

# Instituto Tecnológico Superior de Chicontepec

## Ingeniería en Sistemas Computacionales

Nombre:

**Camelia Bautista Hernández**

Docente:

**Ing. Efrén Flores Cruz**

Asignatura:

**Programación Lógica y Funcional**

Unidad 1

**Conceptos Fundamentales**

**Carpeta de Evidencias**

**8º Semestre**

# Contenido

Introducción .....	3
1.1 Diferencias estilos de programación .....	4
1.2 Analizando diferentes de estilos de programación .....	12
1.2.1 Evaluación de expresiones .....	12
1.2.2 Tipos de Datos .....	16
1.2.3 Disciplina tipos .....	16
1.2.4 Funciones .....	16
Conclusión .....	22

## Introducción

La Programación Lógica y Funcional, es una asignatura que requiere tener conocimientos esenciales acerca de los lenguajes lógicos y funcionales de la inteligencia artificial, incluyendo la metodología y los aspectos relativos a la codificación, con el fin de ampliar el conocimiento de tecnologías alternativas para el desarrollo de sistemas automatizados y la implementación de agentes inteligentes. Estilo de programación (también llamado estándares de código o convención de código) es un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación. Así como también conocemos Una expresión es una combinación de operando y operadores. La evaluación de una expresión consiste en reducirla, esto es, realizar todas las operaciones contenidas en ella para obtener un valor final. Las funciones se crearon para evitar tener que repetir constantemente fragmentos de código.

## 1.1 Diferencias estilos de programación

Programación lógica y funcional

Criterios de evaluación

	DIA	MES	AÑO
Examen	35 %		
Carpetas de evidencias	10 %		
Prácticas	40 %		
Participación	5 %		
Tareas	5 %		
Asistencial	5 %		
			<u>100%</u>

Unidad 1 - Conceptos Fundamentales

### 1.1 - Diferentes estilos de programación

- Forma de hacer las cosas
- Se diferencian por reglas y conceptos que son aplicados para realizar un programa o aplicación
- Reducir el número de errores en el desarrollo de las aplicaciones.

"Si eliges un estilo para tu programa debes de seguir con el mismo estilo. El código debe estar bien escrito. Debe ser entendible para todos"

Cualidades	Consideraciones
<ul style="list-style-type: none"> <li>○ Expansibilidad</li> <li>○ Flexibilidad</li> <li>○ Reparabilidad</li> <li>○ Capacidad de evolución</li> <li>○ Comprendibilidad</li> </ul>	<ul style="list-style-type: none"> <li>○ Estilo debe ser uniforme</li> <li>○ Programas que puedan ser comprendidos de manera inmediata</li> <li>○ No debe promover fragmentos de código (ej. comentarios innecesarios)</li> <li>○ Definir lo más posible como corre el programa</li> <li>○ Indentación que la estructura sea correcta</li> </ul>



## Estilo K & R

El más utilizado en el lenguaje C y PHP, el estilo fue llamado de esta forma por que fue usado por Kernighan y Ritchie en su libro *the C Programming Language*.

Se trata de abrir la llave en la misma linea de declaración de la orden, indentando

## Estilo Allman

Fue definido por Eric Allman, se trata de crear una nueva linea para las llaves e indentar el código debajo de ellas.

La llave de cierre tiene el mismo identado que la de inicio

```
function saludar ($val){
```

```
{
```

```
    if ($val == 1)
```

```
{
```

```
    echo "Hola";
```

```
}
```

```
else
```

```
{
```

```
    echo "AAO";
```

```
}
```

## Estilo BSD KNF

También conocido como estilo Kernel Normal Form, es la más usada para el código de la distribución del software del sistema operativo de Berkeley. Es un extensión del K&R.

Se define un tabulador duro (8 espacios) el cual es usado para indentar bloques de código, mientras un tabulador suave (4 espacios) para todas las líneas continuas que acceden el espacio de visualización de la consola.



# Paradigmas de Programación

Propuesta tecnológica adoptada por una comunidad de programadores incuestionable en cuanto a que únicamente trata de resolver uno o varios problemas claramente delimitados.

Paradigma de programación es una propuesta tecnológica que es adoptada por una comunidad de programadores cuyo núcleo central es incuestionable en cuanto a que únicamente trata de resolver uno o varios problemas claramente delimitados.

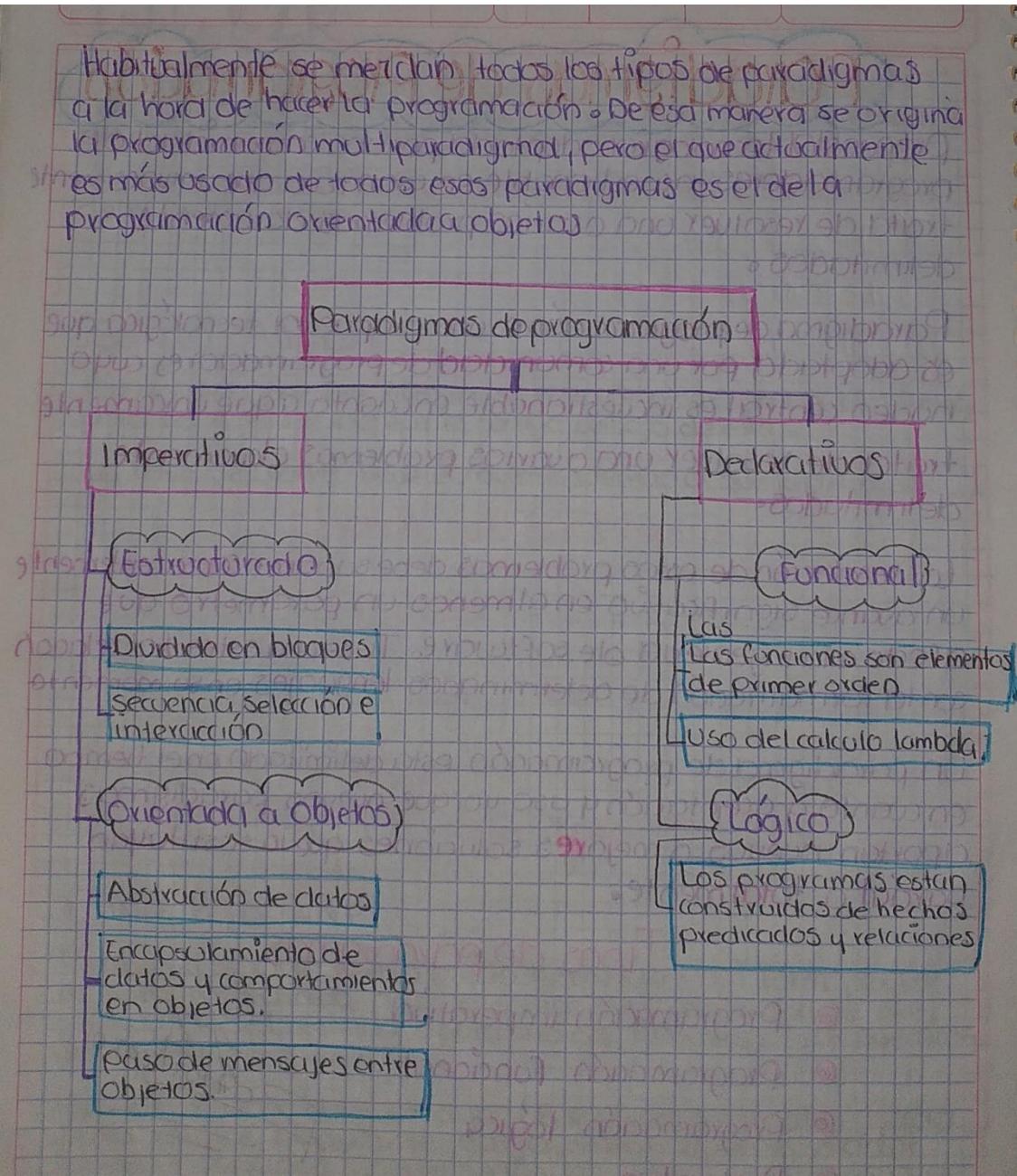
La resolución de estos problemas debe suponer, consecuentemente, un avance significativo en al menos un parámetro que afecte a la ingeniería de software. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento.

Un paradigmático de programación está delimitado en el tiempo en cuanto a aceptación y uso ya que nuevos paradigmas aportan nuevas o mejores soluciones que lo sustituyan parcial o totalmente.

## Tipos de paradigmas

- ① Programación imperativa
- ② Programación funcional
- ③ Programación lógica
- ④ Declarativo
- ⑤ POO orientado a objeto
- ⑥ Por procedimiento

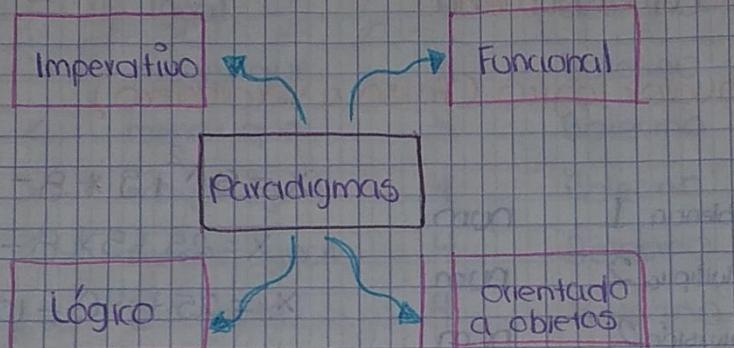
17/01/2020  
✓





## Orientada a objetos

El comportamiento del programado es llevado a cabo por objetos entidades que representan elementos o elementos del problema a resolver y tienen atributos y comportamiento.



Un paradigma de programación es un estilo de desarrollo de programa.

Es decir un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadra en uno o varios paradigmas a la vez al partir del tipo de órdenes que permite implementar algo que permiten implementar algo que tiene una relación directa con su sintaxis.

**Imperativo:** Los programas se componen de un conjunto de sentencias que cambian su estado. Son consecuencias de comandos que ordenan acciones al computador.

**Declarativo:** Opuesto al imperativo. Los programas describen los resultados esperados sin listar explícitamente los pasos a llevar a cabo para alcanzarlos.

**Lógico:** El problema se modela con enunciados de lógica de primer orden.

**Funcional:** Los programas se componen de funciones es decir implementaciones de comportamiento que reciben un conjunto de datos de entrada y devuelven un valor de salida.



Aritmético

Devuelven un valor numérico

Logico

Devuelven un valor logico (falso o verdadero)

^ Potencia 1 num.

\* Multiplic. 2 num

/ División 2 num

+ Suma 3 num

- Resta 3 num

$$x = 5^2 + 5 * 8 - 10^{12}$$

$$x = 25 + 5 * 8 - 100$$

$$x = 25 + 40 - 100$$

$$x = 65 - 100$$

$$\boxed{x = -35}$$

Ejercicio 1

$$7 * 10 - 15 / 3 * 4 + 9$$

$$x = 70 - 5 * 4 + 9$$

$$x = 70 - 20 + 9$$

$$x = 54 + 9$$

$$x = \boxed{49}$$

Ejercicio 2

$$9 + 7 * 8 - 36 / 5$$

$$x = 19 + 56 - 7.2$$

$$x = 165 - 7.2$$

$$x = \boxed{57.8}$$



TECNM  
TECNOLÓGICO NACIONAL DE  
MÉXICO



VERACRUZ  
GOBIERNO DEL ESTADO



SEV  
Secretaría  
de Educación



VERA  
CRUZ  
ME LLENA DE ORGULLO



DET  
Dirección de Educación  
Tecnológica del Estado  
de Veracruz



=	Igual	4	Boleano
>	mayor que	4	Boleano
<	menor que	4	Boleano
$\geq$	mayor igual	4	Boleano
$\leq$	menor igual	4	Boleano
$\neq$	Diferente	4	Boleano

Ejemplo:

$$x = 2 <= 2$$

$$x = 3 > 2$$

$$x = 2 = 2$$

$$x = 2 \neq 0$$

$$x = \text{Verdadero}$$

$$x = \text{Verdadero}$$

$$x = \text{Verdadero}$$

$$x = \text{Verdadero}$$

AND

$$\begin{matrix} V & V & V & 5 \\ V & F & F & 5 \end{matrix}$$

$$\begin{matrix} F & V & F & 5 \\ F & F & F & 5 \end{matrix}$$

$$\begin{matrix} F & F & F & 5 \\ F & F & F & 5 \end{matrix}$$

$$\begin{matrix} R = 10 > 8 & \text{OR} & 10 > 5 \\ V & & V \end{matrix}$$

$$\begin{matrix} R = V \end{matrix}$$

OR

$$\begin{matrix} V & V & V & 5 \\ V & F & V & 5 \end{matrix}$$

$$\begin{matrix} F & V & V & 5 \\ F & F & V & 5 \end{matrix}$$

$$\begin{matrix} F & F & V & 5 \\ F & F & F & 5 \end{matrix}$$

$$\begin{matrix} R = 10 > 8 & \text{OR} & 10 > 5 \\ V & & V \end{matrix}$$

$$\begin{matrix} R = V \end{matrix}$$

NOT

$$\begin{matrix} V - ; F & 5 \\ F - ; V & 5 \end{matrix}$$

$$\begin{matrix} F - ; V & 5 \\ F - ; F & 5 \end{matrix}$$

$$\begin{matrix} F - ; F & 5 \\ F - ; F & 5 \end{matrix}$$

$$\begin{matrix} R = 10 > 8 & \text{AND} & 20 > 5 \\ F & & F \end{matrix}$$

$$\begin{matrix} R = F \end{matrix}$$



$x = 2^3 \times 4 > 7 \times 3 \text{ AND } 12/3 > 2 \times 2$

$x = 8 \times 4 > 21 \text{ AND } 4 > 4$

$x = 32 > 21 \text{ AND } 4 > 4$

$x = V \text{ AND } F$

$x = F$

$\text{NOT } ((2 \times 3 \times 3 > 4 \times 2) \text{ AND } (3^2 > 5^2))$

$\text{NOT } ((18 > 8) \text{ AND } (9 > 25))$

$\text{NOT } (V \text{ AND } F)$

$\text{NOT } = F$

$R = V$

## Estilo whitesmiths

El estilo whitesmiths también llamado estilo wishart. Este estilo coloca las llaves asociadas con la instrucción de control identada en la siguiente linea.

Este estilo pone la llave que sigue a la declaración de un bloque. Se realiza identada en la linea siguiente. Las instrucciones dentro del bloque son identadas en el mismo nivel que la llave.

## Estilo 6NU

El estilo 6NU coloca una llave sobre la siguiente linea.

Las llaves son identadas por 2 espacios y el código que contiene identada por 2 espacios adicionales.

## 1.2 Analizando diferentes de estilos de programación

### 1.2.1 Evaluación de expresiones.

1 - Identar  
 2 - Saltos linea  
 3 - Espacio y linea en blanco

**1.2 Analizando diferentes de estilos de programación.**

**1.2.1 Evaluación de expresiones.**

- ⑥ Orden de la evaluación de los operadores
- ⑦ El orden en que se evalúan los operadores vienen dado por una reglas

**Reglas de precedencia:** El orden de como iras resolviendo la ecuación

**Reglas de asociatividad:** Si solo quedan de + y - se van de izquierda a derecha

**Uso de parentesis:** Se los primeros que se llevan a cada

**Prioridad de operadores**

Precedencia	Operador	Símbolo	Asociatividad
1	Incremento (prefijo)	++	Izquierda
	Decremento (sufijo)	--	
2	Incremento (prefijo)	++	Derecha
	Decremento (prefijo)	--	
	Signo (mas/menos)	+-	
3	Multiplicación/División	* / %	Izquierda
4	Suma / resta	+ -	Izquierda
5	Asignación	= += -= *= /= % =	Derecha



TECNM  
TECNICO NACIONAL DE  
MEXICO



VERACRUZ  
GOBIERNO DEL ESTADO



SEV  
Secretaría  
de Educación



VERA  
CRUZ

ME LLENA DE ORGULLO



DET  
Dirección de Educación  
Tecnológica del Estado  
de Veracruz



## Asociatividad

Operador

+ - \* / %

+ - (paréntesis)

= + - = \* = / = % =

Tipo de asociatividad

Izquierda

Derecha

Derecha

Asociativa por la izquierda: se agrupa de derecha a izquierda

$$3 + x - 7 \% 9 \rightarrow ((3 + x) - 7) \% 9$$

Asociativo por la derecha: se agrupa de izquierda a derecha

$$x = y + - 9 \rightarrow x = (y + - 9)$$

Ejemplo

$$x = a * b - 8 / ( + + b ) + 74 \% 29$$

Ejemplo:

$$x = a * (-8) / (+ + b) + 74 \% 2$$

1- Buscamos operadores con mas prioridad

Operador incremento (prefijo)

Suma / resta / uncarios (signo menor)

Ejemplo

$$(-3 - 4) - 1$$

$$(-7 - 1)$$

$$= -8$$

## Haskell

Funciona siguiendo el modelo de una calculadora, en la que se establece una sesión interactiva entre el ordenador y el usuario.

Una vez arrancado, el sistema muestra un prompt? y espera a que el usuario introduzca una expresión.

(1)

$$(3+2) * 8$$

$$= 5 * 8$$

$$\text{= 40}$$

(2)

$$2 * 5 + 2$$

$$(2 * 5) + 2$$

$$10 + 2 = \text{12}$$

## Asociatividad

Tabla de precedencia/<sup>asociatividad</sup> de operadores

9 !! InfixL: Sirve para declarar operadores asociativos a lo izquierdo

9 . InfixR: Sirve para declarar operadores asociativos a lo Derecho.

8 ^ InfixR:

7 \* infix 7 / ^

Infix 7 / "rem"; "div", "mod"

Infix 6 +/-

Infix 5 //

Infix 5 ++, -



TECNM  
TECNOLOGICO NACIONAL DE  
MEXICO



VERACRUZ  
GOBIERNO DEL ESTADO



SEV  
Secretaría  
de Educación



VERA CRUZ  
ME LLENA DE ORGULLO



DET  
Dirección de Educación  
Tecnológica del Estado  
de Veracruz



DIA	MES	AÑO	FOLIO
12	02	20	

Infix 4 = , / = , < , < = , > , > =

Infix 4 & elem /, NotElem /

Infix R 3 & elem /, NotElem /

Infix R 2 11

$$x + c * s \quad 8 * (s + e)$$

Ejemplo

$$(2+5)-2 \quad x; (y s + t y x) \quad x = y || x$$

7 - 2

$$(x = y) || x$$

5

2 \* 3 + 2 - 5

Suma > Suma 8 4

$$2 * 3 + 8 - 5$$

$$6 + 8 - 5$$

12

$$14 - 5$$

$$\cancel{9}$$

$$\text{Ared. } x y = x * y / 2$$

$$\text{Main} > \text{Ared} 4 * 2$$

$$\text{Perímetro } x = x + x + x + x$$

4

$$\text{Main} > \text{perímetro} 2$$

8

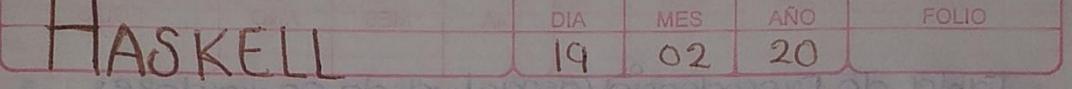
$$\text{Área } x = \pi * x ^ 2$$

$$\text{Main} > \text{área} 6$$

## 1.2.2 Tipos de Datos

## 1.2.3 Disciplina tipos

## 1.2.4 Funciones


  
 El entorno HUGS funcional siguiendo el modelo de un calculador en el que se establece una sesión interactiva entre el ordenador y el usuario

Una vez arrancado, el sistema muestra un prompt "?" y espera a que el usuario introduzca una expresión (denominada expresión inicial) y presione la tecla .

A. Un identificador comienza con una letra un dígito un apostrofe ('') o un subrayado (-).

Los identificadores que representan funciones o variables deben comenzar por letra minúscula (los identificadores que comienzan con letra mayúscula se emplean como funciones constructoras).

Ejemplo

`sum :: Int → [Int] → Int`

Los siguientes identificadores son palabras reservadas y no pueden utilizarse como nombres de funciones o variables

case	of	where	let	in	if
then	else	Data	type	infix	infixl
Infix	Primitive	Class	instance		

Cuando se trabajan con símbolos de operador es necesario tener en cuenta:

La precedencia. La expresión "2 \* 3 + 4" podría interpretarse como "(2 \* 3) + 4" o como "2 \* (3 + 4)".



1001	09	2018	DIA	MES	AÑO	FOLIO
08	10	2018				

Tabla de precedencia / asociatividad de operadores

Infixl	9	!!
Infixr	9	.
Infixr	8	^
Infixl	7	*
Infix	7	/, `div', `rem', `mod'
Infixl	6	+,-
Infix	5	\`
Infixr	5	++ , ;
Infix	4	=, /=, <, <=, >, >=
Infix	4	`elem', `notElem'
Infixr	3	
Infixx	2	

**La asociatividad:** La regla anterior resuelve ambigüedades cuando los símbolos de operador tienen distintos valores de precedencia, sin embargo, la expresión "1-2-3" puede ser tratada como "(1-2)-3" resultando 4 o como "1-(2-3)" resultando 2.

Infixl digit0 ops Para declarar operadores asociativos al izquierdo

Infixr digit0 ops Para declarar operadores asociativos al derecho

Infix digit0 ops Para declarar



Asociativo a la izquierda: si la expresión " $x-y-z$ " se toma como " $(x-y)-z$ "

Asociativo a la derecha: si la expresión " $x-y-z$ " se toma como " $x-(y-z)$ "

No asociativo: Si la expresión " $x-y-z$ " se vecharlo como un error sintáctico a operadores no asociativos.

En el standar prelude el  $(-)$  se toma como asociativo a la izquierda por lo que la expresión " $1-2-3$ " se tratará como " $(1-2)-3$ "

La expresión " $f(x+y)$ " equivale a  $(fx) + (gy)$ "

La expresión " $fx+1$ " que es tratada como " $(fx)+1$ " en el lugar de " $f(x+1)$ ".

### Tipos booleanos

$x \text{ || } y$  es True si y sólo si  $x$  e  $y$  son True

$x \text{ || } y$  es True si y sólo si  $x$  ó  $y$  ó ambas son True

$\text{not } x$  es el valor opuesto de  $x$ . (not True = False, not False = True)

### Operadores

$(+)$  Suma

$(\ast)$  Multiplicación

$(-)$  Substracción

$(^)$  Potenciación

negate menos unario (la expresión " $-x$ " se toma como "negate  $x$ "

díN división entera "

Vem vesto de la división entera - siguiendo la ley



$$(x \operatorname{div} y) * y + (x \% y) = x$$

mod modulo como veremos solo que el resultado tiene el mismo signo que el divisor.

Odd devuelve True si el argumento es impar

Even devuelve True si el argumento es par

gcd máximo común divisor

lcm mínimo común múltiplo

abs valor absoluto

signum devuelve -1, 0 o 1 si el argumento es negativo, cero o positivo respectivamente

## FLOTANTES

Representados por el tipo "float", los elementos de este tipo pueden ser utilizados para representar fraccionarios así como cantidades muy largas o muy pequeñas.

Por ejemplo 1.0e3 equivale a 1000.0

Mientras que 5.0e-2 equivale a 0.05

El standar prelude incluye tambien multiples funciones de manipulación de flotantes

pi, exp, log, sqrt, sin, cos, tan, asin, acos, atan etc

## Caracteres

Representadas por el tipo "char", los elementos de este tipo representan caracteres.

Individuales como los que se pueden introducir por teclado los valores de tipo carácter se escriben encerrando el valor entre comillas simples por ejemplo 'a'.

'o', '-' y 'z'



'\'' Barra invertida  
'\'' Comilla simple  
'\'' Comilla doble  
'\'' Salto de linea  
'\b' or '\bs' Backspace (espacio atrás)

'\oE!' Barrado  
'\or' HT Tabulador  
'\c' or '\BEf' alarma (campana)  
'\f' or '\FF' alimentación de papel

## Listas

El Standard prelude incluye un amplio conjunto de funciones de manejo de listas por ejemplo.

`length xs` devuelve el número de elementos de `xs`

`xs ++ ys` devuelve la lista resultante de concatenar las listas de `xs` e `ys`  
`Concat xs` devuelve la lista resultante de concatenar las listas de `xs`

`map f xs` devuelve la lista de valores obtenidos al aplicar la función `f` a cada uno de los elementos de la lista `xs`

## Ejemplos

? `length [1,3,10]`

3

? `[1,3,10] ++ [2,6,5,7]`

[1,3,10,2,6,5,7]

? `Concat [[1][2,3],[],[4,5,6]]`

[1,2,3,4,5,6]

? `map fromEnum ['H','O','I','D']`

[104,111,108,97]

?



Som [1..10]

La notación [1..10] representa la lista de enteros que van de 1 hasta 10 y sum es una función estandar que devuelve la suma de una lista de enteros. El resultado obtenido por el sistema es:

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55$$

## Cadenas.

Una cadena es tratada como una lista de caracteres y el tipo "string" se toma como una abreviación de "char". Las cadenas pueden ser escritas como consecuencia de caracteres, pueden utilizarse para las cadenas.

? "Hola"

hola.

Puesto que las cadenas son representadas como listas de caracteres, todas las funciones del standar prelude para listas pueden ser utilizadas también con cadenas.

? length "Hola"

4

? "Hola," ++ "amigo"

Holaamigo

? concat ['super', 'cali', 'fragi', 'listico']

supercali fragilistico

? map fromEnon "Hola"

[104, 111, 108, 97]

?

## Conclusión

Para evaluar una expresión es necesario conocer la prioridad de los operadores, con lo cual se puede determinar cuál operación se va a realizar antes que las demás. El estilo de programación es frecuentemente dependiente del lenguaje de programación que se haya elegido para escribir. Todos los lenguajes de programación tienen algunos elementos de formación primitivos para la descripción de los datos y de los procesos o transformaciones aplicadas a estos datos (tal como la suma de dos números o la selección de un elemento que forma parte de una colección). Estos elementos primitivos son definidos por reglas sintácticas y semánticas que describen su estructura y significado respectivamente. Los Tipos de Datos En lenguajes de programación un tipo de dato es un atributo de una parte de los datos que indica al ordenador (y/o al programador) algo sobre la clase de datos sobre los que se va a procesar.