (NOT) quiere decir «no» o «negación», cambia el valor al su inverso, es decir si es verdadero, el resultado cambiara a falso, y viceversa.

El operador lógico ! (NOT)

P	!P
V	F
F	V

Ahora bien, ¿cómo utilizamos esto en programación?, la respuesta es simple, podemos combinar los operadores relacionales con los lógicos.

```
int n1 = 78;
int n2 = 90;
int n3 = 9;

bool resultado;

resultado = n1 <  n2 && n2 > n3;
//en este caso el resultado es verdadero
//debido a que n1 < n2 es verdadero
//y n2> n3 tambien es verdadero
//aplicamos el operador &&  ( leer operador &&)

resultado = n1 != 70 || n3> 10;
//en este caso el resultado es verdadero
//debido a que n1 != 70 es verdadero
//n3 > 10 es falso , pero el operador OR solo necesita uno verdadero
//aplicamos el operador || ( leer operador ||)

resultado = true;
resultado = !resultado;
//en este caso el resultado es falso
//debido a que la variable resultado contiene el valor true(verdadero)
//pero aplicando el operador  de negacion NOT(!), hemos invertido el valor

//en este caso usamos el if para aprovechar los operadores
if(n1>10){
 Console.WriteLine("Es Mayor");//por consola
MessageBox.Show("Es mayor");//Por ventana
}
```

Tanto los operadores relacionales y los lógicos son utilizados por las estructuras de control de flujo (if, while, for).

Operador de incremento

Estos operadores incrementan y reducen el valor de una variable en 1.

el operador de incremento es «++», y se aplica a la variable que se quiere incrementar.

```
int contador = 0;

contador++; //aqui la variable incremento en 1 su valor y vale 1
contador++; //aqui la variable incremento en 1 su valor y vale 2
contador++; //aqui la variable incremento en 1 su valor y vale 3
contador++; //aqui la variable incremento en 1 su valor y vale 4
contador++; //aqui la variable incremento en 1 su valor y vale 5
contador++; //aqui la variable incremento en 1 su valor y vale 6
contador++; //aqui la variable incremento en 1 su valor y vale 7
contador++; //aqui la variable incremento en 1 su valor y vale 8
contador++; //aqui la variable incremento en 1 su valor y vale 9
contador++; //aqui la variable incremento en 1 su valor y vale 10
```

el operador decremento es «–» y se aplica a la variable que se quiere reducir.

```
int contador = 10;

contador--; //aqui la variable incremento en 1 su valor y vale 9
contador--; //aqui la variable incremento en 1 su valor y vale 8
contador--; //aqui la variable incremento en 1 su valor y vale 7
contador--; //aqui la variable incremento en 1 su valor y vale 6
contador--; //aqui la variable incremento en 1 su valor y vale 5
contador--; //aqui la variable incremento en 1 su valor y vale 4
contador--; //aqui la variable incremento en 1 su valor y vale 3
contador--; //aqui la variable incremento en 1 su valor y vale 2
contador--; //aqui la variable incremento en 1 su valor y vale 1
contador--; //aqui la variable incremento en 1 su valor y vale 0
```

COMO CONVERTIR TIPOS DE DATOS EN C#

En el mundo de la programación es esencial saber trabajar con los tipos de datos, y saber como convertir tipos de datos, ya que en la mayoría de los caso tendremos que convertir un tipo de dato a otro, ya sea para realizar operaciones aritméticas, concatenar cadenas, redondear números, etc.

En estos casos tendremos que hacer uso de una técnica llamada parseo (parsing en ingles), también conocido como casting, la cual nos permite convertir un tipo de dato en otro, siempre y cuando cumpla con ciertos requisitos.

Cada lenguaje de programación tiene sus métodos para realizar la conversión de tipo, en C#(C Sharp), tenemos la Clase **Convert.**

Esta clase contiene los métodos necesarios para convertir a casi todos los tipos de datos existentes en C#.

Las siguientes variables contienen números, pero no todas son tipo numérico, por lo tanto si quiero realizar una suma, solo las 3 primeras pueden ser utilizadas, las otras 3 deben ser convertidas a otro tipo de dato numérico, como pueden ser interger(int), double, o float.

```
int numero = 56; // variable tipo entero
double flotanteLargo = 56.77878; // variable tipo flotante largo
float flotante = 2.9f; // variable tipo flotante corto

char caracter = '9'; // Variable tipo carácter
string cadena = "56"; // Variable tipo cadena
object objeto = 45.25;
```

Para convertirlas usaremos la clase **convert** mencionada anteriormente.

```
int n_cadena;
n_cadena = Convert.ToInt32(cadena);

double n_caracter;
n_caracter = Convert.ToDouble(caracter);

double n_objeto;
n_objeto = Convert.ToDouble(objeto);
```

Las Conversiones pueden ser implícitas o explicitas.

Conversiones implícitas: no se requiere una sintaxis especial porque la conversión se realiza con seguridad de tipos y no se perderán datos.

por ejemplo, de un **entero** a un **flotante** o **double**

```
int numero = 90; // numero contiene el valor 90
double n_numero; //se declara una variable de tipo double
n_numero = numero;
//se le asigna el valor de numero a n_numero, realizamos la conversion de int a double

//numero tiene como valor 90
//n_numero tiene como valor 90.0
```

también en el caso de que el tipo de datos sea ***Object***, puede ser convertido implícitamente a otro tipo de la siguiente forma.

```
float flota = (float)objeto;
int n_entero = (int)objeto;
char n_char = (char)objeto;
string n_object = (string)objeto;
double n_doble = (double)objeto;
```

Conversiones explícitas (conversiones de tipos): las conversiones explícitas requieren un operador de conversión. se utilizan cuando es posible la perdida de información.

En este caso nos auxiliamos de **Clases** y **metodos**

Cada tipo de dato tiene una clase que posee un método llamado **parse** este convierte al tipo de dato desde un **string**

```
string cadena = "5";

float flota = float.Parse(cadena);
int n_entero = int.Parse(cadena);
char n_char = char.Parse(cadena);
double n_doble = double.Parse(cadena);
```

sí necesitas convertir cualquier tipo a String(cadena) solo tienes que llamar el metodo .ToString() de la siguiente forma.

```
float flota = 45.9f;
string n_string = flota.ToString();
```

RESOLUCIÓN DE PROEBLEMAS EN C#

Ejemplo1 Diseñar un programa en C# que nos permita realizar la suma de dos números.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  namespace Guia3Ejemplo1
6  {
7   class Suma_de_dos_numeros
8   {
9    static void Main(string[] args)
10   {
11    Console.Title = "Programa que suma 2 numeros";
12    // Declaracion de variables
13    Double n1, n2, resp;
14    Console.WriteLine("\nPrograma que suma dos números");
15    // Entrada de datos
16    Console.WriteLine("\nDigitar el primer número:");
17    n1 = Double.Parse(Console.ReadLine());
18    Console.WriteLine("\nDigitar el segundo número:");
19    n2 = Double.Parse(Console.ReadLine());
20    // Proceso de los datos
21    resp = n1 + n2;
22    // Salida de los datos
23    Console.WriteLine("\nLa suma de los numeros digitados es:" + resp);
24    Console.ReadKey();
25   }
26  }
27 }
```

Ejemplo2

Diseñar un programa en C# que nos permita encontrar el promedio de ventas de un vendedor. Es importante preguntar el nombre y apellido de la persona.

```
1    static void Main(string[] args)
2     {
3     Console.ForegroundColor = ConsoleColor.Black;
4     Console.BackgroundColor = ConsoleColor.White;
5     Console.Clear();
6     Console.Title = "Promedio de ventas";
7     String nombre,apellido;
8     Double v1,v2,v3,prom;
9     Console.WriteLine("Digitar nombre del vendedor");
10    nombre = (Console.ReadLine());
11    Console.WriteLine("Digitar apellido del vendedor");
12    apellido = (Console.ReadLine());
13    Console.Write("Digitar la primer venta: ");
14    v1 = Double.Parse(Console.ReadLine());
15    Console.Write("Digitar la segunda venta: ");
16    v2 = Double.Parse(Console.ReadLine());
17    Console.Write("Digitar la tercer venta: ");
18    v3 = Double.Parse(Console.ReadLine());
19    prom = (v1 + v2 + v3)/3;
20    Console.WriteLine("El promedio de " + nombre + " es: " + prom);
21    Console.ReadKey();
22    }
```

EJERCICIOS DE TABLAS DE VERDAD

Indicaciones: Resolver a mano las siguientes expresiones.

PARTE I. EJERCICIOS DE TABLAS DE VERDAD

Si A y B son enunciados verdaderos y X e Y son falsos, cuál es el valor de verdad de los siguientes enunciados:

1. ~(A v X)
2. ~ A v ~ X
3. ~ B ^ ~ Y
4. A v (X ^ Y)
5. (A ^ X) v (B ^Y)
6. A ^ [X v (B ^ y)]
7. A v [X ^ (B v Y)]
8. X v [A ^ (Y v B)]
9. [(A^X) v ~B] ^ ~ [(A^ X) v ~B]
10. [(X^A) v ~Y] v ~ [(X^A) v ~y]

PARTE II. EJERCICIOS SOBRE OPERADORES Y SU JERARQUÍA

Evalúe las siguientes expresiones y muestre el resultado.

1. ¿Cuál es el valor de la expresión 4 * 7 / 2 + 8 * 4 mod 3 - 5?
   a) -4
   b) 11
   c) 14

2. Not(S > 3 AND S<=10) OR (T >= 100 AND T< 200) , Para S = 5 y T = 70?
   a) Falso
   b) Verdadero

Represente las siguientes expresiones algebraicas en expresiones algorítmicas:

$V = \frac{1}{3} \pi r^2 h$

$X = 6(x + y)$

$$x = \frac{a + b + \frac{a}{b}}{c}$$

$$x = \frac{a + b + c}{a + \frac{b}{c}}$$

EJERCICIOS DE ESTRUCTURAS SECUENCIALES

Indicaciones: Resolver en lenguaje C# los siguientes problemas y crear su respectivo diagrama de flujo.

1. Escribir un programa que pregunte al usuario su nombre, apellido, DUI, edad, teléfono y dirección, y que lo muestre en pantalla.
2. Calcular el perímetro y área de un rectángulo dada su base y su altura.
3. Dados los catetos de un triángulo rectángulo, calcular su hipotenusa.
4. Dados dos números, mostrar la suma, resta, división y multiplicación de ambos.
5. Escribir un programa que convierta un valor dado en grados Fahrenheit a grados Celsius. Recordar que la fórmula para la conversión es: C = (F-32) *5/9
6. Calcular la media de tres números pedidos por teclado.
7. Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.
8. Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuanto deberá pagar finalmente por su compra.
9. Un alumno desea saber cuál será su calificación final en la materia de lógica de programación. Dicha calificación se compone de los siguientes porcentajes:

55% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

15% de la calificación de un trabajo final.

10. Se pide crear un programa que calcule el porcentaje de alumnos y alumnas de un salón de clase.
11. En un hospital existen tres áreas: Pediatría, Oncología y Traumatología. El presupuesto anual del hospital se reparte conforme a la siguiente tabla:

AREA	PORCENTAJE DEL PRESUPUESTO
Pediatría	30%
Oncología	40%
Traumatología	30%

ESTRUCTURAS DE SELECCIÓN EN C#

Por lo general, las instrucciones en una aplicación se ejecutan una después de la otra, en el orden en que se escriben. A este proceso se le conoce como ejecución secuencial. Varias instrucciones de C# le permiten especificar que la siguiente instrucción a ejecutar no es necesariamente la siguiente en la secuencia. A este se le conoce como transferencia de control.

Estructuras de Selección en C#

C# cuenta con tres tipos de estructuras de selección, que de aquí en adelante denominaremos instrucciones de selección. La instrucción if realiza (selecciona) una acción si una condición es verdadera, o ignora la acción si la condición es falsa.

A. La instrucción if…else realiza una acción si una condición es verdadera o realiza una acción distinta
B. si la condición es falsa.
C. La instrucción switch realiza una de varias acciones distintas, dependiendo del valor de una
D. expresión (expresión de control).
E. A la instrucción if se le llama instrucción de selección simple, debido a que selecciona o ignora
F. una acción individual.
G. A la instrucción if...else se le llama instrucción de selección doble, debido a que selecciona una
H. de dos acciones distintas (o grupos de acciones).
I. A la instrucción switch se le llama instrucción de selección múltiple, debido a que selecciona una
J. de varias acciones distintas (o grupo de acciones).

Sintaxis de las Instrucciones Condicionales

Una instrucción **if** simple responde a la siguiente sintaxis:

```
if (expresión booleana)
{
        Instrucción(es) de condición verdadera.
}
```

Advertencia: Las llaves son opcionales si hay una sola acción. Normalmente, es aconsejable añadirlas de todas formas. Su omisión puede ser causa de fallos inesperados.

La instrucción **if** solo garantiza la ejecución de determinadas acciones basándose en una condición verdadera (true). Se ejecutan o no se ejecutan. Para controlar tanto la condición **true** como la falsa (false) es necesario utilizar la instrucción **if...else**. Su sintaxis en la siguiente:

```
if (expresión booleana)
{
        Instrucción(es) de condición verdadera.
}
else
{
        Instrucción(es) de condición falsa.
}
```

Recordar:

La condición siempre se evalúa en el if. NUNCA se evaluará una condición después de la sentencia else.

Muchas veces vamos a necesitar evaluar más de dos condiciones, por lo que nos encontramos con el siguiente tipo de estructura, para el cual se muestra la sintaxis e interpretación:

```
if (condicion1)
{
        Bloque de instrucciones que se ejecutan si la condición 1 es verdadera.
}
else if(condicion2)
{
        Bloque de código que se ejecuta si la condición 2 es verdadera.
}
else if(condicion n)
{
        Bloque de código se la siguiente condición n es verdadera.
}
else
{
        Bloque de código que se ejecuta si ninguna de las condiciones previas fue verdadera.
}
```

Nota:

- La operación (x&&y) corresponde a la operación (x and y).
- La operación (x||y) corresponde a la operación (x or y).

Instrucción Switch

Cuando hay muchas condiciones a evaluar, la instrucción if…else puede resultar demasiado compleja de manejar. Una solución mucho más limpia en estos casos consiste en usar la instrucción switch. La instrucción switch permite cualquier valor entero o de cadena con múltiples valores. Cuando se produce una coincidencia se ejecutan todas las instrucciones asociadas con ella. Esta estructura es muy utilizada a la hora de trabajar con menú dentro de las aplicaciones.

La sintaxis es la siguiente:

```
Switch (expresión de control )
{
        case<literal-1>:
                Instrucción(es)
        break;
        .
        .
        case<literal-n>:
                Instrucción(es)
                break;
        default:
                Instrucción(es)
}
```

Caracteres ASCII utilizados:

ALT + 92 = \	ALT + 91 = [	ALT + 93 =]	ALT + 35 = #
ALT + 94 = ^	ALT + 123 = {	ALT + 125 = }	ALT +124 = |

EJEMPLOS DE ESTRUCTURAS SELECTIVAS

Ejemplo2

Desarrollar un programa que solicite al usuario un número y determine si dicha cantidad es par o impar, en caso de que el número sea par, el programa deberá verificar si el número está entre el rango [10 - 100].

```
1   static void Main(string[] args)
2   {
3    int Numero;
4    Console.Title = "VERIFICANDO NUMEROS PARES";
5    Console.Write("\nIngrese un numero:");
6    Numero = int.Parse(Console.ReadLine());
7    if (Numero < 0)
8    {
9    Console.Write("\n\tNumero Negativo...Ingrese un numero positivo");

10   }
11   else if (Numero % 2 == 0)
12   {
13   Console.Write("\n\tEl numero (" + Numero + ") es par.");
14    if (Numero >= 10 && Numero <= 100)
15   {
16   Console.Write("\n\tEl numero (" + Numero + ") se encuentra en el rango [10-100]");
17   }
18   else
19   {
20   Console.Write("\n\tEl numero (" + Numero + ") NO esta en el rango [10-00]");
21   }
22   }
23  else
24  {
25  Console.Write("\n\tEl numero ingresado es impar...");
26  }
27  Console.ReadKey();
28  }
29  }
30 }
```

Ejemplo6

Se necesita un programa que muestre un menú con las siguientes opciones:

1. Suma.
2. Resta.
3. Multiplicación.
4. División.
5. Raíz Cuadrada.
6. Exponenciación.
7. Salir del programa.

```
1   static void Main(string[] args)
2   {
3    int op;
4    Double x, y, z;
5    Console.WriteLine("\nDigitar el primer numero:");
6   x = Double.Parse(Console.ReadLine());
7   Console.WriteLine("\nDigitar el segundo numero:");
8   y = Double.Parse(Console.ReadLine());
9   Console.Title = "SWITCH CASE";
10  Console.WriteLine("\n\tOPERACIONES MATEMATICAS");
11  Console.WriteLine("\n\t1. SUMAR");
12  Console.WriteLine("\n\t2. RESTAR");
13  Console.WriteLine("\n\t3. MULTIPLICAR");
14  Console.WriteLine("\n\t4. DIVIDIR");
15  Console.WriteLine("\n\t5. RAIZ CUADRADA");
16  Console.WriteLine("\n\t6. POTENCIACION");
17  Console.WriteLine("\n\t7. SALIR DEL PROGRAMA");
18  Console.WriteLine("\n\n\t Digite su opcion: ");
19  op = int.Parse(Console.ReadLine());
20  switch (op)
21  {
22   case 1:
23   z = x + y;
24   Console.WriteLine("\t El resultado de la suma es: " + z);
25   break;
26   case 2:
27   z = x - y;
28   Console.WriteLine("\t El resultado de la resta es: " + z);
```

```
29        break;
30        case 3:
31        z = x * y;
32        Console.WriteLine("\t El resultado de la multiplicacion es: " + z);
33        break;
34        case 4:
35        if ( y == 0 )
36         {
37          Console.WriteLine("\t Division Invalida...");
38         }
39        else
40         {
41          z = x / y;

42          Console.WriteLine("\t El resultado de la division es: " + z);
43         }
44       break;
45       case 5:
46       z = Math.Sqrt(x);
47       Console.WriteLine("\t La raiz cuadrada del primer numero es " + z);
48       break;
49       case 6:
50       z = Math.Pow(y,2);
51       Console.WriteLine("\t El cuadrado del segundo numero es " + z);
52       break;
53       case 7:
54       Environment.Exit(0);
55       break;
56       default:
57       Console.WriteLine("\t Opcion no definida.. intente de nuevo");
58       break;
59      }
60     Console.ReadKey();
61    }
62   }
63  }
```

EJERCICIOS DE ESTRUCTURAS SELECTIVAS

1. Dado como dato el sueldo de un trabajador, aplique un aumento del 15% si su sueldo es mayor o igual a $550. Imprimir en ese caso el nuevo sueldo del trabajador.
2. Dado el sueldo de un empleado, encontrar el nuevo sueldo si obtiene un aumento del 10% si su sueldo es inferior a $600, en caso contrario no tendrá aumento.
3. En un hospital se ingresa un paciente cobrándole $25 diarios por hospitalización. Si el paciente es operado deberá además cancelar $1000 por los gastos más 20% del pago total por honorarios del doctor. Dados n días que estuvo el paciente, escriba el nombre, número de días que estuvo ingresado y el detalle de todos los pagos hechos.
4. En un supermercado se hace una promoción mediante la cual el cliente obtiene un descuento dependiendo de un número que escoge al azar. Si el número escogido es menor a 74 el descuento es del 15% sobre el total de la compra y si es mayor o igual a 74 es de 20%. Obtener cuánto dinero se le descuenta y el pago final.
5. Un obrero necesita calcular su salario semanal, el cual se obtiene de la siguiente manera:
   Si trabaja 40 horas o menos se le paga $4 por hora.
   Si trabaja más de 40 horas se le paga $4 por cada una de las primeras 40 horas y $6 por cada hora extra.
   Calcule el nuevo salario del obrero.
6. En una tienda de descuento se efectúa una promoción en la cual se hace un descuento sobre el valor de la compra total según el color de la bolita que el cliente saque al pagar en caja. Si la bolita es blanca no se le hará descuento alguno, si es verde se le hará un 10% de descuento, si es amarilla un 25%, si es azul un 50% y si es roja un 100%. Determinar la cantidad final que un cliente deberá pagar por su compra. Se sabe que sólo hay bolitas de los colores mencionados.
7. Elaborar un sistema de facturación que pida el nombre del vendedor, nombre del cliente, fecha de factura, número de factura y suma de ventas realizadas. A la suma de ventas realizadas, se les hará un descuento según la siguiente tabla:

VENTAS	DESCUENTO
0 < V < 100	0%
100 <=V <=500	15%
500 < V <=1000	20%
V>1000	30%

8. Leer como datos el modelo de un vehículo y su precio, determinar el valor final que debe pagar el comprador. El concesionario está haciendo descuentos, teniendo en cuenta el modelo con base en la siguiente tabla:

MODELO	DESCUENTO
Volskwagen	8%
Toyota	9%
Hyundai	6%
Mazda	5%