

SISTEMA BINARIO Y RECURSIVIDAD

MARIANA HENAO MORALES
LAURA QUINTERO MONTOYA
22 DE OCTUBRE DE 2020

1 CONTENIDO

1	CONTENIDO.....	1
2	PRESENTACIÓN.....	2
3	SISTEMA BINARIOS.....	3
3.1	SISTEMA POSICIONAL BINARIO	4
3.2	CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS	6
3.3	CONVERSIÓN: 0, 1 A PALABRAS: CERO, UNO.....	9
3.4	CONTAR NÚMERO DE UNOS EN UN BINARIO.....	11
3.5	CONTAR LA POSICION DE LOS NÚMERO (COMENZANDO DESDE EL 0) DE UN NÚMERO BINARIO	14
3.6	CONTAR LA POSICION DE LOS NÚMERO (COMENZANDO DESDE EL NÚMERO MENOR) DE UN NÚMERO BINARIO.....	17
4	CONVERSIÓN EXTENDIDA	20
5	COMPUERTAS AND, OR, NOT.....	24
6	RECURSIVILIDAD.....	25
7	CONCLUSIONES	29
8	BIBLIOGRAFÍA.....	30

2 PRESENTACIÓN

La presente monografía costará de todos los programas desarrollados en clase, el cual nos presentarán de forma clara todos los temas, conceptos y ejercicios desarrollados en la materia de Introducción a la informática. Teniendo como objetivo desarrollar de forma concisa el concepto de sistema numérico, con un enfoque en el sistema de base dos.

AUTOR: MARIANA HENAO MORALES

1007510604

mariana.henao2@utp.edu.co

<https://github.com/marianah5>

AUTOR: LAURA QUINTERO MONTOYA

1006574950

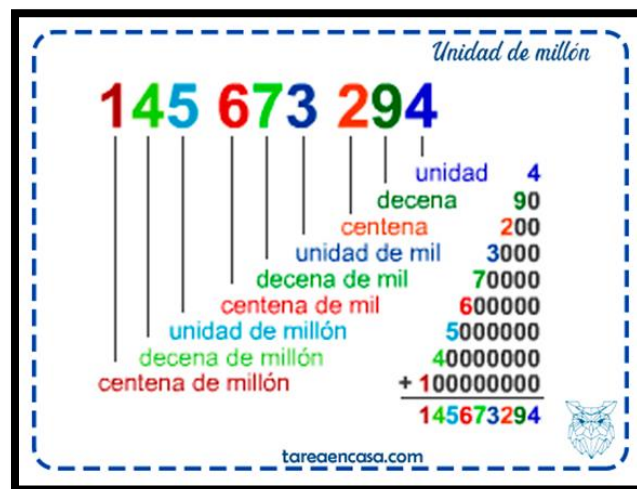
l.quintero2@utp.edu.co

<https://github.com/laura8812/informatica>

3 SISTEMA BINARIOS

Sistema **numérico binario**, en matemáticas, sistema numérico posicional que emplea 2 como base y, por lo tanto, requiere solo dos símbolos diferentes para sus dígitos, 0 y 1. La importancia del sistema binario para la teoría de la información y la tecnología informática se deriva principalmente del compacto y una manera confiable en la que los 0 y los 1 se pueden representar en dispositivos electromecánicos con dos estados.

Sistema Posicional Decimal: El sistema de numeración decimal (también llamado sistema de numeración posicional de base diez. es el sistema estándar para denotar números enteros y no enteros.

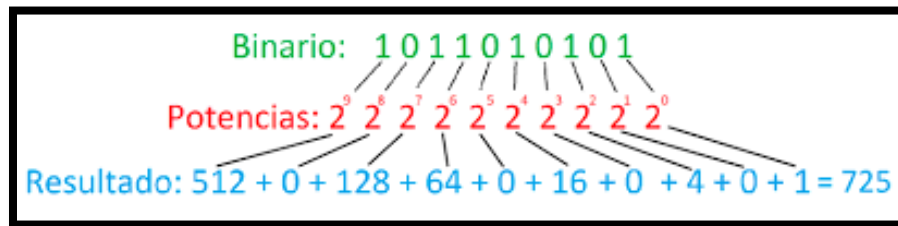


Potencias de 10: Las potencias de base 10 se utilizan para expresar números muy grandes. Toda potencia de base 10 es igual a la unidad seguida de tantos ceros como unidades indica el exponente.

$$\begin{aligned}
 100 &= 10 \cdot 10 \\
 &= 10^2 \\
 1000 &= 10 \cdot 10 \cdot 10 \\
 &= 10^3 \\
 10000 &= 10 \cdot 10 \cdot 10 \cdot 10 \\
 &= 10^4
 \end{aligned}$$

3.1 SISTEMA POSICIONAL BINARIO

A continuación, se presenta el algoritmo básico para la conversión numérica basada en divisiones sucesivas.



Como se ve en el diagrama, la conversión se realiza multiplicando las potencias dependiendo de su posición y si el número binario es un cero o un uno.

El resultado se obtiene sumando todo el resultado de las potencias de dos.

El proceso finaliza cuando no hay ningún número que dividir.

A continuación, se presentan las imágenes de los códigos requeridos, para implementar el proceso mostrado en JavaScript. Cada imagen presenta una función distinta, o la ejecución final del programa. Se debe escribir en un solo archivo el código mostrado, y se sugiere un entorno como repl.it.

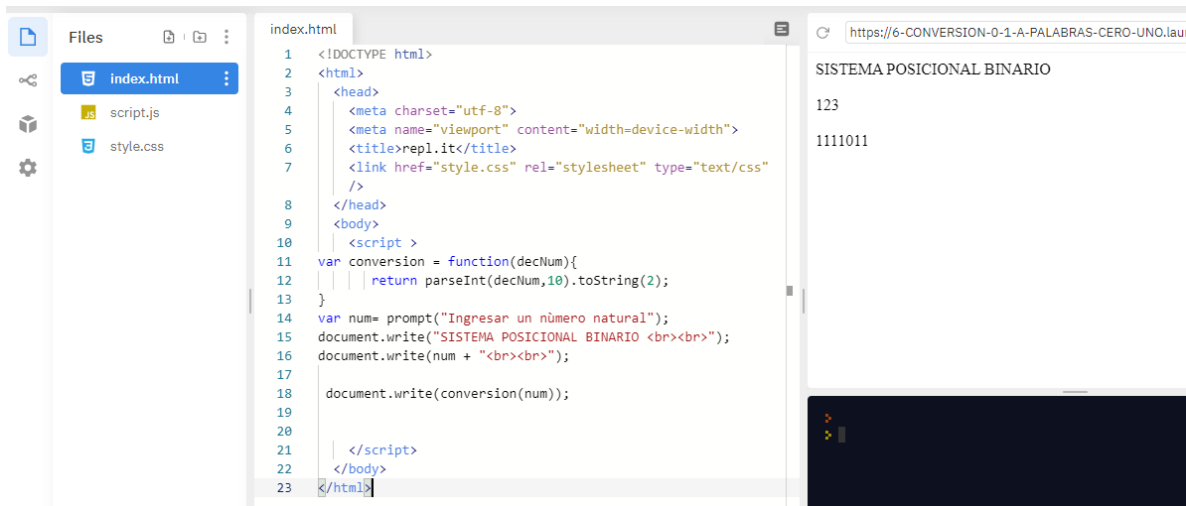
```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width">
6      <title>repl.it</title>
7      <link href="style.css" rel="stylesheet" type="text/css" />
8    </head>
9    <body>
10     <script>
11     var conversion = function(decNum){
12       return parseInt(decNum,10).toString(2);
13     }
14     var num= prompt("Ingresar un número natural");
15     document.write("SISTEMA POSICIONAL BINARIO <br><br>");
16     document.write(num + "<br><br>");
17
18     document.write(conversion(num));
19
20
21     </script>
22   </body>
23 </html>

```

Prácticamente lo que hace ese código es pasar los números enteros a números binarios, con la variable **“num”**, ingresamos el número, y con la variable **conversión** se hace la función **“decNum”** se hace la operación, se retorna los números con el **parseInt** que nos devolver un entero de la base especificada, en ese caso la base es 10.y ya luego con el **toString()** la cadena se vuelve texto. Ya para finaliza se llama la función **document.write**, para que se visualice en la página con el título, el numero que se ingresó y su resultado.

A continuación, se muestra el programa en el entorno repl.it, con los datos de ejecución del programa.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>repl.it</title>
7   <link href="style.css" rel="stylesheet" type="text/css" />
8 </head>
9 <body>
10 <script>
11 var conversion = function(decNum){
12   return parseInt(decNum,10).toString(2);
13 }
14 var num= prompt("Ingresar un número natural");
15 document.write("SISTEMA POSICIONAL BINARIO <br><br>");
16 document.write(num + "<br><br>");
17
18   document.write(conversion(num));
19
20
21 </script>
22 </body>
23 </html>
```

https://6-CONVERSION-0-1-A-PALABRAS-CERO-UNO.lau

SISTEMA POSICIONAL BINARIO

123

1111011

3.2 CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS

A continuación, se presenta el algoritmo básico para la conversión numérica basada en divisiones sucesivas.



Como se ve en el diagrama, la conversión se realiza dividiendo el número a convertir entre la base seleccionada.

El resultado se obtiene con base en los residuos de las divisiones.

El proceso finaliza cuando se obtiene cero en el resultado de las divisiones.

A continuación se presentan las imágenes de los códigos requeridos, para implementar el proceso mostrado en JavaScript. Cada imagen presenta una función distinta, o la ejecución final del programa. Se debe escribir en un solo archivo el código mostrado, y se sugiere un entorno como repl.it.

```
function texto( cadena, num_saltos = 0 ) {
  document.write( cadena );
  var i = 0;
  while (i < num_saltos ) {
    document.write( "<br />" );
    i = i + 1;
  }
}
```

```

function conversion( numero, base ) {
  var division, resto;
  var result = "";
  var control = 0;
  var bandera = 0;
  while ( bandera == 0 ) {
    ///////////////////////////////////
    division = Math.trunc( numero / base );
    resto = numero - division * base;
    result = resto.toString() + result;
    numero = division;
    if (numero <= 0) {
      bandera = 1;
    }
    control = control + 1;
    if ( control > 1000 ) {
      bandera = 1;
    }
  }
  return result;
}

```

```

texto( "PROGRAMA DE CONVERSIÓN NUMÉRICA", 1);
texto( "Octubre 13 de 2020");
texto( "", 2);

var n = 83; // El número a convertir
var b = 16; // La base de conversión

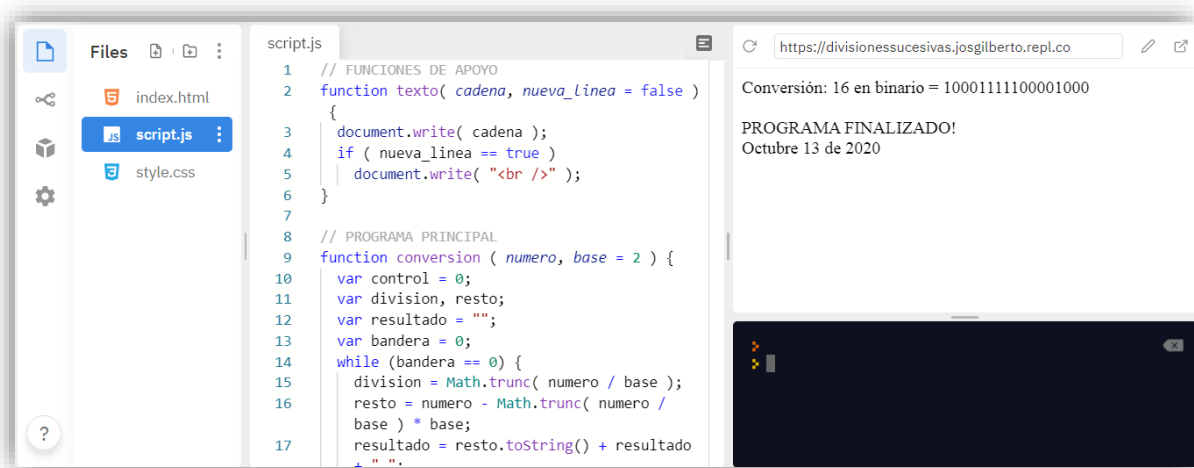
var resultado;
resultado = conversion( n, b );

texto( "Número: " + n, 1);
texto( "Base:   " + b, 1);
texto( "Resp:   " + resultado, 1);

</script>

```

A continuación se muestra el programa en el entorno repl.it, con los datos de ejecución del programa.

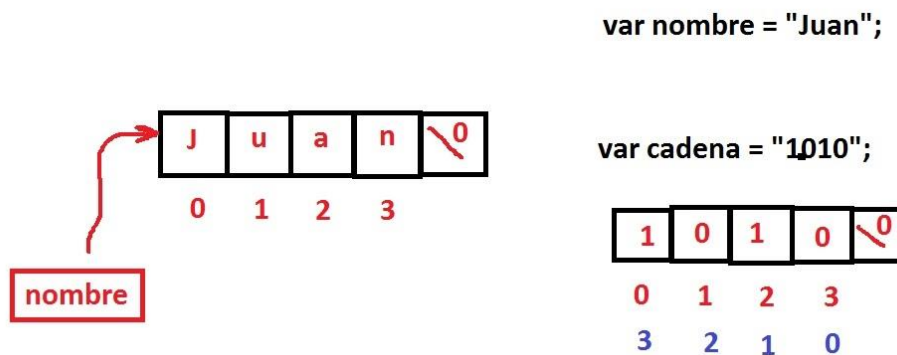


Es interesante como se, por medio de las variables, funciones y condiciones, podemos hacer una división y convertirlo en un numero decimal, todo eso por medio del programa de Javascript, y una estructura bien implantada, se crea varias operaciones para obtener un resultado deseado, las funciones **“Math.trunc”**, ha sido fundamental en el desarrollo de la actividad, debido a su características de poder dividir los números.

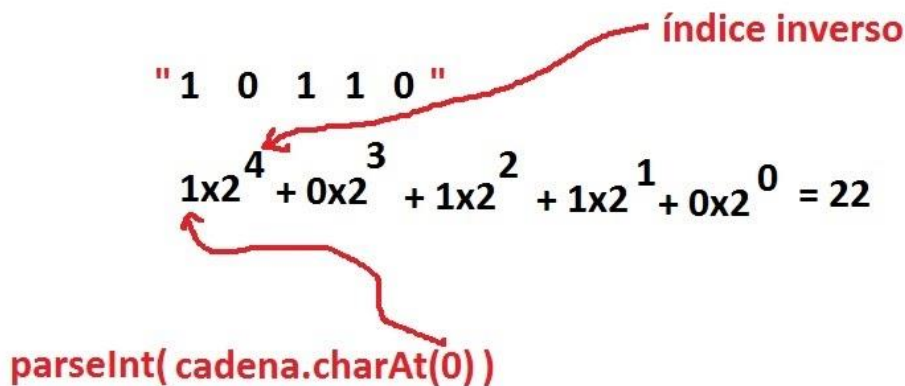
3.3 CONVERSIÓN: 0, 1 A PALABRAS: CERO, UNO

Vamos a presentar el programa que convierte los dígitos binarios 0, 1 a las palabras Cero, Uno.

Algunos elementos teóricos. En primer lugar, observamos cómo se almacena una cadena en la memoria:



En la siguiente gráfica, presentamos el índice inverso:



A continuación, presentamos el código fuente, seccionado por partes:

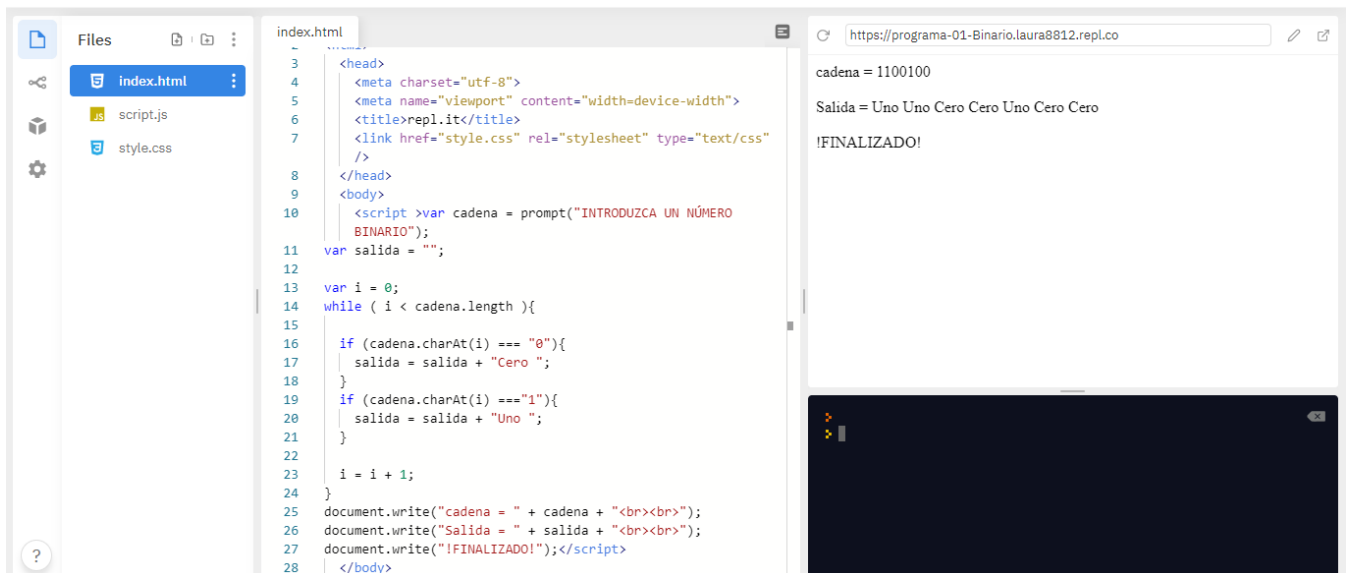
```

3 // PROGRAMA - 01
4 // -----
5 // Programa que permite leer en una cadena de texto, un número binario
6 // Los ceros y los unos contenidos en la cadena de texto
7 // Son convertidos a las palabras: Cero y Uno, respectivamente
8
9 var cadena = prompt("Introduzca un número binario");
10 var salida = "";
11
12 var i = 0;
13 while ( i < cadena.length ) {
14
15     if (cadena.charAt(i) === "0") {
16         salida = salida + "Cero ";
17     }
18     if (cadena.charAt(i) === "1") {
19         salida = salida + "Uno ";
20     }
21
22     i = i + 1;
23 }

```

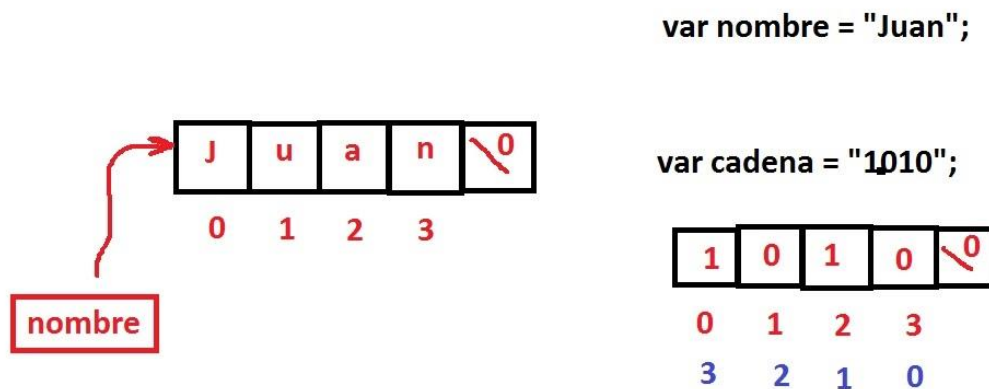
En estas líneas se comenta el programa, se crean algunas variables, se pregunta por una cadena binaria y se procede luego a realizar la conversión, dígito a dígito.

Al ejecutar en repl.it, se obtiene el resultado mostrado seguidamente:



3.4 CONTAR NÚMERO DE UNOS EN UN BINARIO

Vamos a presentar el programa que cuenta el número de unos disponibles en un binario. Veamos la estructura de la cadena que contiene el binario:



El número de unos se calcula recorriendo la cadena de izquierda a derecha, evaluando cada una de las posiciones en la cadena, primero creando las variables que se va necesitar en el ejercicio, en ese caso es la “cadena”, “contador” (que va ser igual a 0) y la “i” (que son los dígitos que se va leer el programa), cada vez que el contenido de dicha posición sea uno, sumaremos 1 a un contador preparado para ello.

Seguidamente presentamos el código fuente.

```

1  <script>
2  // PROGRAMA - 02
3  // -----
4  // Programa que permite leer en una cadena de texto, un número binario
5  // El programa cuenta el número de unos
6
7  var cadena = prompt("Introduzca un número binario");
8
9  var contador = 0; // Aquí se cuenta el número de unos
10
11 var i = 0;
12 while ( i < cadena.length ) {
13
14     if (cadena.charAt(i) === "1") {
15         contador = contador + 1;
16     }
17
18     i = i + 1;
19 }

```

```

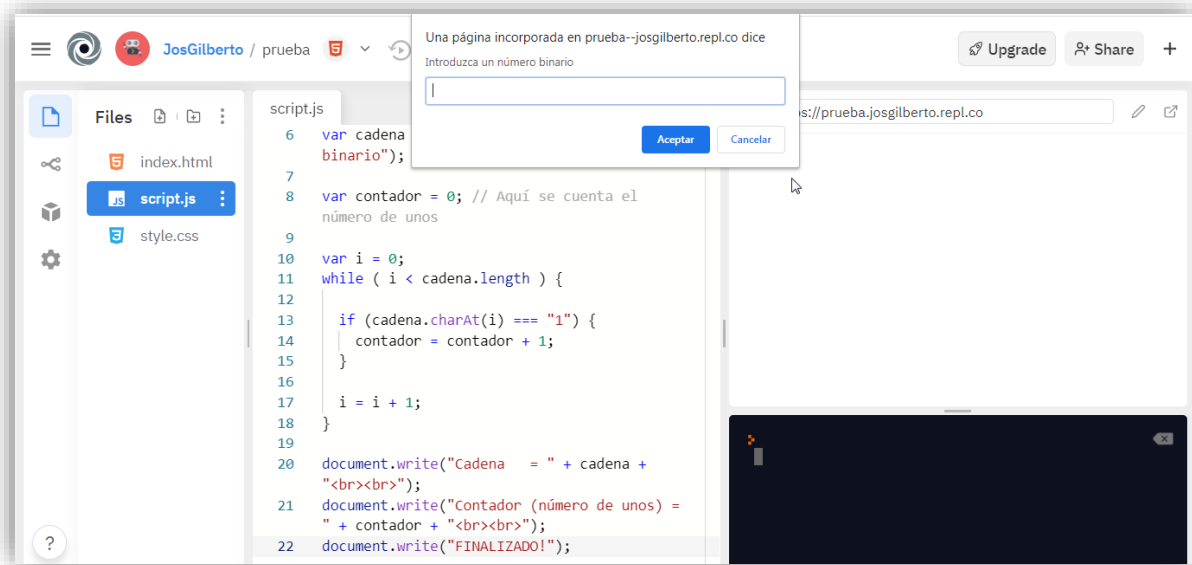
21 document.write("Cadena  = " + cadena + "<br><br>");
22 document.write("Contador (número de unos) = " + contador + "<br><br>");
23 document.write("FINALIZADO!");
24 </script>

```

En este código utilizamos **“prompt”** para ingresar el número binario que queremos contar sus números 1, entonces lo que se hace es crea un contador (que comienza desde cero) , y por cada digitito que “1” que se encuentra en la cadena, se le agregara “1” en el contador, todo eso con **“while (i < cadena.length)”**, con el **charAt** se analiza cada digito encontrando los “1” en la cadena.

Para finalizar con **“document.write”** se visualizara en la página, la cadena que ingresamos en primer lugar, los números “1” que se encontraron en esta y la finalización del programa.

Al ejecutar en repl.it, se obtiene el resultado mostrado seguidamente:



Una página incorporada en prueba--josgilberto.repl.co dice

Introduzca un número binario

Aceptar
Cancelar

The screenshot shows a web-based code editor interface. On the left, a file explorer shows 'index.html', 'script.js', and 'style.css'. The main editor displays a JavaScript script in 'script.js' that prompts the user to enter a binary number and counts the number of '1's. The script is as follows:

```

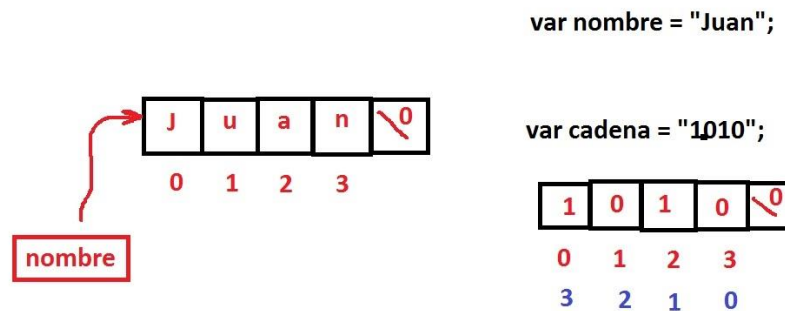
6  var cadena = prompt("Introduzca un número
7  binario");
8  var contador = 0; // Aquí se cuenta el
   número de unos
9
10 var i = 0;
11 while ( i < cadena.length ) {
12
13   if (cadena.charAt(i) === "1") {
14     contador = contador + 1;
15   }
16
17   i = i + 1;
18 }
19
20 document.write("Cadena   = " + cadena +
   "<br><br>");
21 document.write("Contador (número de unos) =
   " + contador + "<br><br>");
22 document.write("FINALIZADO!");

```

On the right, the output of the script is displayed in a terminal-like window. It shows the input string 'Cadena = 11110110101', the calculated count 'Contador (número de unos) = 7', and the final status 'FINALIZADO!'.

3.5 CONTAR LA POSICION DE LOS NÚMERO (COMENZANDO DESDE EL 0) DE UN NÚMERO BINARIO

Vamos a presentar el programa que cuenta el número de unos disponibles en un binario. Veamos la estructura de la cadena que contiene el binario:



El número de unos se calcula recorriendo la cadena de izquierda a derecha, Programa que lee en una cadena de texto un número binario, primero creando las variables que se va necesitar en el ejercicio, en ese caso es la “cadena”, “índice” (que va ser igual a 0) y la “i” (que son los dígitos que se va leer el programa), crea así un índice numérico de las posiciones de la cadena, cuando termine en leer un número de la cadena, se le suma 1 al índice.

```

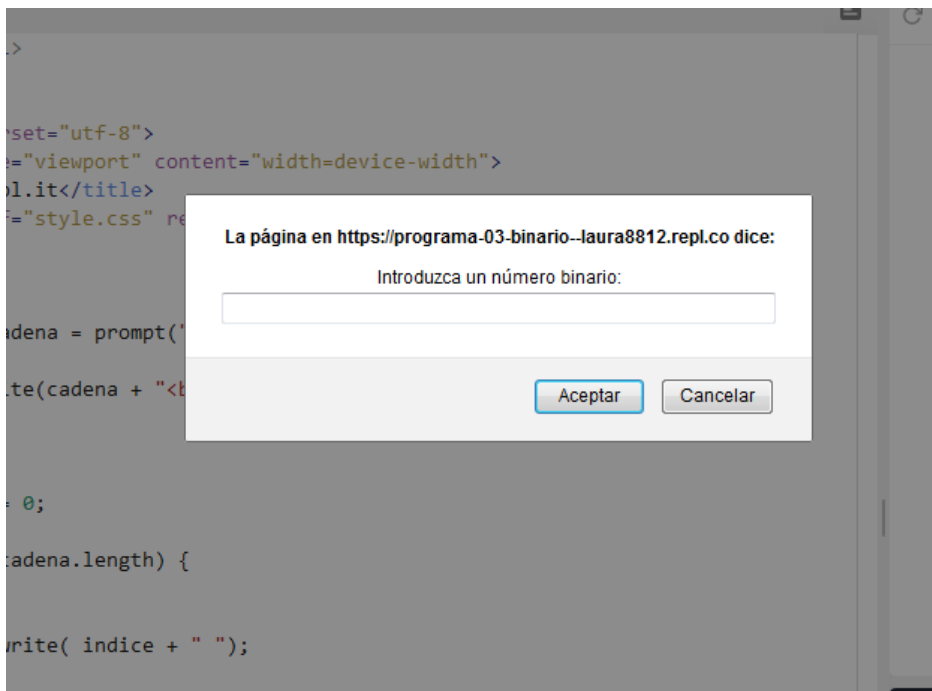
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <meta name="viewport" content="width=device-width">
6    <title>TALLER 3</title>
7    <link href="style.css" rel="stylesheet" type="text/css" />
8  </head>
9  <body>
10   <script>
11     var cadena = prompt("Introduzca un número binario:");
12
13     document.write(cadena + "<br><br>");
14
15
16
17     var indice = 0;
18     var i = 0;
19     while (i < cadena.length) {
20
21
22       document.write( indice + " ");
23
24       indice = indice + 1;
25
26       i = i + 1;
27     }
28
29     document.write("<br><br>¡FINALIZADO!");
30   </script>
31 </body>
32 </html>

```

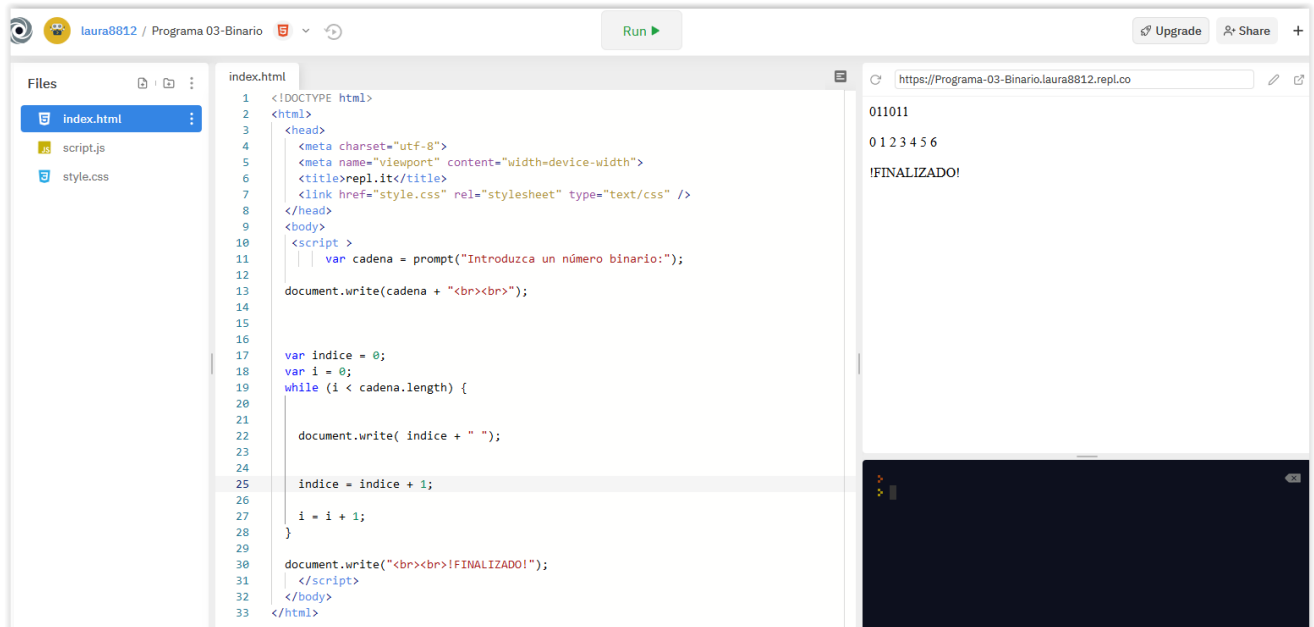
En este código utilizamos **"prompt"** para ingresar el número binario que queremos contar sus posiciones, entonces lo que se hace es crea un índice (que comienza desde cero), y por cada digitito que "1" o "0" que se encuentra en la cadena, se le agregara "1" en el índice, todo eso con **"while (i < cadena.lenght)"**, se analiza cada dígito para que vaya aumentando el índice (como si fuera un contador); para que se visualice el índice de cada dígito se coloca la función **"document.write"** y entre el paréntesis colocar el (índice + " ") , las comillas ayuda a generar espacio entre el índice y así poder organizar de una forma adecuado; se crea dos funciones que es el **"índice"** (el que se mostrara en la pantalla) y el **"i"** (los dígitos que está en la cadena), este se le suma 1 , para hacer el conteo y así repetir el proceso hasta que ya no haya más dígitos que leer, sin embargo no se verá ese proceso en la página (debido que simplemente lo que nos interesa en el ejercicio es el índice).

Para finalizar con **"document.write"** se visualizara en la página, la cadena que ingresamos en primer lugar, las posiciones de esta cadena se visualizara junto con la palabra "finalización" del programa.

Al ejecutar en repl.it, se obtiene el resultado mostrado seguidamente:



Se introduce el número binario



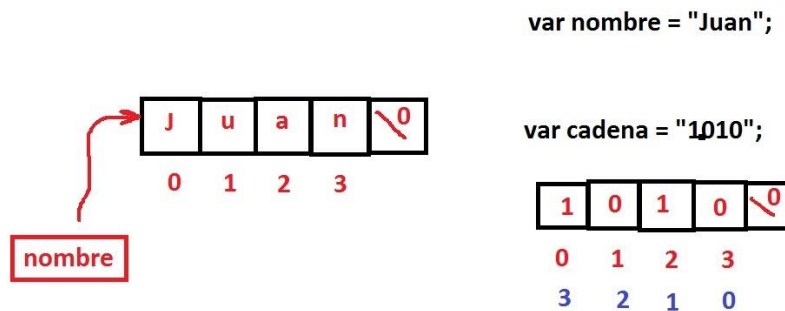
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width">
6   <title>repl.it</title>
7   <link href="style.css" rel="stylesheet" type="text/css" />
8 </head>
9 <body>
10  <script>
11    | var cadena = prompt("Introduzca un número binario:");
12
13    document.write(cadena + "<br><br>");
14
15
16
17    var indice = 0;
18    var i = 0;
19    while (i < cadena.length) {
20
21      document.write( indice + " ");
22
23
24      indice = indice + 1;
25
26      i = i + 1;
27    }
28
29    document.write("<br><br>¡FINALIZADO!");
30  </script>
31 </body>
32 </html>
```

011011
0 1 2 3 4 5 6
¡FINALIZADO!

Ahí se muestra la introducción del número binario, y el programa cuenta su posición comenzando desde el 0.

3.6 CONTAR LA POSICION DE LOS NÚMERO (COMENZANDO DESDE EL NÚMERO MENOR) DE UN NÚMERO BINARIO

Vamos a presentar el programa que cuenta el número de unos disponibles en un binario. Veamos la estructura de la cadena que contiene el binario:



El número de unos se calcula recorriendo la cadena de izquierda a derecha, Programa que lee en una cadena de texto un número binario, primero creando las variables que se va necesitar en el ejercicio, en ese caso es la **"cadena"**, **"indice"** (que va ser igual a cadena.length – 1, debido que vamos a contar los números desde el mayor hasta el 0) y la **"i"** (que son los dígitos que se va leer el programa que va ser igual a cero), crea así un índice numérico de las posiciones de la cadena, cuando termine en leer un número de la cadena, se le restar 1 al índice.

```

1
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width">
7     <title>TALLER 4</title>
8     <link href="style.css" rel="stylesheet" type="text/css" /
9   </head>
10  <body>
11    <script >
12      var cadena = prompt("Introduzca un número binario:");
13
14      document.write(cadena + "<br><br>");
15
16
17

```

```

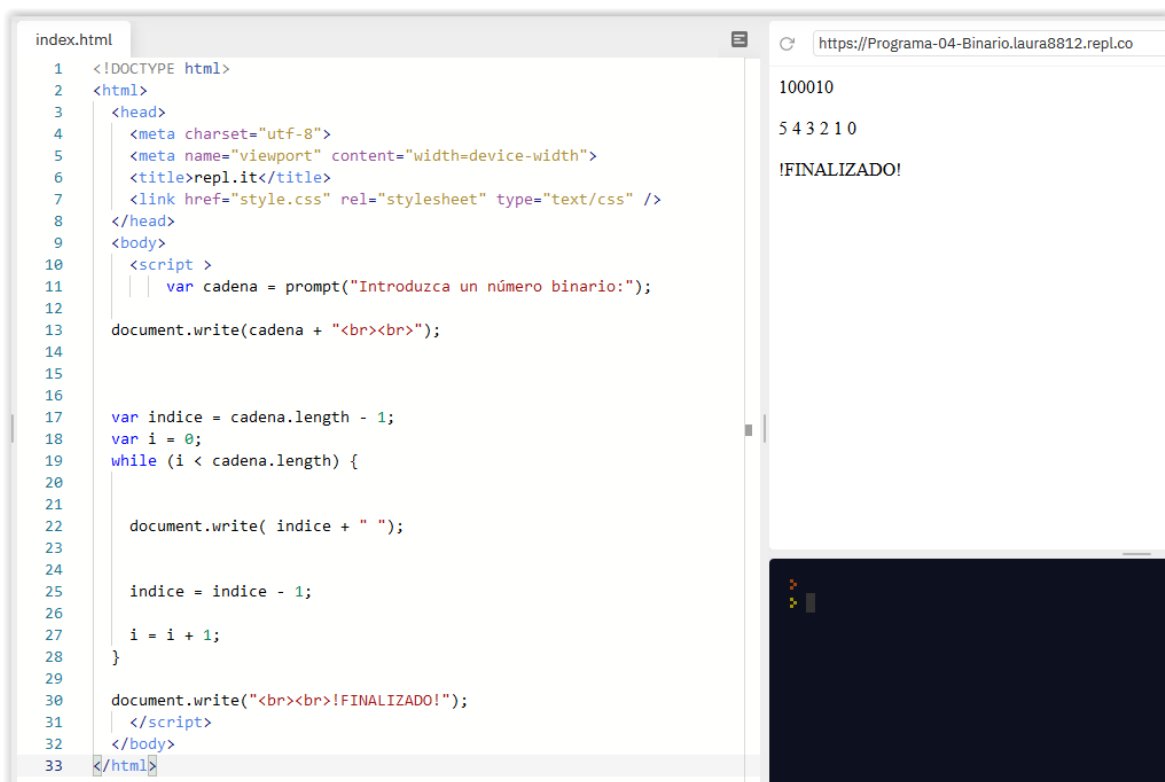
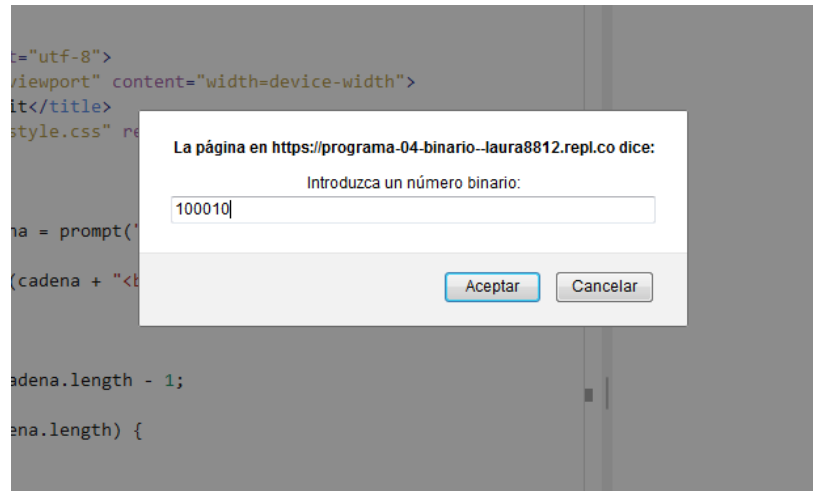
18  var indice = cadena.length - 1;
19  var i = 0;
20  while (i < cadena.length) {
21
22
23      document.write( indice + " ");
24
25
26      indice = indice - 1;
27
28      i = i + 1;
29  }
30
31  document.write("<br><br>!FINALIZADO!");
32  </script>
33  </body>
34  </html>

```

En este código utilizamos **"prompt"** para ingresar el número binario que queremos contar sus posiciones, entonces lo que se hace es crea un índice (que comienza desde cero), y por cada dígito que "1" o "0" que se encuentra en la cadena, se le restará "1" en el índice, todo eso con **"while (i < cadena.length)"**, se analiza cada dígito para que vaya disminuyendo el índice (como si fuera un contador); para que se visualice el índice de cada dígito se coloca la función **"document.write"** y entre el paréntesis colocar el (índice + " ") , las comillas ayuda a generar espacio entre el índice y así poder organizar de una forma adecuado; se crea dos funciones que es el **"índice"** (el que se mostrara en la pantalla, este va restando con el 1) y el **"i"** (los dígitos que está en la cadena), este se le suma 1 , para hacer el conteo y así repetir el proceso hasta que ya no haya más dígitos que leer, sin embargo no se verá ese proceso en la página (debido que simplemente lo que nos interesa en el ejercicio es el índice).

Para finalizar con **"document.write"** se visualizara en la página, la cadena que ingresamos en primer lugar, las posiciones de esta cadena se visualizara junto con la palabra **"finalización"** del programa.

Al ejecutar en repl.it, se obtiene el resultado mostrado seguidamente:



Lo curioso de ese código, es que es similar al anterior ejercicio, la única diferencia es que ya la variable índice no se suma , sino que se resta con el 1, para que así se podría ver los números de mayor a menor .

4 CONVERSIÓN EXTENDIDA

A continuación presentamos el programa de conversión extendida, la cual se encarga de dar tratamiento a los números en base 16.

En esta conversión el numerador se divide con el denominador, y determina su resto, hay que tener en cuenta en algo; es que los restos mayores de nueve deben ser visualizados por una letra.

A continuación se presentan las imágenes de los códigos requeridos, para implementar el proceso mostrado en JavaScript. Cada imagen presenta una función distinta, o la ejecución final del programa. Se debe escribir en un solo archivo el código mostrado:

```
<script>
// FUNCIONES DE APOYO
function texto( cadena, numero_saltos = 0 ) {
    document.write( cadena );
    var i = 0;
    while ( i < numero_saltos ) {
        document.write( "<br/> " );
        i = i + 1;
    }
}
```

En esta función se crea **“function texto”** que cada vez que la llamamos todo lo que estará dentro de esta se volverá una cadena que se visualizará en la página.

En el siguiente código se mostrarán dos partes: en la primera utilizando la recursividad y los condicionales y posibles resultados.

```
// PROGRAMA PRINCIPAL
function conversion ( numero, base = 2 ) {
  var control = 0;
  var division, resto;
  var resultado = "";
  var bandera = 0;
  while (bandera == 0) {
    division = Math.trunc( numero / base );
    resto = numero - Math.trunc( numero / base ) * base;

    if (base == 16 && resto > 9 ) {
      var aux = "";
      switch (resto) {
        case 10:
          aux = "A";
          break;
        case 11:
          aux = "B";
          break;
        case 12:
          aux = "C";
          break;
        case 13:
          aux = "D";
          break;
        case 14:
          aux = "E";
          break;
        case 15:
          aux = "F";
          break;
        default:
          aux = "";
          break;
      }
      resultado = aux + resultado + " ";
    }
    else {
      resultado = resto.toString() + resultado + " ";
    }
  }
}
```

Lo primero que hay que tener en cuenta son establecer las variables, ya definidas podemos colocarles la operación que vamos a bordar, creando así la **“function conversión”**, que es la división y de esa división sacarle el resto (que es el resultado de ese ejercicio), dependiendo de lo que dé en este resto se define la condición if, si el “resto” di un número mayor de 9, se coloca el case, su número y su determinada letra, por ejemplo si el resto dio “10” la letra que debe mostrar es “A”; así se haría hasta el case “15” (No se hace más números debido que la base que estamos utilizando es 16 y es imposible que de un número mayor a ese).

Lo siguiente que se establece es el resultado, se hace dos versiones: la primera lo que pasaría si el “resto” dio un número mayor a 9, entonces ahí se define que su resultado debe mostrar la letra (“aux” determinado en el programa), el resultado ya después de pasar por el if (utilizando la recursividad del if) y una comillas “” (para separar espacio. La segunda versión

es lo que pasaría si el número es menor que “9”, simplemente se mostraría en el resultado lo que dio en el resto.

```

    }
    resultado = aux + resultado + " ";
  }
  else {
    resultado = resto.toString() + resultado + " ";
  }

  numero = division;

  if ( division < 1 )
    bandera = 1;

  control = control + 1;
  if (control > 1000) {
    texto( "ERROR. Se superó el número de 1000 iteraciones", 1 );
    bandera = 1;
  }
}

return resultado;
}

```

Por último ahí para determinar si la división es dio un número menor a “1”, en ese caso se determinó que el resultado de la división (no resto) es uno; adicionalmente se creó el “control” para vigilar que no superar los 1000 iteraciones, ya que de no ser así el programa haría muchos ciclos y se demoraría mucho en ejecutarlo. Al final se dio la función “return” para retomar el resultado (en caso si resto dio mayor a 9, sino no se retoma).

```

// EJECUCIÓN DEL PROGRAMA
var n = 175;
var b = 16;

var resp = conversion( n, b );

texto( "Número: " + n, 1);
texto( "Conversión a base: " + b, 1);
texto( "Resultado: " + resp, 2);

// PROGRAMA TERMINADO
texto("PROGRAMA FINALIZADO!", 1 );
texto("Octubre 22 de 2020");
</script>

```

Para finalizar, se estable las variables “n” y “b” (numerador y base respectivamente), también la **“function resp”** que se encarga de llamar la función “conversión para poder dividir n y b, después se estable como se mostrar los resultados en la página, llamando la función “texto” se ejecuta el programa y se convierte en cadenas, por último, se escribir la finalización del programa junto con la fecha de su realización.

A continuación, se muestra el programa en el entorno repl.it, con los datos de ejecución del programa.

```

1 // FUNCIONES DE APOYO
2 function texto( cadena, numero_saltos = 0 ) {
3   document.write( cadena );
4   var i = 0;
5   while ( i < numero_saltos ) {
6     document.write( "<br/> " );
7     i = i + 1;
8   }
9 }
10
11 // PROGRAMA PRINCIPAL
12 function conversion ( numero, base = 2 ) {
13   var control = 0;
14   var division, resto;
15   var resultado = "";
16   var bandera = 0;
17   while (bandera == 0) {
18     division = Math.trunc( numero / base );
19     resto = numero - Math.trunc( numero / base ) * base;
20
21     if (base == 16 && resto > 9 ) {
22       var aux = "";
23       switch (resto) {
24         case 10:
25           aux = "A";
26           break;
27         case 11:
28           aux = "B";
29           break;
30         case 12:
31           aux = "C";
32           break;
33         case 13:
34           aux = "D";
35           break;
36         case 14:
37           aux = "E";
38           break;
39         case 15:
40           aux = "F";
41           break;
42       }
43       resultado = aux + resultado;
44     }
45     numero = division;
46     control++;
47     if (control == 10) {
48       bandera = 1;
49     }
50   }
51   texto( resultado, 10 );
52 }
53
54 // Ejecución
55 conversion( 175 );
56
57 // Finalización
58 console.log( "PROGRAMA FINALIZADO!" );
59 console.log( "Octubre 22 de 2020" );

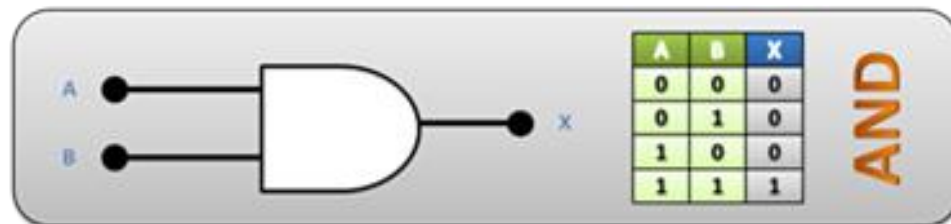
```

Número: 175
 Conversión a base: 16
 Resultado: AF
 PROGRAMA FINALIZADO!
 Octubre 22 de 2020

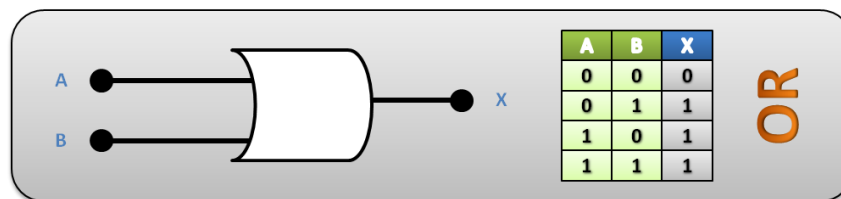
5 COMPUERTAS AND, OR, NOT

Las puertas lógicas son los componentes básicos de cualquier sistema digital. Es un circuito electrónico que tiene una o más de una entrada y solo una salida. La relación entre la entrada y la salida se basa en una cierta lógica. En base a esto, las puertas lógicas se denominan AND, OR, y NOT, dependiendo de cada situación.

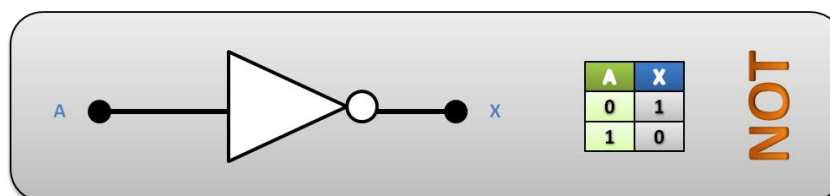
- **AND:** Condición donde si o si tiene que cumplir con los dos requisitos. Por ejemplo: La puerta AND acepta dos cables, y si ambos cables están "encendidos" (1, 1), genera 1. Si alguno de esos cables está "apagado" (que representa 0,1), entonces genera 0.



- **OR:** Condición donde no es necesario que se cumpla dos o más requisitos. La puerta lógica OR acepta dos entradas, y siempre que cualquiera de esas entradas sea un (1,0), genera un 1.

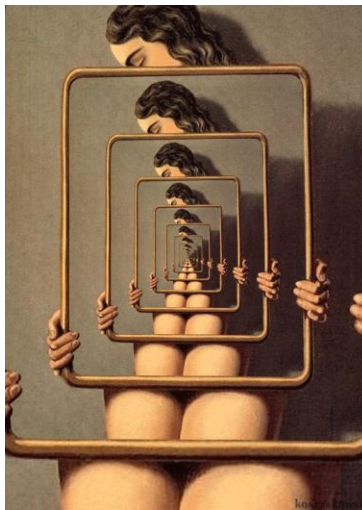


- **NOT:** Son dispositivos de entrada única que tienen un nivel de salida que normalmente está en el nivel lógico "1" y va "BAJO" a un nivel lógico "0" cuando su entrada única está en el nivel lógico "1", es decir: Condición que cambiar su contrario

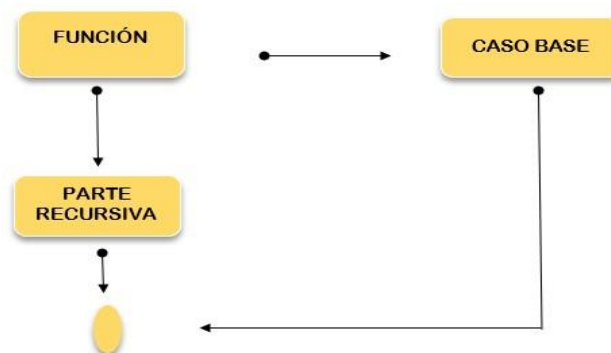


6 RECURSIVIDAD

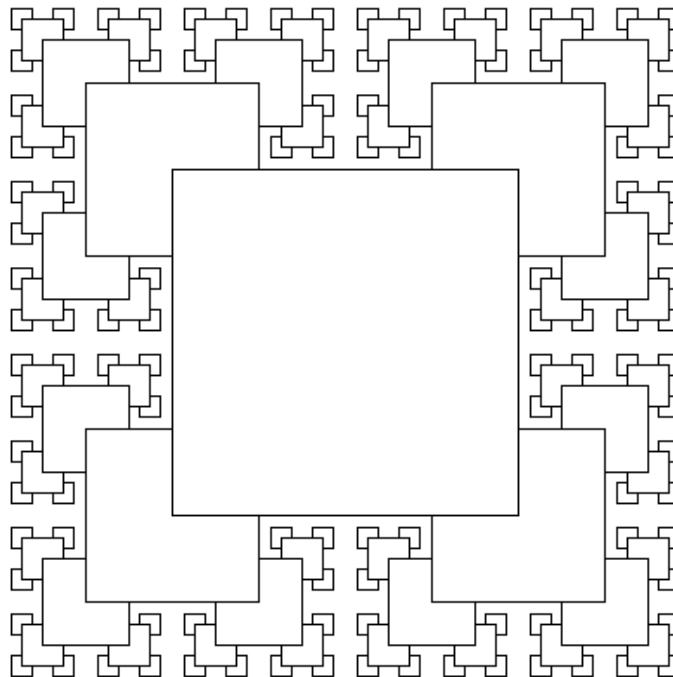
Podemos definir la recursividad como una técnica utilizada en programación que nos permite que cierta cantidad de instrucciones se ejecute un cierto número de veces, llamándose a sí mismas durante su propia ejecución. Cuando un procedimiento incluye una llamada a sí mismo se conoce como recursión directa y cuando un procedimiento llama a otro procedimiento y éste causa que el procedimiento original sea invocado, se conoce como recursión indirecta. Esto lo podemos ver en el siguiente gif, ejemplificando de forma idónea lo mencionado anteriormente en un aspecto más general al repetirse una y otra vez la misma imagen contribuyendo al principio de recursividad.



Estas funcionan de forma similar a las iteraciones, pero debe encargarse de planificar el momento en que dejan de llamarse a sí mismas o tendrá una función recursiva infinita, teniendo el caso base y su parte recursiva.



Como parte de este proceso, podemos clasificar los diferentes pasos que ejecuta la función para llevar a cabo esta tarea de forma eficiente. Primero, cuando la función comprueba que el dato x es diferente al caso base; El procedimiento se ejecuta y se llama a sí mismo. Después, el problema se resuelve, llevando dentro de él todos los datos almacenados hasta ese momento y, por último, el proceso en el que estos datos cambian, evita la aparición de un ciclo infinito, encontrando de esta forma el caso base, dando fin a su continua ejecución.



Para entender todo lo mencionado anteriormente, se ejemplificará la recursividad por medio de la función Fibonacci, siendo esta base y una de las funciones recursivas más conocidas en toda la programación. La sucesión o serie de Fibonacci hace referencia a la secuencia ordenada de números descrita por Leonardo de Pisa, matemático italiano del siglo XIII:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,...

El código fuente de esta función en JavaScript, es el siguiente:

```
function fib ( n ) {
  if ( n>1) { ///el punto de ruptura es 1,porque en esta sucesión
    solamente calcula los números mayores de 1.
    return fib ( n - 1 ) + fib ( n - 2 );//Es la suma de las dos numeros
    anteriores al actual de la sucesión, cada vez que se suma, se
    disminuye -1 al lado izquierdo y se disminuye -2 al lado derecho;para
    que así poder dar esa función, este al final da dos resultados o
    casos , si da 0 o 1
  }
  else if (n==1){
    return 1;///si es uno nos devolverá 1 y el número que nos dio en la
    suma
  }
  else if (n==0){
    return 0;///si es cero nos devolverá 0 y el número que nos dio en la
    suma
  }
}

var n = 8;

respuesta = fib ( n );

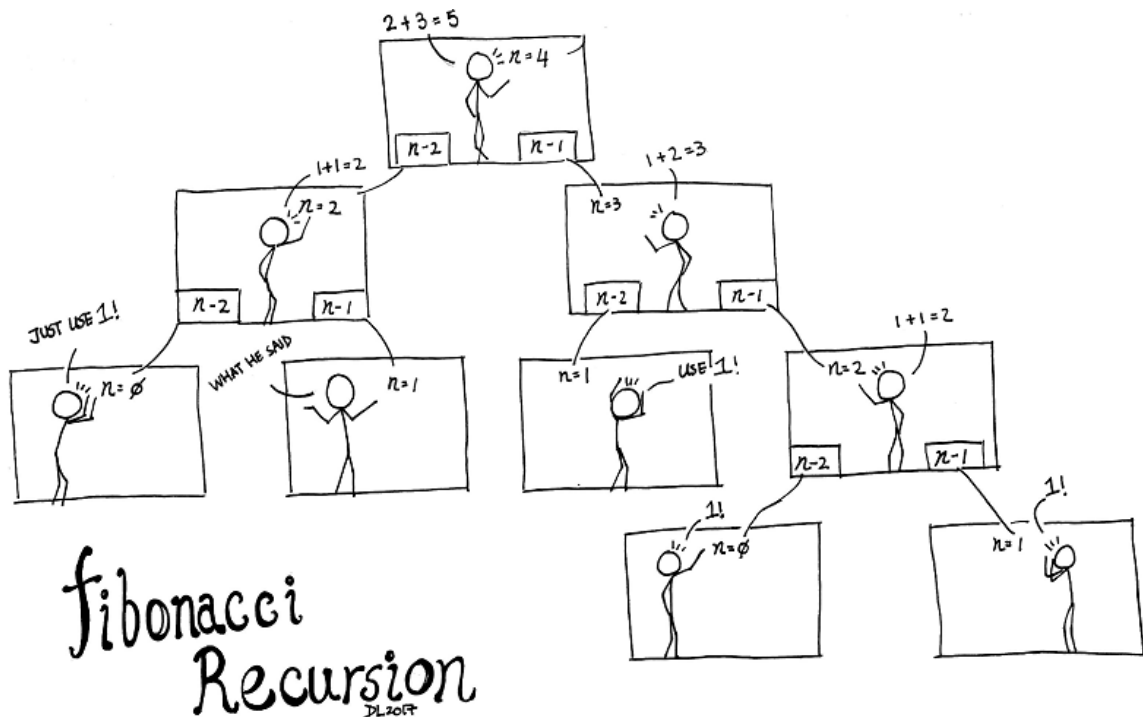
texto ( "SUCESIÓN DE FIBONACCI DE UN NÚMERO" );
texto ( n + " = " + respuesta ); salto();///se llama n, con respuesta
para que se visualice en el programa
```

Analizando el código anterior, podemos ver cada uno de las partes y el perfecto funcionamiento de la recursividad. Podemos observar claramente que el dato base en este caso es el numero 1 ó 0 y, el dato al ser diferente a estos dos, ejecuta la función recursiva hasta llegar al dato base de la siguiente manera:

$$\text{Fib}(n) = \begin{cases} 0 & \text{si } N = 0 \text{ (base)} \\ 1 & \text{si } N = 1 \text{ (base)} \\ \text{fib}(n-1) + \text{fib}(n-2); & \text{si } N > 1 \text{ (recursión)} \end{cases}$$

Es evidente que este razonamiento recursivo se basa en las dos partes mencionadas anteriormente: El caso base y la parte recursiva. El caso base impide que se ejecute el proceso recursivo y es el punto tanto de partida como de terminación de la función.

Ya enfocándonos en el proceso que requiere esta función, cuando el dato x es diferente al caso base, lo que hace la función es sumar los dos números anteriores al actual de la sucesión, cada vez que se suma, se disminuye -1 al lado izquierdo y se disminuye -2 al lado derecho; para que así poder dar esa función, este al final da dos resultados o casos, si da 0 o 1.



Es importante usar estos procedimientos recursivos, ya que generalmente son más fácil de analizar, son más adecuados para estructuras de datos recursivas y estos algoritmos recursivos proporcionan soluciones estructuradas, sencillas y elegantes. Además, es importante para simplificar el código. Aunque, debemos tener cuidado de caer en un ciclo infinito y evitar usar este razonamiento cuando los métodos usen arreglos largos, cuando el método cambia de manera impredecible de campos y cuando las iteraciones sean la mejor opción.

7 CONCLUSIONES

El desarrollo de las temáticas elaboradas en clase utilizando el lenguaje JavaScript prueba ser un mecanismo de gran valor para el aprendizaje de los conceptos básicos de la materia.

Todos los conceptos mencionados anteriormente son base de la programación, por lo cual es necesario tenerlos lo más claro posible para evitar confusiones al incrementar el nivel en la programación.

Podemos ver muchos de estos términos aplicados en la vida cotidiana al explorar y entenderlos un poco más.

8 BIBLIOGRAFÍA

<https://javautodidacta.es/metodo-tostring-java/>

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/parseInt#:~:text=parseInt%20es%20una%20funci%C3%B3n%20de,16%20hexadecimal%2C%20y%20as%C3%AD%20sucesivamente.

<https://www.logicbus.com.mx/compuertas-logicas.php>

<http://service.udes.edu.co/modulos/documentos/pedropatino/compuertas.pdf>

<https://quantdare.com/numeros-de-fibonacci/>

<https://sites.google.com/site/portafoliosenati/fundamentos-de-programacion/13-aplicar-recursividad>

<https://repl.it>