FUNDAMENTOS DE PROGRAMACIÓN ESTRUCTURADA

Estructura de contenidos

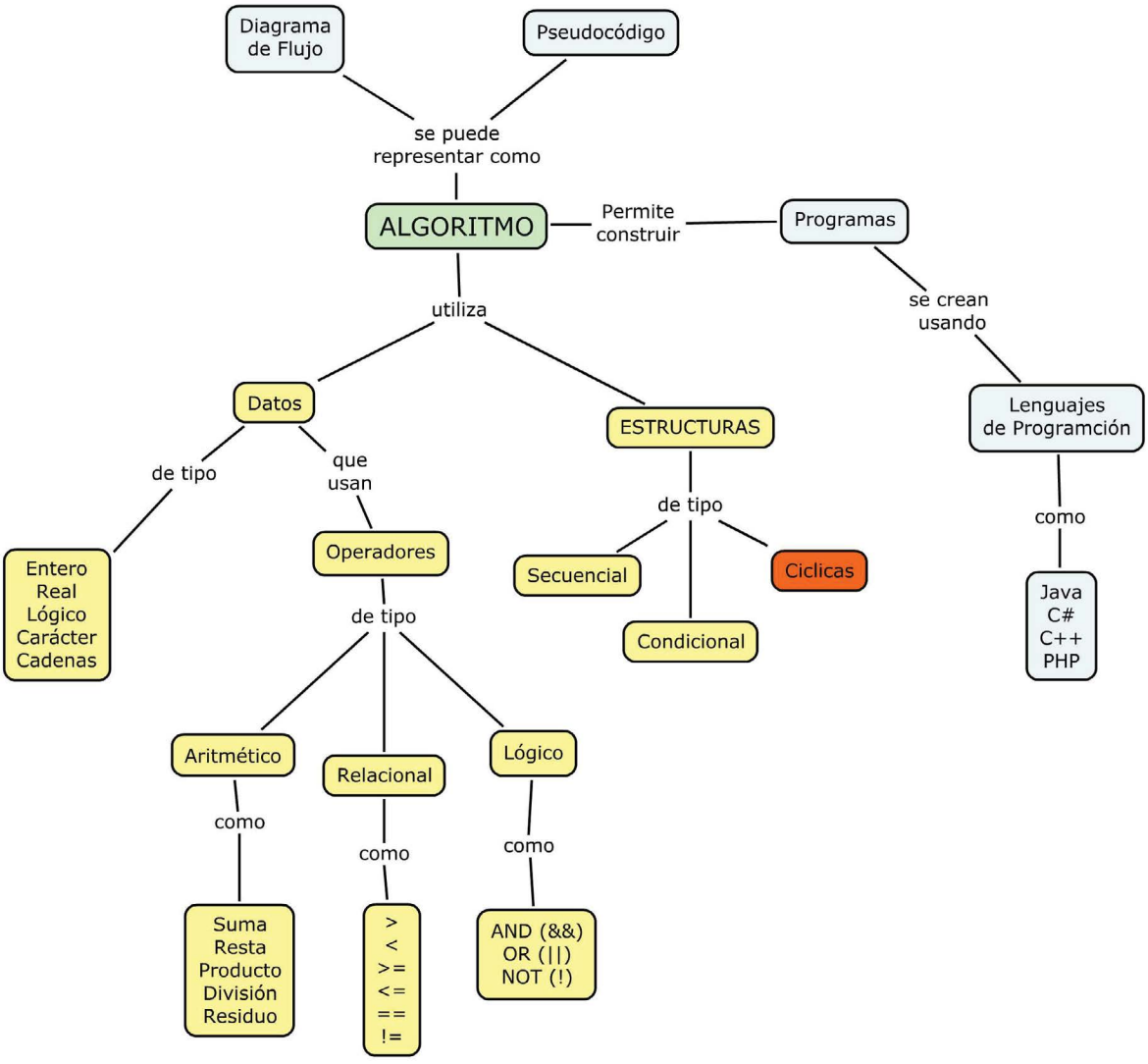
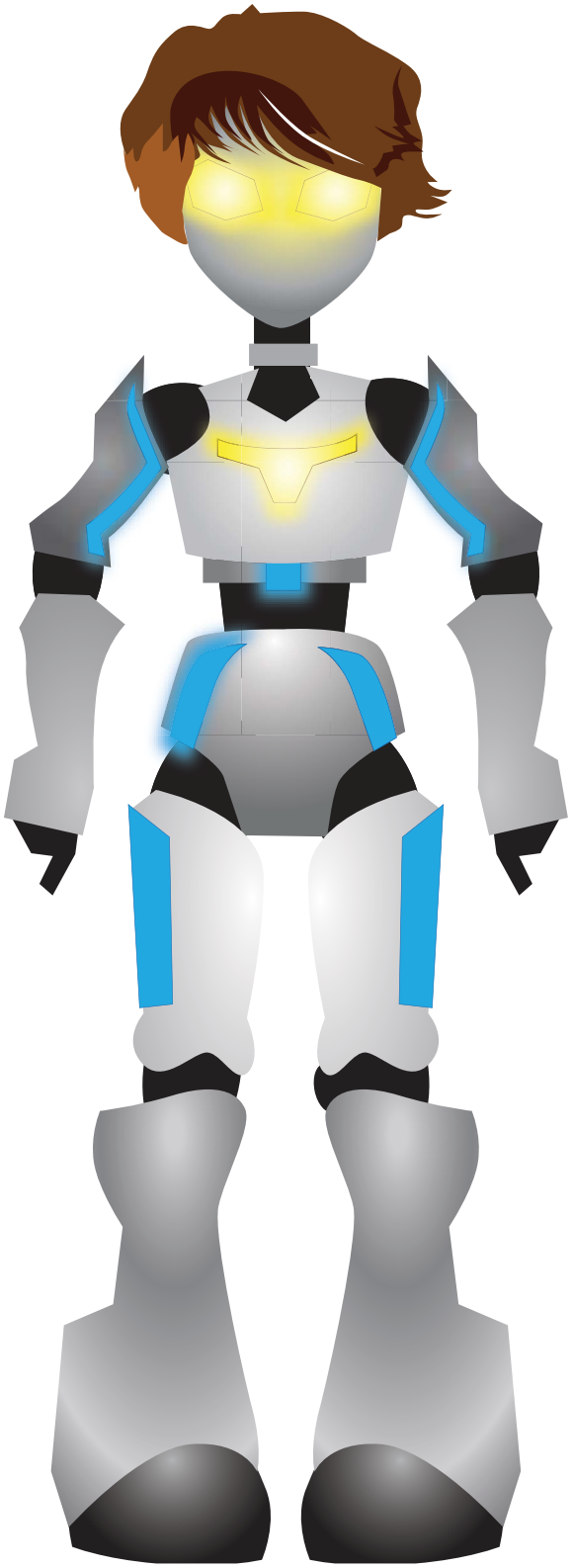


Diagrama de Flujo
Pseudocódigo
se puede representar como
ALGORITMO
Permite construir
Programas
utiliza
se crean usando
Datos
ESTRUCTURAS
Lenguajes de Programción
de tipo
que usan
de tipo
como
Entero
Real
Lógico
Carácter
Cadenas
Operadores
Secuencial
Ciclicas
Condicional
Java
C#
C++
PHP
de tipo
Aritmético
Relacional
Lógico
como
como
como
Suma
Resta
Producto
División
Residuo
>
<
>=
<=
==
!=
AND (&&)
OR (||)
NOT (!)

INTRODUCCIÓN



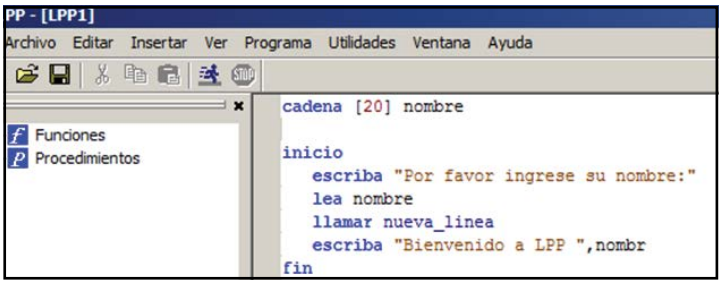
Para un Analista y Desarrollador de Sistemas de Información es primordial adquirir una gran destreza en el desarrollo de soluciones algorítmicas, ya que estas se convertirán posteriormente en programas de computador capaces de automatizar las tareas cotidianas de una organización, empresa o individuo. Los fundamentos de programación pueden ser comparados con los fundamentos para conducir un vehículo, pues estos fundamentos son aplicables a cualquier tipo de vehículo sin importar su marca o cilindraje. De la misma manera, los fundamentos de programación son aplicados en cualquier lenguaje de programación.

La herramienta LPP (Lenguaje de Programación para Principiantes), permite adquirir los fundamentos de programación necesarios para construir soluciones de software, familiarizando al programador con tareas rutinarias como la declaración de variables, el uso de estructuras de control de flujo, arreglos, subrutinas y muchas otras actividades que forman parte del día a día de un desarrollador de sistemas de información.

1. MI PRIMER PROGRAMA.

1.1 La Codificación.

La Codificación consiste en pasar el algoritmo al lenguaje de programación seleccionado, en este caso el lenguaje es LPP. Nuestro primer programa consiste en leer el nombre de una persona y presentar un mensaje personalizado.



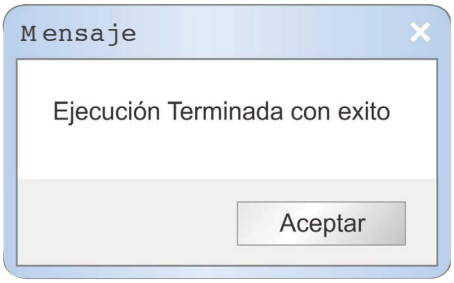
Línea	Significado
Cadena [20] nombre	Antes del inicio, se deben declarar todas las variables que se van a emplear en el programa
Inicio	Marca el inicio del programa
escriba "Por favor ingrese su nombre"	Presenta un mensaje en la pantalla
lea nombre	Captura información por parte del usuario
llamar nueva_linea	Permite pasara la siguiente línea de la pantalla del usuario
escribe "Bienvenido a Lpp"	Presenta mensaje combinando parte textual con parte variable
Fin	Marca el final del programa

1.2 La Compilación.

La Compilación permite detectar los errores sintácticos (sintaxis: conjunto de normas que regulan la codificación de un programa), también conocidos como errores de compilación. Para compilar un programa en LPP, seleccionamos del menú Programa la opción Compilar.

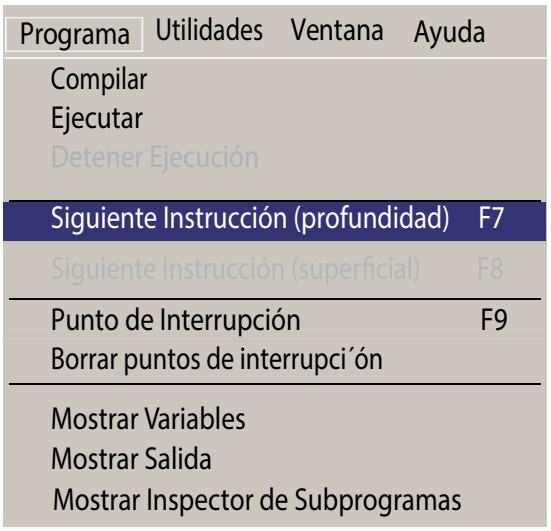
```
cadena [ 20 ] nombre

inicio
    escriba "Por Favor ingrese su nombre:"
    lea nombre
    llamar nueva_linea
    escriba "Bienvenido a LPP" , nombr
fin
```

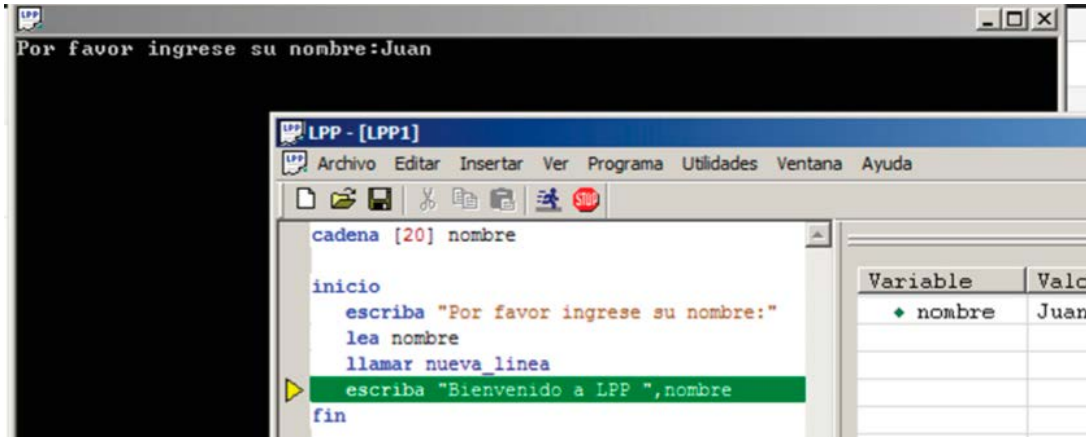


1.3 La Compilación.

La Depuración permite hacer el seguimiento paso a paso de un programa.
Con la depuración es posible pasar de instrucción en instrucción e ir observando el comportamiento que va teniendo el programa y los valores que van tomando las variables. De esta manera el programador puede encontrar tanto errores de sintaxis como errores de lógica.

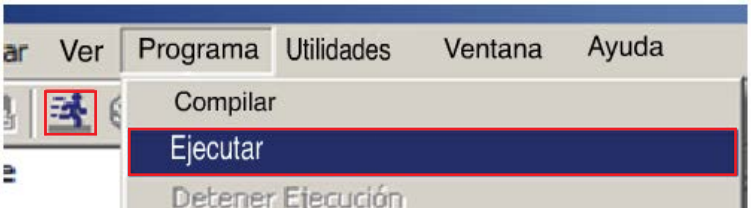


Para depurar un programa en LPP, se cuenta con una serie de opciones desde el menú Programa.



1.4 La Ejecución.

La Ejecución del programa permite observar su comportamiento de la manera como lo percibirá el usuario final. Para ejecutar un programa en LPP, se selecciona del menú Programa la opción Ejecutar, o se hace click en el botón correr de la barra de herramientas.

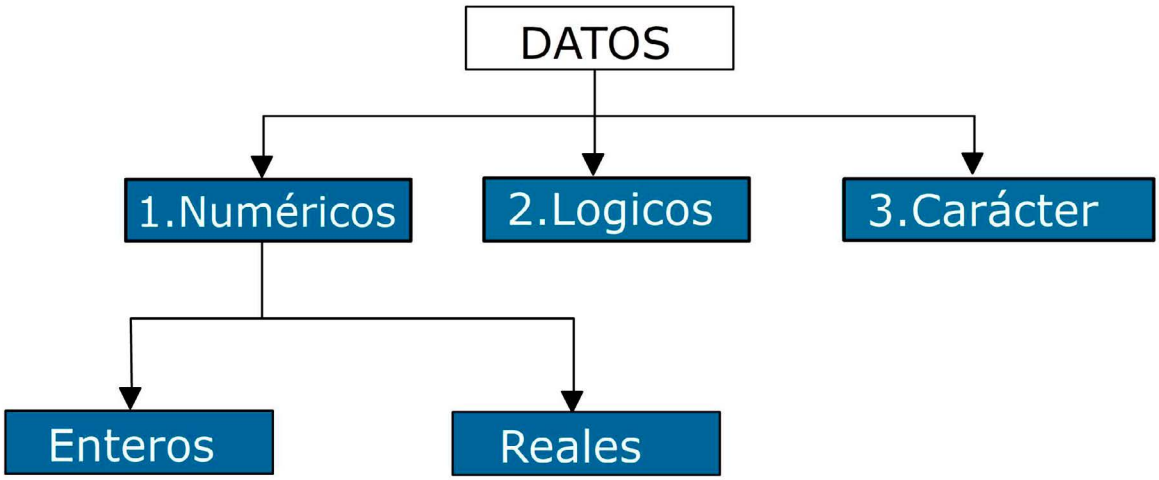


2. TIPOS DE DATOS.

Al desarrollar un programa de computador, el programador debe trabajar con diferentes datos, por ejemplo, si el programa necesita mostrar la información de un estudiante, seguramente requerirá de datos como su nombre, edad, género, nota, está Matriculado, entre otros. Cada uno de estos datos son de una naturaleza diferente.

Tipo de Dato	Nombre en LPP	Comentarios
Entero	Entero	Números sin decimales
Real	Real	Números que pueden tener decimales
Lógico	Booleano	Solo recibe valores de **falso** o **verdadero**
Caracter	Caracter	Solo recibe un único caracter, que puede ser una letra, un número o un signo
	Cadena	Puede recibir un conjunto de caracteres

Nombre= Juan Perez
Edad= 19
Género= M
Nota= 4.5
estaMatriculado= Verdadero



PROGRAMA DE EJEMPLO

Como programadores una de las tareas cotidianas es definir las variables requeridas en un programa y asignarle correctamente su respectivo tipo de dato. Esto hará que el programa funcione de manera eficiente.

A

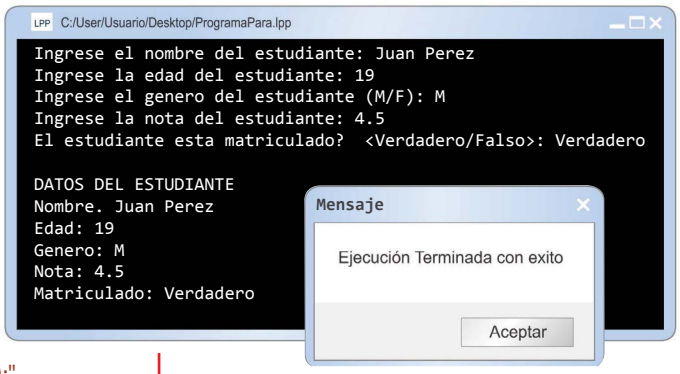
```
Cadena [30] nombre
Entero edad
Caracter genero
Real nota
Booleano estaMatriculado
```

B

```
Inicio
  escriba "Ingrese el nombre del estudiante:"
  lea nombre
  escriba "Ingrese la edad del estudiante:"
  lea edad
  escriba "Ingrese el genero del estudiante (M/F):"
  lea genero
  escriba "Ingrese la nota del estudiante:"
  lea nota
  escriba "El estudiante esta matriculado? (Verdadero/Falso):"
  lea estaMatriculado
  llamar nueva_linea
  escriba "DATOS DEL ESTUDIANTE"
  llamar nueva_linea
```

C

```
  escriba "Nombre: ",nombre
  llamar nueva_linea
  escriba "Edad: ",edad
  llamar nueva_linea
  escriba "Género: ",genero
  llamar nueva_linea
  escriba "Nota: ",nota
  llamar nueva_linea
  escriba "Matriculado: ",estaMatriculado
Fin
```



Sección	Comentario
A	Declaración de cada una de las variables empleadas en el programa con su respectivo tipo de dato.
B	Declaración de cada una de las variables empleadas en el programa con su respectivo tipo de dato.
C	Declaración de cada una de las variables empleadas en el programa con su respectivo tipo de dato.

3. OPERADORES Y EXPRESIONES ARITMÉTICAS.

3.1 Los Operadores Aritméticos.

La mayoría de los programas de computador requieren realizar cálculos u operaciones que involucran operadores aritméticos, por esta razón, como programadores es necesario conocer cada uno de ellos y la manera como el computador los interpreta para calcular los resultados de una determinada expresión o fórmula.

¿sabes cuál es la resultado de la siguiente extresión?

3 + 5 x 2 = _____?

Explicación

5 x 2 = 10

3 + 10 = 13

la respuesta correcta es 13, pues bien, además de conocer los diferentes operadores aritméticos, también es importantísimo conocer los niveles de prioridad de cada uno de ellos.

En el caso de la expresión 3 + 5 x 2, primero se realiza la multiplicación 5 x 2 cuyo resultado es 10 y posteriormente se realiza la operación 3 + 10, dando como resultado final 13

3.2 Reglas De Prioridad En Los Operadores Aritméticos.

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

En LPP el signo igual (=) , se representa mediante una flecha dirigida hacia la variable que recibe el valor, esta flecha está conformada por los caracteres menor que (<) y menos (-) así: <-

Operadores Aritméticos y su Prioridad

Prioridad	Operador	Significado	Ejemplo
1	^	Exponenciación	4 ^ 2 = 16 3 ^ 3 = 27
2	*	Multiplicación	2 * 4 = 8 7 * 5 = 35
	/	División	5 / 2 = 2.5 6 / 3 = 2
3	DIV	División Entera	5 DIV 2 = 2 7 DIV 4 = 1
	MOD	Residuo de la División	5 MOD 2 = 1 8 MOD 4 = 0
4	+	Suma	3 + 4 = 7 2 + 9 = 11
	-	Resta	8 – 5 = 3 7 – 6 = 1

Por ejemplo, para representar la siguiente expresión:
X = 3 + 5 En LPP sería: X <- 3 + 5

Programa De Ejemplo

Ahora, después de conocer los operadores aritméticos y sus reglas de prioridad, puedes encontrar el resultado de la siguiente expresión:

veamos

2 + 2 ^ 2 * 2 + 2 MOD 2 = _____?

2 + 2 ^ 2 * 2 + 2 MOD 2 = _____?
2 + 4 * 2 + 2 MOD 2 = _____?
2 + 8 + 2 MOD 2 = _____?
2 + 8 + 0 = 10

3.3 Uso de los paréntesis en las expresiones aritméticas:

Cuando el programador 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. Retomando el ejemplo de la expresión:

- 3 + 5 x 2 = 13 pero (3 + 5) x 2 = 16
- 5 x 2 = 10 (3 + 5) = 8
- 3 + 10 = 13 8 x 2 = 16

3.4 Uso De Los Paréntesis En Las Expresiones Aritméticas:

Programa No. 3 Manejando Expresiones: A un programador le solicitan realizar una aplicación que calcule la nota promedio de un alumno a partir de las 2 notas que tiene en una asignatura.

Durante el análisis, el programador toma un caso de prueba para descubrir cuál es el procedimiento que debe llevar a cabo. En el caso de prueba toma como la primera nota el valor de 4 y como segunda nota el valor de

Nota 1	Nota 2	NotaPromedio
4	3	3.5

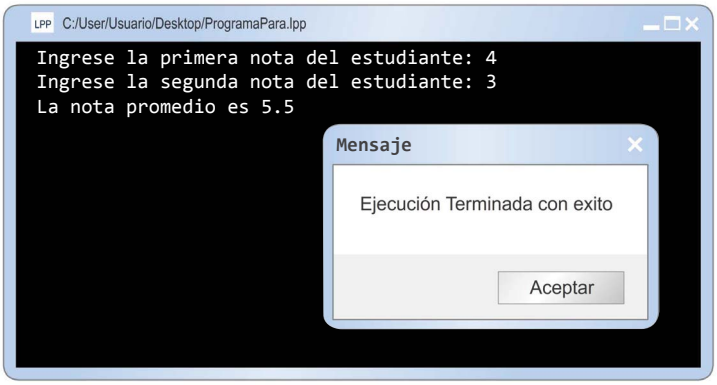
Durante el análisis, el programador identifica que debe sumar las dos notas y el resultado lo debe dividir entre dos: 4 + 3 = 7 luego 7 / 2 = 3.5

A partir de este análisis, el programador desarrolla la siguiente aplicación en LPP:

```
Real nota1, nota2, notaPromedio

Inicio
  escriba "Ingrese la primera nota del estudiante:"
  lea nota1
  escriba "Ingrese la segunda nota del estudiante:"
  lea nota2
  notaPromedio <-  nota1 + nota2  / 2
  escriba "La nota promedio es ",notaPromedio
Fin
```

Al ejecutar la aplicación ingresando los datos de prueba, el programador obtiene el siguiente resultado:



Después de buscar el error, se da cuenta que este se encuentra en la siguiente línea:

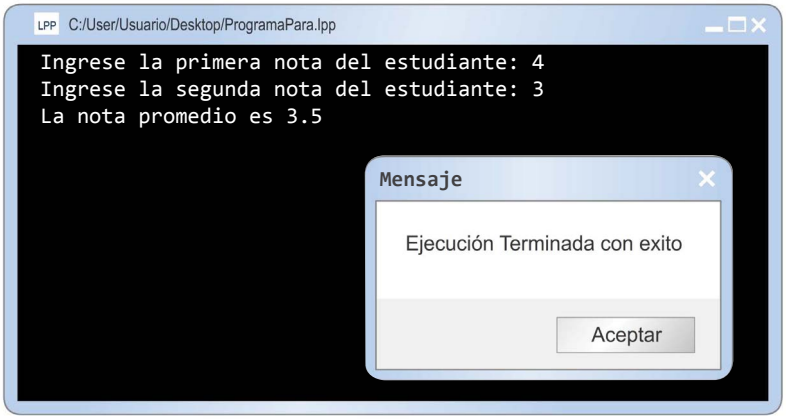
```
Real nota1, nota2, notaPromedio

Inicio
  escriba "Ingrese la primera nota del estudiante:"
  lea nota1
  escriba "Ingrese la segunda nota del estudiante:"
  lea nota2
  notaPromedio <- (nota1 + nota2) / 2
  escriba "La nota promedio es ",notaPromedio
Fin
```

Recuerda las reglas de prioridad y concluye que la primera operación que se está ejecutando es la división, por lo tanto,

El programador realiza un ajuste al programa para definir el orden deseado de ejecución de los operadores aritméticos mediante el uso de paréntesis. 3 / 2 = 1.5 y 4 + 1.5 = 5.5

Al ejecutar nuevamente la aplicación, obtiene el resultado esperado:



4. OPERADORES RELACIONALES Y LÓGICOS:

4.1 Los Operadores Relacionales.

Los operadores relacionales y lógicos son empleados para definir condiciones dentro de un programa. El resultado de una expresión que contiene estos operadores es un resultado de tipo lógico, es decir, solo puede ser **FALSO o VERDADERO.**

Operador	Significado	Ejemplo
>	Mayor que	3 > 4 FALSO 8 > 5 VERDADERO
<	Menor que	4 < 6 VERDADERO 7 < 4 FALSO
>=	Mayor o igual que	3 >= 3 VERDADERO 4 >= 4 FALSO
<=	Menor o igual que	2 <= 2 VERDADERO 3 <= 2 FALSO
=	Igual que	4 = 4 VERDADERO 3 = 4 FALSO
<>	Diferente que	6 <> 7 VERDADERO 7 <> 7 FALSO

4.2 Los Operadores Lógicos.

Los operadores lógicos son empleados para concatenar dos o más expresiones con operadores relacionales. Por ejemplo, la expresión:

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

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

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

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

- El operador lógico "Y" solo da como resultado Verdadero si ambas expresiones son verdaderas.

Operador "Y"			
Expresión 1	**Operador**	**Expresión 2**	**Resultado**
FALSO	**Y**	FALSO	**FALSO**
FALSO	**Y**	VERDADERO	**FALSO**
VERDADERO	**Y**	FALSO	**FALSO**
VERDADERO	**Y**	VERDADERO	**VERDADERO**

- El operador "O" da como resultado Verdadero cuando al menos una de las expresiones sea verdadera.

Operador "O"			
Expresión 1	**Operador**	**Expresión 2**	**Resultado**
FALSO	**O**	FALSO	**FALSO**
FALSO	**O**	VERDADERO	**VERDADERO**
VERDADERO	**O**	FALSO	**VERDADERO**
VERDADERO	**O**	VERDADERO	**VERDADERO**

4.3. Expresiones Con Operadores Relacionales Y Lógicos:

Una tarea habitual y muy importante a la hora de desarrollar programas de computador consiste en definir condiciones dentro del programa. Para que estas condiciones respondan exactamente a lo que el cliente de la aplicación necesita, se deben crear correctamente las expresiones que usen los operadores relacionales y lógicos.

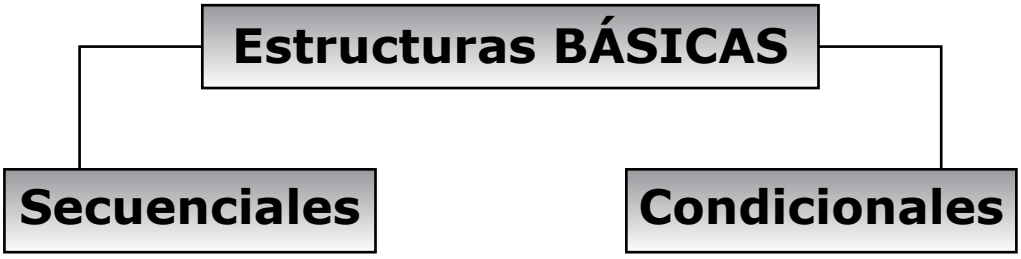
Para determinar los estudiantes menores de edad cuya nota es 4 o superior y que pertenecen a un estrato inferior a 3, la expresión

(edad < 18)
(nota >= 4)
(estrato < 3)

(modelo < 2010)

Por ejemplo, para conocer los vehículos cuyo modelo sea inferior al año 2010, la expresión sería:

5. ESTRUCTURAS BASICAS DE PROGRAMACIÓN.



5.1 Los Operadores Relacionales.

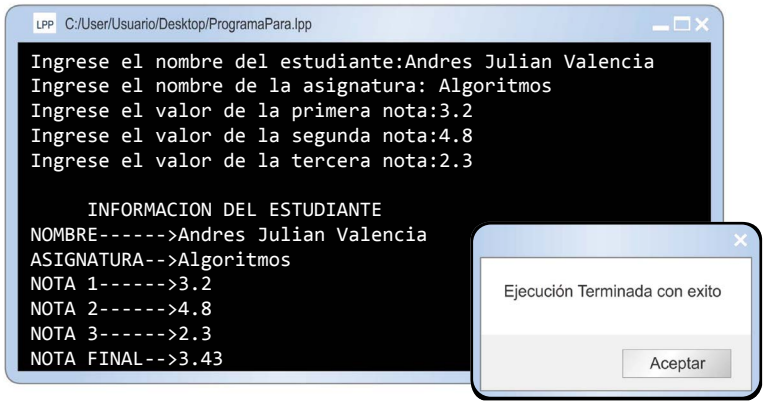
La estructura secuencial está conformada por instrucciones que se ejecutan consecutivamente una después de la otra. En una aplicación, después de ejecutar una instrucción secuencial, el programa siempre continúa con la siguiente instrucción.

```
Inicia Robot
        Avanzar
        Avanzar
        Avanzar
        Avanzar
        GirarIzquierda
        Avanzar
        Avanzar
        GirarDerecha
        Avanzar
        Avanzar
        Avanzar
        Detener
Termina Robot
```

Instrucción	Significado	Ejemplo
Inicio	Determina el comienzo del programa	Inicio
Fin	Determina el final del programa	Fin
Escriba	Muestra un mensaje en pantalla	Escriba “Bienvenido”
Lea	Almacena un dato suministrado por el usuario en una variable del programa	Lea nombre
Asignación	Permite asignarle a una variable un valor o el resultado de una expresión	X <- 2 area<- (b*h)/2

PROGRAMA SECUENCIAL:

Se requiere una aplicación que lea el nombre de un estudiante, el nombre de la asignatura y sus 3 notas parciales y presente un mensaje con sus datos y nota final.




```
//Declaración de Variables
Cadena [25] nombre
Cadena [20] asignatura
Real nota1, nota2, nota3, notaFinal

Inicio
  //Lectura de los datos de entrada
  escriba "Ingrese el nombre del estudiante:"
  lea nombre
  escriba "Ingrese el nombre de la asignatura:"
  lea asignatura
  escriba "Ingrese el valor de la primera nota:"
  lea nota1
  escriba "Ingrese el valor de la segunda nota:"
  lea nota2
  escriba "Ingrese el valor de la tercera nota:"
  lea nota3
  //Cálculo de la nota final
  notaFinal <- (nota1 + nota2 + nota3) / 3
  //Escritura de la salida
  llamar nueva_linea
  escriba "     INFORMACION DEL ESTUDIANTE"
  llamar nueva_linea
  escriba "NOMBRE------>",nombre
  llamar nueva_linea
  escriba "ASIGNATURA-->",asignatura
  llamar nueva_linea
  escriba "NOTA 1------>",nota1
  llamar nueva_linea
  escriba "NOTA 2------>",nota2
  llamar nueva_linea
  escriba "NOTA 3------>",nota3
  llamar nueva_linea
  escriba "NOTA FINAL-->",notaFinal
Fin
```

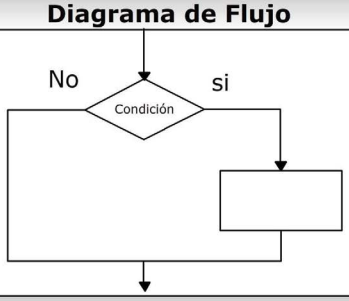
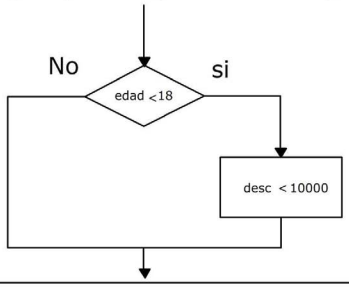
5.2 ESTRUCTURAS CONDICIONALES

5.2.1 Condicional Simple:

La estructura condicional simple verifica una condición y si esta es verdadera ejecuta las instrucciones que se encuentren dentro de la estructura. Por esta razón, a diferencia de los programas secuenciales donde siempre se ejecutan las mismas instrucciones, en los programas condicionales, existen instrucciones que su ejecución depende del cumplimiento de determinadas condiciones.

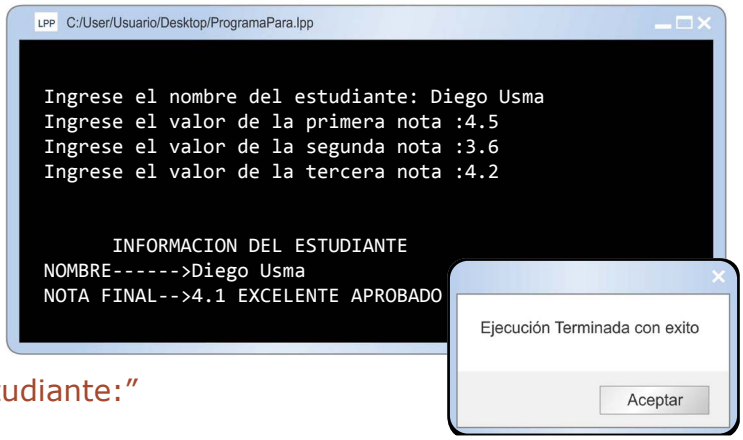
```
Inicia Robot
SI hay obstáculo Quitar
SI no hay obstáculo Avanzar
SI hay obstáculo Quitar
SI no hay obstáculo Avanzar
SI hay obstáculo Quitar
SI no hay obstáculo Avanzar
SI hay obstáculo Quitar
SI no hay obstáculo Avanzar
GirarIzquierda
SI hay obstáculoQuitar
SI no hay obstáculo Avanzar
SI hay obstáculoQuitar
SI no hay obstáculo Avanzar
GirarDerecha
SI hay obstáculoQuitar
SI no hay obstáculo Avanzar
SI hay obstáculoQuitar
SI no hay obstáculo Avanzar
SI hay obstáculoQuitar
SI no hay obstáculo Avanzar
Detener
Termina Robot
```

Sintaxis de una estructura condicional simple:

Diagrama de Flujo	LPP
	Si condición Entonces xxxxxxx Fin Si
Ejemplo Diagrama de Flujo	**Ejemplo LPP**
	Siedad < 18Entonces Desc<- 10000 Fin Si

PROGRAMA CONDICIONAL SIMPLE:

Se requiere una aplicación que lea el nombre de un estudiante y sus 3 notas parciales y presente un mensaje con su nombre y nota final. Si la nota final es inferior a 3, presentar el mensaje "REPROBADO", si su nota final es superior o igual a 3 y menor a 4, presentar el mensaje "APROBADO", y si su nota final es 4 o superior, presentar el mensaje "EXCELENTEMENTE APROBADO".




```
//Declaración de Variables
Cadena [25] nombre
Real nota1, nota2, nota3, notaFinal

Inicio
  //Lectura de los datos de entrada
  escriba "Ingrese el nombre del estudiante:"
  lea nombre
  escriba "Ingrese el valor de la primera nota:"
  lea nota1
  escriba "Ingrese el valor de la segunda nota:"
  lea nota2
  escriba "Ingrese el valor de la tercera nota:"
  lea nota3
  //Cálculo de la nota final
  notaFinal <- (nota1 + nota2 + nota3) / 3
  //Escritura de la salida
  llamar nueva_linea
  llamar nueva_linea
  escriba "     INFORMACION DEL ESTUDIANTE"
  llamar nueva_linea
  escriba "NOMBRE------>",nombre
  llamar nueva_linea
  escriba "NOTA FINAL-->",notaFinal
  Si notaFinal < 3 Entonces
    escriba " REPROBADO"
  Fin Si
  Si (notaFinal >= 3) Y (notaFinal < 4) Entonces
    escriba " APROBADO"
  Fin Si
  Si notaFinal >= 4 Entonces
    escriba " EXCELENTEMENTE APROBADO"
  Fin Si
Fin
```

5.2.2. Condicional Compuesto:

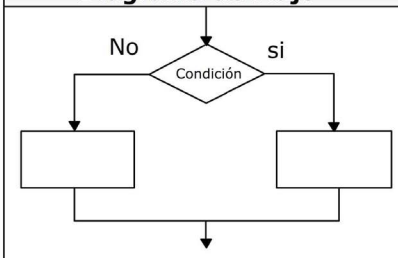
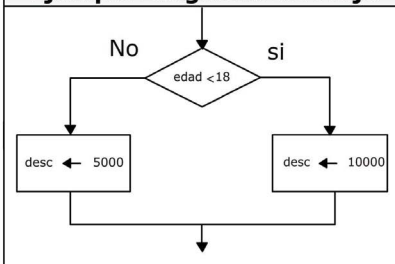
La estructura condicional compuesta verifica una condición y si esta es verdadera ejecuta unas instrucciones y si la condición es falsa, ejecuta otras instrucciones.

Su estructura es muy similar a la condicional simple, pero ahora se debe indicar las instrucciones que se deben ejecutar cuando la condición no se cumpla

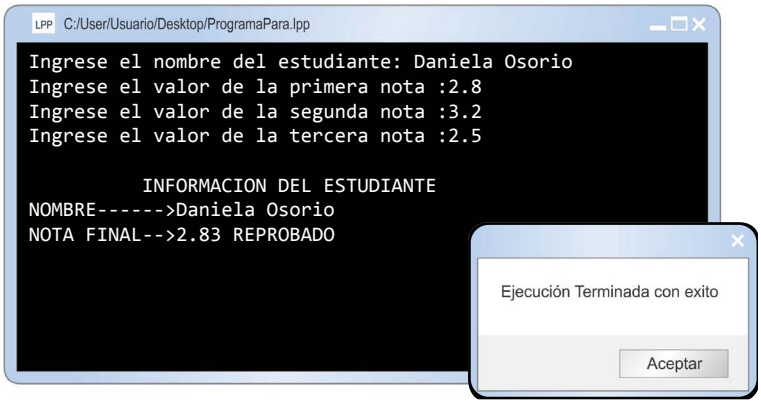
```
Inicia Robot

SI hay obstáculo
    Quitar
SINO
    Avanzar
FinSI

SI hay obstáculo Quitar SINO Avanzar
SI hay obstáculo Quitar SINO Avanzar
SI hay obstáculo Quitar SINO Avanzar
GirarIzquierda
SI hay obstáculo Quitar SINO Avanzar
SI hay obstáculo Quitar SINO Avanzar
GirarDerecha
SI hay obstáculo Quitar SINO Avanzar
SI hay obstáculo Quitar SINO Avanzar
SI hay obstáculo Quitar SINO Avanzar
Detener
Termina Robot
```

Diagrama de Flujo	LPP
	Si condición Entonces xxxxxxx Sino yyyyyyy Fin Si
Ejemplo Diagrama de Flujo	**Ejemplo LPP**
	Si edad < 18 Entonces Desc<- 10000 Sino Desc<- 5000 Fin Si

Se requiere una aplicación que lea el nombre de un estudiante y sus 3 notas parciales y presente un mensaje con su nombre y nota final. Si la nota final es inferior a 3, presentar el mensaje **"REPROBADO"**, en caso contrario, presentar el mensaje **"APROBADO"**.




```
//Declaración de Variables
Cadena [25] nombre
Real nota1, nota2, nota3, notaFinal

Inicio
  //Lectura de los datos de entrada
  escriba "Ingrese el nombre del estudiante:"
  lea nombre
  escriba "Ingrese el valor de la primera nota:"
  lea nota1
  escriba "Ingrese el valor de la segunda nota:"
  lea nota2
  escriba "Ingrese el valor de la tercera nota:"
  lea nota3
  //Cálculo de la nota final
  notaFinal <- (nota1 + nota2 + nota3) / 3
  //Escritura de la salida
  llamar nueva_linea
  llamar nueva_linea
  escriba "    INFORMACION DEL ESTUDIANTE"
  llamar nueva_linea
  escriba "NOMBRE------>",nombre
  llamar nueva_linea
  escriba "NOTA FINAL-->",notaFinal
  //Estructura Condicional Doble
  Si notaFinal < 3 Entonces
    escriba " REPROBADO"
  Sino
    escriba " APROBADO"
  Fin Si
Fin
```

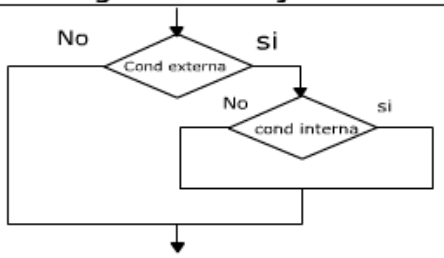
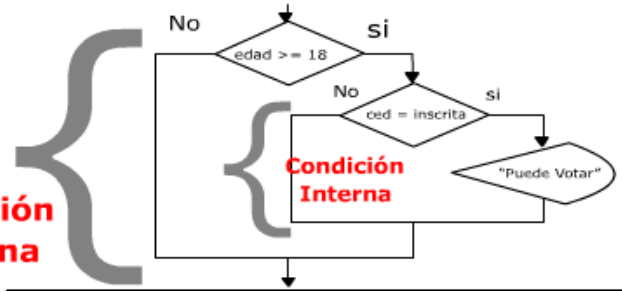
5.2.3. Condiciones Anidadas:

Las Condiciones Anidadas son simplemente la definición de una condición al interior de otra, no se trata de una estructura diferente o nueva, pero se debe tener un especial cuidado con su implementación al interior de un programa debido a que cada una de las estructuras debe estar correctamente definida.

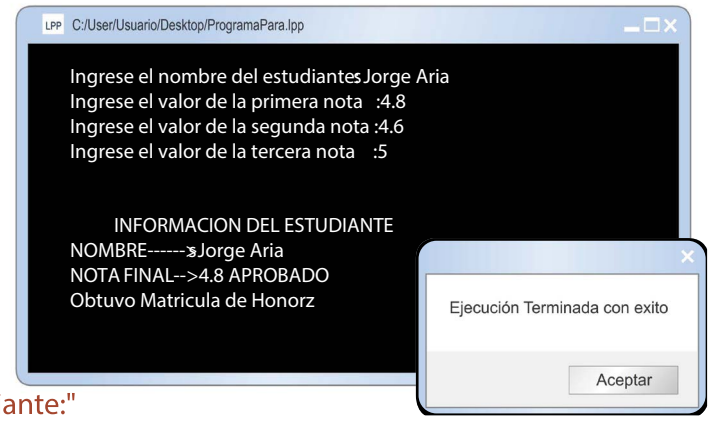
La condición interna puede estar en el flujo verdadero de la condición externa, en el flujo falso o en ambos. La condición interna además, puede contener otras condiciones en su interior.

```
Inicia Robot
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
GirarIzquierda
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
GirarDerecha
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
SI hay obstáculo Quitar
    SI COLOR = AZUL  Acomodar Izquierda
    SINO Acomodar Derecha
SINO Avanzar
    Detener
Termina Robot
```

Sintaxis de una estructura condicional compuesta:

Diagrama de Flujo	LPP
	Si condiciónExt Entonces Si condiciónInt Entonces Xxxxxxx Fin Si Sino yyyyyyy Fin Si
Ejemplo Diagrama de Flujo	**Ejemplo LPP**
	Si edad >= 18 Entonces Si ced = inscrita Entonces Escriba "Puede Votar" Fin Si Fin Si

Se requiere una aplicación que lea el nombre de un estudiante y sus 3 notas parciales y presente un mensaje con su nombre y nota final. Si la nota final es inferior a 3, presentar el mensaje **"REPROBADO"**, en caso contrario, presentar el mensaje **"APROBADO"**. A los estudiantes Aprobados cuya nota final esté por encima de 4.7 Indicarles que obtienen mención de honor.




```
//Declaración de Variables
Cadena [25] nombre
Real nota1, nota2, nota3, notaFinal

Inicio
  //Lectura de los datos de entradaZ
  escriba "Ingrese el nombre del estudiante:"
  lea nombre
  escriba "Ingrese el valor de la primera nota:"
  lea nota1
  escriba "Ingrese el valor de la segunda nota:"
  lea nota2
  escriba "Ingrese el valor de la tercera nota:"
  lea nota3
  //Cálculo de la nota final
  notaFinal <- (nota1 + nota2 + nota3) / 3Z
  //Escritura de la salida
  llamar nueva_linea
  llamar nueva_linea
  escriba "    INFORMACION DEL ESTUDIANTE"
  llamar nueva_linea
  escriba "NOMBRE------>",nombre
  llamar nueva_linea
  escriba "NOTA FINAL-->",notaFinal
  //Estructura Condicional Doble
  Si notaFinal < 3 Entonces
    escriba " REPROBADO"
  Sino
    escriba " APROBADO"
    Si notaFinal > 4.7 Entonces
      llamar nueva_linea
      escriba "Obtuvo Matricula de Honor"
    Fin Si
  Fin Si
Fin
```

BIBLIOGRAFÍA

Castillo Suazo, Rommel. (2001). Programación en LPP. 3-30.

GLOSARIO

DATO: Es una representación simbólica numérica, alfabética, algorítmica que puede ser un atributo o característica. Este, procesado se convierte en información.

INSTRUCCIÓN: Una instrucción es una unidad de creación de procedimientos a partir de la cual se construyen los programas.

LPP: Lenguaje de Programación para Principiantes.

SINTAXIS: Conjunto de normas que regulan la codificación de un programa.

VARIABLE: En ellas se pueden almacenar valores y son nombradas con identificadores, es decir nombres para poder identificarlas.

OBJETO DE APRENDIZAJE	Análisis Y Desarrollo De Sistemas De Información
Desarrollador de contenido Experto temático	**Andrés Julián Valencia Osorio**
Asesor Pedagógico	**Rafael Neftalí Lizcano Reyes** **Claudia Milena Hernandez Naranjo**
Productor Multimedia	**Diego Fernando Vega Ariza**
Programadores	**Daniel Eduardo Martínez Díaz**
Líder expertos temáticos	**Ana Yaqueline Chavarro Parra**
Líder línea de producción	**Santiago Lozada Garcés**