**UNIDAD 1. INTRODUCION A LA PROGRAMACION.**

La racionalidad es la capacidad que permite pensar, evaluar, entender y actuar de acuerdo a ciertos principios lógicos y de [consistencia](https://es.wikipedia.org/wiki/Consistencia_l%C3%B3gica), con el fin de satisfacer algún objetivo o finalidad.

Nuestro actuar como personas se basa en la capacidad que tiene el hombre como único ser racional de la naturaleza, somos el único individuo que desarrolla instrucciones secuenciales y lógicas de acuerdo al pensamiento.

La evolución de la sociedad ha permitido que hoy en día un computador pueda realizar tareas y manejar datos obedeciendo secuencias de pasos lógicos para lo cual ha sido programado por el hombre a través algoritmos los cuales permiten solucionar problemas por medio de la escritura secuencial paso a paso del lenguaje natural y organizado para luego llevarlos a un computador mediante un lenguaje de programación.

* 1. **QUE ES UN ALGORITMO**. Un algoritmo es una secuencia lógica y finita de pasos que permite solucionar un problema o cumplir con un objetivo.

Los algoritmos deben ser precisos e indicar el orden lógico de realización de cada uno de los pasos, debe ser definido y esto quiere decir que si se ejecuta un algoritmo varias veces se debe obtener siempre el mismo resultado, también debe ser finito ósea debe iniciar con una acción y terminar con un resultado o solución de un problema.

 Cuando se elabora un algoritmo tenga en cuenta lo siguiente.

* Tener claro cuál es el problema que va a solucionar.
* Establecer un objetivo que permita medir la solución del problema.
* Elaborar un algoritmo que solucione el problema.
* Realizar pruebas al algoritmo para verificar los resultados.

Ejemplo 1. Elabore un algoritmo que permita ir de la casa al trabajo.

 Objetivo: Ir de la casa al Trabajo.

 Inicio.

1. Salir de la casa
2. Si está lejos del lugar de trabajo entonces tomar un medio de transporte que lo deje cerca del mismo.
3. Si no está lejos del lugar de trabajo entonces dirigirse caminando hacia el mismo
4. Llegar a la puerta del trabajo

Fin

Ejemplo 2. Plantee un algoritmo que permita adquirir un boleto para ir a cine.

 Objetivo. Adquirir un boleto para ir a cine.

Inicio

1. Dirigirse hacia el teatro donde quiere ver la película
2. Si hay gente esperado el boleto entonces hacer la fila y avanzar con la misma hasta llegar a la taquilla
3. Si no hay gente esperando comprar el boleto entonces dirigirse a la taquilla
4. Comprar el boleto para ver la película.

Fin

Existen dos tipos de algoritmos. Los algoritmos convencionales que son los que hacen las personas todos los días en sus diferentes actividades para solucionar problemas del trabajo o de la vida cotidiana. Y los algoritmos computacionales que son los que utilizan el computador a través de programas diseñados por desarrolladores de software y programadores para dar solución a problemas de cálculo o de manejo de información.

* 1. **LOS DIAGRAMAS DE FLUJO.** Un diagrama de flujo permite representar de forma gráfica un algoritmo a través de símbolos.

****

**R1.**

Las entradas son datos o insumos que necesita el algoritmo para que se pueda elaborar.

Los procesos son las acciones que permiten transformar las entradas (insumos o datos) en otros datos u otros insumos que permitirán dar solución el problema.

Las salidas hacen referencia a los resultados que debe dar al final el algoritmo.

Las decisiones se usan para tomar decisiones lógicas y de acuerdo a estas ejecutar o no conjuntos de instrucciones.

Las iteraciones permiten repetir un conjunto de instrucciones dentro de un algoritmo

Para elaborar diagrama de flujo se deben tener en cuenta las siguientes reglas:

* Los diagramas se deben realizar de arriba hacia abajo y de izquierda a derecha.
* El algoritmo debe arrancar con el símbolo de inicio y terminar con símbolo de fin.
* La dirección de flujo se debe representar por medio de flechas.
* Todas las líneas de flujo deben llegar a un símbolo o a otra línea.

Ejemplo 1. Elabore un algoritmo que permita ir de la casa al trabajo.

|  |  |
| --- | --- |
| Inicio.1. Salir de la casa
2. Si está lejos del lugar de trabajo entonces tomar un medio de transporte que lo deje cerca del mismo.
3. 3. Si no está lejos del lugar de trabajo entonces dirigirse caminando hacia el mismo.
4. 4. Llegar a la puerta del trabajo

Fin |  |

**R2.**

 Ejemplo 2. Plantee un algoritmo que permita adquirir un boleto para ir a cine.

|  |  |
| --- | --- |
| Inicio1. Dirigirse hacia el teatro a donde quiere ver la película
2. Si hay gente esperado el boleto entonces hacer la fila y avanzar con la misma hasta llegar a la taquilla
3. Si no hay gente esperando comprar el boleto entonces dirigirse a la taquilla
4. Comprar el boleto para ver la película.

Fin |  |

**R3.**

* 1. **LOS LENGUEJES DE PROGRAMACION**

Los programas de computadora están en todas partes: desde máquinas de café y reproductores MP3, satélites, robots, teléfonos celulares, relojes etc. Pero así como la astronomía es más que usar telescopios, la ciencia de la computación es más que usar computadoras, es programarlas para decirles exactamente qué deben hacer.

Un lenguaje de programación es un sistema estructurado bajo un lenguaje formal (código) y diseñado principalmente para que las máquinas y computadoras puedan entender los algoritmos computacionales escritos por programadores. Los computadores solo entienden el lenguaje binario (cadenas interminables de números 1 y 0), para el hombre es complejo es complejo programar computadores mediante el sistema binario y es por eso que se crearon los lenguajes de programación, el lenguaje de programación se encarga entonces de convertir las instrucciones de los programadores de un lenguaje formal a unos (1) y ceros (0) para que las computadoras y las maquinas las puedan entender e interpretar.

Los lenguajes de programación se pueden clasificar de varias formas, los hay por nivel, y por la forma como se ejecutan.

Por nivel: Hay lenguajes de programación de alto nivel y lenguajes de bajo nivel. Los lenguajes de alto nivel permiten que con pocas palabras se logre hacer lo mismo que se logra con un lenguaje de bajo nivel.

Ejemplos de lenguajes de alto nivel: C++, Python, Java

Ejemplos de lenguaje de bajo nivel: assembler

Por la forma como se ejecutan Hay lenguajes compilados e interpretados.

Los lenguajes compilados necesitan de un programa especial que lea el código fuente y cree un archivo binario ejecutable para una plataforma específica. Ejemplo: C++, Pascal.

Los lenguajes interpretados necesitan de un programa que traduzca en directo el código fuente escrito a instrucciones de la plataforma en la que se ejecutan. Ejemplo Python, JavaScript.

Los lenguajes compilados son más rápidos, mientras que los lenguajes de programación interpretados son más lentos, esto debido a que al compilar un programa las ordenes son más entendibles para la computadora, mientras que al interpretarlo la máquina primero debe leer el código y convertir al paso las instrucciones a instrucciones de máquina entendibles para ella.

En la actualidad existen decenas de lenguajes de programación, algunos de los lenguajes utilizados hoy en día son [php](http://www.alegsa.com.ar/Dic/php.php), [ASP](http://www.alegsa.com.ar/Dic/asp.php).net , [ActionScript](http://www.alegsa.com.ar/Dic/actionscript.php), [ada](http://www.alegsa.com.ar/Dic/ada.php), python, [delphi](http://www.alegsa.com.ar/Dic/pascal.php), c++, visual [Basic](http://www.alegsa.com.ar/Dic/basic.php), , matlab, [JAVA](http://www.alegsa.com.ar/Dic/java.php), [JavaScript](http://www.alegsa.com.ar/Dic/javascript.php), etc.



**R4.**

**1.4 EL LENGUEJE JAVA SCRIPT**

Javascript es un lenguaje de programación interpretado utilizado para crear pequeños programas encargados de realizar acciones dentro de una página web.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc.

En este curso de fundamentos de la programación se usara la sintaxis de javascript para explicar conceptos básicos de la programación, elaborar ejercicios y talleres.

Nota: este curso no es un curso de JavaScript, es más bien un curso de fundamentos de la programación que usa como lenguaje base el lenguaje de programación javascritp

Lo único que debe tener en cuenta es que el código JavaScript se encierra entre etiquetas <script>

Ejemplo.

<script>

 alert("Bien venido al curso de fundamentos de programación");

</script>

**UNIDAD 2 MANEJO DE VARIABLES, ENTRADA Y SALIDA DE DATOS**

* 1. **QUE ES UNA VARIABLE.** Una Variable es un espacio de la memoria del computador que permite almacenar información de un determinado tipo de dato.

El tipo de dato indica como es el dato que se almacena en la variable, en programación los tipos de datos básicos son los numéricos, los carácter y los lógicos.

Las variables de tipo numérico se utilizan para almacenar valores numéricos, ya sean enteros o reales con los cuales se pueden realizar operaciones aritméticas como la suma, resta, multiplicación, división entre otras.

Ejemplo.
<script>
variva = 16;// variable tipo entero
var total = 58945; // variable tipo decimal
<script>

Las variables de tipo carácter se usan para almacenar uno solo o varios caracteres como los que forman una palabra o una frase, estos deben estar encerrados entre comillas dobles o simples, con el fin de que el lenguaje de programación las pueda reconocer como variables de tipo carácter.

Ejemplo
<script>

var nombre = “Carlos Andres”;

varDireccion= “Cll 24 #5-18”;

var sexo “M”;

</script>

Las variables de tipo lógicas se conocen como boolean o booleano y permiten almacenar uno de los dos estados lógicos (verdadero o falso).

Ejemplo
varclienteRegistrado=false;

varivaIncluido=true;

Nota:

* Las variables de JavaScript se crean de la misma forma (mediante la palabra reservada var), la forma en la que se les asigna un valor depende del tipo de valor que se quiere almacenar (números, textos, etc.)
* Los nombre de Variables se conocen también como identificadores, un identificador es una secuencia de caracteres de letras, dígitos o subrayados “\_”, en donde el primer carácter debe ser una letra o un subrayado y después de este pueden ir otras letras, subrayados o dígitos. Las letras mayúsculas y minúsculas son diferentes, se recomienda que los nombres de variable tengan entre 4 y 12 caracteres y que estos hagan referencia al dato que se almacenara en la variable

Ejemplo:

Var ValorHora;

Var Num\_horas;

varNomCli;

var Cedula;

Para Asignar una variable un dato usa el operador igual (=).

Sintaxis:
NombreVariable = Expresión;

Ejemplo:
<script>
var base;
var altura;
vararea;
 base = 10;
 altura =5;
area = (base \* altura)/2;
</script>

**2.2. JERARQUIA DE OPERADORES.** En programación una expresión es una fórmula aritmética que permite calcular un valor, cuando se construye la formula se debe tener en cuenta la jerarquía de operadores. La jerarquía de operadores determina el orden en el que se resuelven las expresiones cuando se involucran operaciones aritméticas como la suma, resta, multiplicación, división y potencia.



Cuando dos operadores tienen el mismo nivel de prioridad, dentro de una expresión se evalúan de izquierda a derecha.

Ejemplo 1.



R5.

Ejemplo 2.



R6.

Cuando se desea determinar un orden específico de ejecución en una expresión aritmética, puede emplear los paréntesis para agrupar, de esta manera, las operaciones que se encuentren dentro del paréntesis serán las primeras en ejecutarse.

Ejemplo 3.



R7.

**2.3. LEER UN DATO (ENTRADA).** En programación, leer o entrar un dato hace referencia a la captura un dato (normalmente por el teclado) y almacenarlo en una variable.

Para leer un dato por el teclado en javascript se usa la función prompt, cada vez que se necesite ingresar un dato con esta función, aparecerá una ventana donde se escribirá el valor.

Sintaxis:
NombreVariable = prompt('Mensaje');

Ejemplo:
var a, b;
<script>

 a = prompt('Ingrese El valor de A');
</script>

R8.

En el ejemplo el dato que se captura por el teclado queda almacenado en la variable “a”

**2.4. ESCRIBIR (SALIDA).** En programación escribir hace referencia a mostrar un mensaje o el contenido de una variable en la pantalla.

En javascript para mostrar un mensaje en la pantalla o ver el contenido de una variable se usa la función document.write

Sintaxis:
document.write(“Mensaje”); // Para mostrar un mensaje
document.write (NombreVariable); // Para mostrar el contenido de una variable

|  |
| --- |
| Ejemplo 1. Elaborar un programa que capture por el teclado la base y la altura de un triángulo y que calcule y muestre en pantalla su áreaSolución.El objetivo del programa es calcular y mostrar en pantalla el área de un triángulo conociendo su base y altura.Siempre que se realiza un algoritmo es importante analizar cómo se va a elaborar el programa, para esto es importante mirar la solución desde el punto de vista de entradas procesos y salidasRecuerde que las entradas son datos que requiere el programa para poder realizar operaciones y procedimientos, los procesos son cálculos aritméticos y/o llamados a funciones que permiten obtener resultados, y las salidas son los mensajes y datos que el programa muestra al usuario en la pantalla del computador (los resultados). |
| Diagrama de Flujo | Código en Javascript |
| b‘Entre la Base’h‘Entre la Altura’a = (b\*h)/2‘El área del triangulo es ’ , a | <script>var base=0;var altura=0;vararea=0;base = prompt('Ingrese la base');altura = prompt('Ingrese la altura');area= (base\*altura)/2;document.write(' El Area del Triángulo es ' , area);</script> |

R9.

|  |
| --- |
| Ejemplo 2. Elaborar un programa que capture por el teclado el valor de una temperatura en grados Celsius, el programa debe calcular y mostrar en pantalla su valor equivalente grados Fahrenheit.Solución.El objetivo del programa es convertir una temperatura en grados Celsius a grados Fahrenheitgrados\_F = grados\_C \* 1.8 + 32 Grados\_Cgrados\_F |
| Diagrama de Flujo | Código en Javascript |
|  | <script> var grados\_C=0; var grados\_F=0;   grados\_C = prompt('Escriba la temperatura en  grados Celsius');  grados\_F= grados\_C \* 1.8 + 32 ;   document.write(' la temperatura en grados  Farenheit es ' , grados\_F);</script> |

R10.

Nota:

* Cuando este capturando datos numéricos en javascript se recomienda que el dato que se captura con la función prompt se convierta a numérico y para esto puede usar la funcione parseInt () para convertir el dato a tipo entero y parseFloat () para convertir el dato a tipo real.

Ejemplo:

<script>

 var n1 = parseFloat(prompt("Dime un numero"));

 var n2 = parseFloat(prompt("Dime otro numero"));

 document.write("La suma de "+ n1 + " y "+ n2 + " es: ");

 document.write(n1+n2);

</script>

**UNIDAD 3. ESTRUCTURAS DE CONTROL (PARTE 1)**

Las estructuras de control permiten determinar la secuencia de ejecución de las sentencias o instrucciones de un programa. En programación existen tres tipos de estructuras de control, la secuencia, las de decisión (también llamadas selectivas o condicionales) y las repetitivas.

**R11.**

La estructura secuencial permite ejecutar las instrucciones de un programa en forma de secuencia, cada una de las instrucciones están separadas por el carácter punto y coma (;) y se ejecutan una después de la otra.

Las estructuras de decisión (condicionales) determinan si se ejecutan un grupo de instrucciones u otras según se cumpla o no una determinada condición. En programación existen básicamente tres tipos de estructuras de decisión, el if, el if – else y la estructura switch

Las estructuras repetitivas permiten ejecutar de forma repetida un bloque específico de instrucciones. Existen tres tipos de ciclos o estructuras repetitivas, la estructura while, la do – while y la estructura for.

**3.1. MANEJO DE OPERADORES EN PROGRAMACIÓN**. Los operadores en programación se dividen en aritméticos, relacionales y lógicos.

**3.3.1. Operadores aritméticos.** Se usan para realizar operaciones de la matemática básica como la suma, resta, multiplicación, división.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Símbolo** | **Acción** |
| Suma | + | Suma dos operados |
| Resta | - | Resta el segundo operando del primero |
| Multiplicación | \* | Multiplica los dos operandos |
| División | / | Divide el primer operando por el segundo |
| Resto (módulo) | % | Facilita el resto de la división entre dos operandos (residuo de la división) |

* + 1. **Operadores de Relacionales.** Los operadores relacionales se usan para evaluar expresiones condicionales y se basan en el concepto de verdadero o falso

|  |  |  |
| --- | --- | --- |
| **Operador** | **Acción** | **Ejemplo** |
| > | Mayor que | 3 > 4 FALSO8 > 5 VERDADERO |
| >= | Mayor o igual | 3 >= 3 VERDADERO4 >= 4 FALSO |
| < | Menor que | 4 < 6 VERDADERO7 < 4 FALSO |
| <= | Menor o igual | 2 <= 2 VERDADERO3 <= 2 FALSO |
| == | Igual | 4 = 4 VERDADERO3 = 4 FALSO |
| != | Distinto | 6 != 7 VERDADERO7 != 7 FALSO |

* + 1. **Operadores Lógicos.** Los operadores lógicos son usados para concatenar dos o más expresiones con operadores relacionales.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Símbolo** | **Significado** |
| AND | && | AND lógico |
| OR | || | OR lógico |
| NOT | ! | Negación |

Ejemplos.

3 > 2 && 4 < 5 **VERDADERO**, porque ambas expresiones son verdaderas

3 > 2 && 4 < 3 **FALSO**, porque hay una expresión falsa

3 > 2 || 4 < 3 **VERDADERO**, Porque hay una expresión verdadera

6 < 4 || 7 > 8 **FALSO**, Porque ambas expresiones son verdaderas

El operador lógico “and” solo da como resultado Verdadero si ambas expresiones son verdaderas.

|  |  |  |
| --- | --- | --- |
| Expresión A | Expresión B | A && B |
| Verdadero | Verdadero | Verdadero |
| Verdadero | Falso | Falso |
| Falso | Verdadero | Falso |
| Falso | Falso | Falso |

El operador “or” da como resultado Verdadero cuando al menos una de las expresiones sea verdadera.

|  |  |  |
| --- | --- | --- |
| Expresión A | Expresión B | A || B |
| Verdadero | Verdadero | Verdadero |
| Verdadero | Falso | Verdadero |
| Falso | Verdadero | Verdadero |
| Falso | Falso | Falso |

* 1. **ESTRUCTURA DE DECISION SI SIMPLE (IF).** Se evalúa la condición y si ésta se cumple se ejecuta un determinado grupo de instrucciones. En caso contrario se saltan dicho grupo de instrucciones.

|  |  |
| --- | --- |
|  | **Forma Lógica:**1. Evalúa la condición, en caso de que sea verdadera va al paso #2; en caso de que la condición sea falsa va al paso #3.
2. Ejecuta las instrucciones de la Parte Verdadera y va al paso #3.
3. Termina la estructura Condicional y continua secuencia del programa.
 |

**R12.**

**Sintaxis**

if(condición)

{

 ...

}

Ejemplo:
Elaborar un algoritmo que permita capturar un número y que calcule su valor absoluto

|  |  |
| --- | --- |
|  | <script> var numero = prompt('Escriba un numero');  if (numero<0) { numero = numero \* (-1) }     document.writeln('Su valor Absoluto es ' ,  numero); </script> |

R13.

* 1. **ESTRUCTURA DE DECISION SI COMPUESTA ( IF-ELSE).** Se evalúa la condición y si ésta se cumple se ejecuta un determinado grupo de instrucciones. Si no se cumple se ejecuta otro grupo de instrucciones.

|  |  |
| --- | --- |
|  | **Forma Lógica:**1. Evalúa la condición, en caso de que sea verdadera va al paso #2; en caso de que la condición sea Falsa va al paso #3.
2. Ejecuta las instrucciones de la parte verdadera y va al paso #4
3. Ejecuta las instrucciones de la parte falsa y va al paso #4.
4. Termina la estructura condicional y continua la secuencia del programa.
 |

**R14.**

**Sintaxis**

if(condicion)

 {

        //instrucciones (Parte Verdadera)

 }

else

 {

        //instrucciones (Parte Falsa)

 }

**Ejemplo:**Elaborar un algoritmo que permita capturar un número y que indique si el número capturado es positivo o negativo.

|  |  |
| --- | --- |
|  | <script>var numero = prompt('Escriba un numero');if (numero>=0) { document.writeln('El número es ', numero, '  positivo'); } else { document.writeln('El número es ', numero, '  negativo'); }</script> |

R15.

**3.4 ESTRUCTURA DE SELECCION POR CASOS (SWITCH).** Es una estructura de selección múltiple en el que se evalúa el valor de una expresión, de acuerdo a dicho valor se pueden ejecutar un conjunto de instrucciones según el caso.

|  |  |
| --- | --- |
|  | Forma lógica1. Se evalúa la expresión
2. En caso de que el valor de la expresión sea el valor\_1 ejecuta las instrucciones del valor\_1 y va al #6
3. En caso de que el valor de la expresión sea el valor\_2 ejecuta las instrucciones del valor\_2 y va al #6
4. En caso de que el valor de la expresión sea el valor\_n ejecuta las instrucciones del valor\_n y va al #6
5. En caso de que el valor de la expresión no se alguno de los anteriores ejecuta las instrucciones por defecto y va al #6
6. Termina la estructura switch y continua la secuencia del programa
 |

R16.

Sintaxis

switch (expresión)

{

case valor\_1:

// instrucciones valor\_1;

break;

case valor\_2:

// instrucciones valor\_2;

break;

 · · ·

casevalor\_n:

// instruccionesvalor\_n;

break;

default:

//instrucciones por defecto;

 }

* La expresión normalmente es una variable usada como ***selector*** puede ser de tipo numérico o carácter según el caso.
* Cada **etiqueta** (valor\_1, valor\_2,…, valor\_n) es un valor único y constante.
* La palabra **break** permite terminar la estructura switch, provocando que el control del programa pase a la primera instrucción después de la estructura **switch**.

Ejemplo:

Elaborar un algoritmo que capture el valor de un mes en número y que muestre en pantalla su equivalente en letras.

|  |  |
| --- | --- |
|  | <script> var mes=''; var numero = prompt('Escriba un número de mes  (1-12)?'); switch (numero) { case '1': mes="enero"; break; case '2': mes="febrero"; break; // // ... (complete el programa) // case '12': mes="diciembre"; break; default: mes="no existe"; } document.writeln('El mes ' , numero , '  corresponde a ', mes);</script> |

R17.

**UNIDAD 4 ESTRUCTURAS DE CONTROL (PARTE 2)**

Algunas veces cuando se está diseñando un algoritmo computacional es necesario repetir de forma controlada un conjunto de instrucciones. Las estructuras repetitivas también llamadas ciclos permiten repetir una secuencia de instrucciones un número determinado de veces, al hecho de repetir la ejecución de una secuencia de acciones se le llama iteración.

En programación existen tres tipos de estructuras repetitivas, el ciclo mientras, el ciclo hacer – mientras y la estructura para

**4.1 ESTRUCTURA MIENTRAS QUE (WHILE).** La estructura de control while permite ejecutar una serie de instrucciones de forma repetida mientras que la condición que controla la estructura sea verdadera.

|  |  |
| --- | --- |
|  | Forma lógica1. Se evalúa la condicional que existe dentro del ciclo while, en caso de que la condición sea verdadera va al #2; en caso de que la Condicional sea falsa va al #3.
2. Ejecuta las instrucciones de la Parte Verdadera y vuelve al #1.
3. Termina el ciclo while y continua la secuencia del programa.
 |

**R18.**

**Sintaxis**

 While (condición)

{

 --------------

 --------------

 --------------

}

|  |
| --- |
| Ejemplo 1:Elaborar un algoritmo que capture un número y que realice la tabla de multiplicar del uno al diez de dicho número. |
|  | <script> var n, m, i;  n = prompt('Por favor ingresa un número');   i=1;  while (i<=10) {  m= n \* i;  document.writeln(n , ' \* ' , i ,' = ' , m , '<br>');  i++; }</script> |

R19.

|  |
| --- |
| Ejemplo 2: Elaborar un algoritmo que muestre en pantalla los números pares que hay entre el 2 y el 50 |
|  | <script> var i = 2;  while (i<=50) { document.writeln(i, '<br>'); i = i +2; }</script> |

R20.

**4.2 ESTRUCTURA PARA (FOR).** Esta estructura permite ejecutar una serie de instrucciones un número determinado de veces

|  |  |
| --- | --- |
|  | **Forma lógica:**1. Iniciativa de una variable declarada que permite controlar el ciclo (inicialización)
2. Se evalúa la condición y si es verdadera va al #3, si la condición es falsa va al #5
3. Ejecuta las instrucciones que hay dentro del ciclo
4. Incrementa la variable que controla el ciclo y retorna al #2
5. Termina el Ciclo for y continua la secuencia del programa.
 |

R21.

**Sintaxis**

For(inicialización; condición; incremento)

{

 -----------------

 -----------------

 -----------------

}

|  |
| --- |
| Ejemplo1:Elaborar un algoritmo que capture un número y que realice la tabla de multiplicar del uno al diez de dicho número |
|  | <script> var n, m, i;  n = prompt('Por favor ingresa un número');  for (i=1; i<=10; i++) {  m= n \* i; document.writeln(n , ' \* ' , i ,' = ' , m , '<br>'); }</script> |

R22.

|  |
| --- |
| Ejemplo 2: Elaborar un algoritmo que muestre en pantalla los números pares que hay entre el 2 y el 50 |
|  | <script> var i ;  for (i=2 ; i <=50; i = i+2) { document.writeln(i, '<br>');  }</script> |

**R23.**

**4.3 ESTRUCTURA HACER – MIENTRAS (Do-While).**Al igual que la en la estructura while Las instrucciones se ejecutan mientras la condición sea verdadera, la diferencia está en que en la estructura do – while la condición se comprueba al final del ciclo por lo que las instrucciones se ejecutarán al menos una vez.

|  |  |
| --- | --- |
|  | Forma lógica1. Ejecuta las instrucciones que hay dentro del ciclo do – while y va al #2.
2. Evalúa condición del ciclo do –while, caso de que la condición sea verdadera retorna al #1; en caso de que la condición sea falsa va al #3.
3. Termina el Ciclo do - while y continua la secuencia del programa.
 |

**R24.**

**Sintaxis**

do

{

 --------------

 --------------

 --------------

}

while (condición);

|  |
| --- |
| Ejemplo 1.Elaborar un algoritmo que capture un número y que realice la tabla de multiplicar del uno al diez de dicho número. |
|  | <script> var n, m, i; n = prompt('Por favor ingresa un número');  i=1; do { m= n \* i; document.writeln(n , ' \* ' , i ,' = ' , m , '<br>'); i++; }  while (i<=10);</script> |

R25.

|  |
| --- |
| Ejemplo 2: Elaborar un algoritmo que muestre en pantalla los números pares que hay entre el 2 y el 50 |
|  | <script> var i = 2;  do { document.writeln(i, '<br>'); i = i +2; } while (i<=50); </script> |

R26.