Lenguajes de programación

Programación Secuencial en Erlang

Erlang

- Erlang es un lenguaje de programación concurrente, donde para su programación secuencial es un lenguaje funcional con evaluación ávida, tipos dinámicos y recolección de basura.
- Desarrollado por Joe Armstrong (1986) en Ericcson para soportar aplicaciones distribuidas, tolerantes a fallas, de tiempo real y que no tuvieran que parar.
- Llamado en honor de Agner K. Erlang (1878 –1929)
 - Matemático, estadístico y matemático danés
 - Inventor de la ingeniería de tráfico y la teoría de colas

Módulos y Funciones

- Un programa consiste de una declaración de módulo, una declaración de funciones a exportar y definiciones de funciones (secuencia de cláusulas).
- Se deben de grabar en un archivo texto con el mismo nombre que el módulo y extensión erl.
- Ejemplo:

```
% eleva un valor al doble
-module(doble).
-export([doble/1]).
doble(X) -> 2 * X.
```

```
Compilación y
Ejecución

Compilación Módulo: debe estar en directorio de trabajo

{ok, doble}

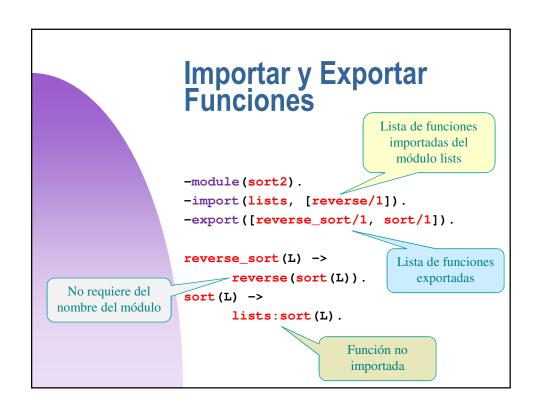
Ejecución

2> doble: doble (4) .

8

Módulo

Función
```



Operadores Aritméticos

Operador	Descripción	Prioridad
+ X	+ X	1
- X	Negación	1
X * Y	Multiplicación	2
X / Y	División	2
X div Y	División Entera	2
X rem Y	Residuo	2
X + Y	Suma	3
X – Y	Resta	3

```
Recursión
                 % calcula el factorial
                 -module(fact).
Caso Base:
                 -export([factorial/1]).
solución
                                            Guardia
directa
                 factorial(0) -> 1;
                 factorial(N) when N > 0 ->
                    N * factorial(N - 1).
Caso General:
                                           Variables en
llamada recursiva
                 1> fact:factorial(5).
                                          Mayúsculas
                 120
```

Guardias

- Condiciones que deben cumplirse para que una cláusula sea seleccionada
- Puede ser una condición simple () o una secuencia de condiciones simples separadas por comas
- La condición simple puede ser una comparación numérica, una comparación de un término o la llamada a un predicado predefinido por el sistema
- Ejemplos:

```
Tuplas

-module(conv).
-export([conv_long/1]).

conv_long({cm, X}) ->
{in, X / 2.54};
conv_long({in, Y}) ->
{cm, Y * 2.54}.

Atomo

1> conv:conv_long({cm, 2.54}).
{in, 1.0}
```

```
Listas

-module(llong).
-export([list_long/1]).

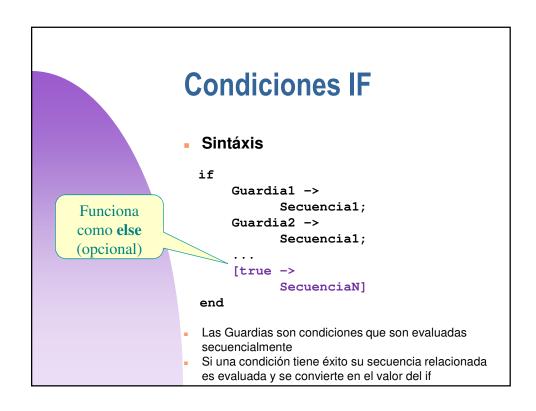
Constructor
de Listas

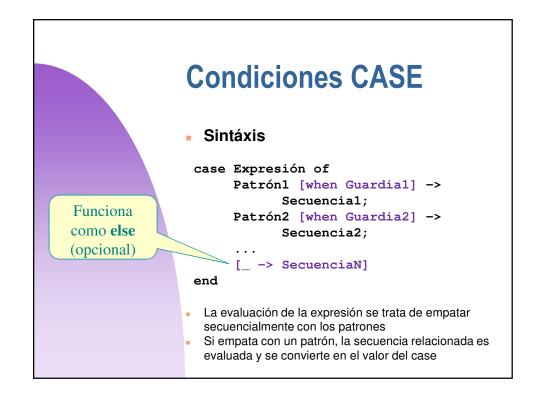
0;
list_long([_ | Rest]) ->

Variable
Anónima

1> llong:list_long([1,2,3,4]).

4
```





Operadores de Comparación

Operator	Description	Type
X > Y	X greater than Y	coerce
X < Y	X less than Y	coerce
X =< Y	X equal to or less than Y	coerce
X >= Y	X greater than or equal to Y	coerce
X == Y	X equal to Y	coerce
X /= Y	X not equal to Y	coerce
X =:= Y	X equal to Y	exact
X =/= Y	X not equal to Y	exact

Ejemplo If y Case

Ejemplo If y Case

```
case Month of
    sep -> 30;
                            Guardia
    apr -> 30;
    jun -> 30;
    nov -> 30;
    feb when Leap == leap -> 29;
    feb -> 28;
    jan -> 31;
   mar -> 31;
    may -> 31;
    jul -> 31;
    aug -> 31;
    oct -> 31;
    dec -> 31
end.
```

Funciones Predefinidas

- Usar el comando m(módulo).
- Módulos relevantes al curso:
 - erlang (precargado)
 - math
 - ♦ lists
 - ♦ io

Funciones Lambda

```
Ejemplo 1:
90> Xf = fun(X) -> X * 2 end.
#Fun<erl eval.5.123085357>
91> Xf(5).
10

Ejemplo 2:
47> (fun(X,Y) -> X * Y end)(2,3).
6
48>
```

Funciones de Orden Superior

foreach(Función, Lista)

```
95> Print_City = fun({City, {X, Temp}}) ->
io:format("~-15w ~w ~w~n", [City, X, Temp])
end.
#Fun<erl eval.5.123085357>
```

Despliegue con formato

```
96> lists:foreach(Print_City, [{moscow, {c, -10}}, {cape_town, {f, 70}}, {stockholm, {c, -4}}, {paris, {f, 28}}, {london, {f, 36}}]).

moscow c -10
cape_town f 70
stockholm c -4
paris f 28
london f 36
ok
```

Funciones de Orden Superior

map(Función, Lista)

Funciones con nombre

```
98> ctemps:convert_list_to_c([{moscow, {c, -10}}, {cape_town, {f, 70}}, {stockholm, {c, -4}}, {paris, {f, 28}}, {london, {f, 36}}]).
[{moscow, {c, -10}}, {cape_town, {c, 21}}, {stockholm, {c, -4}}, {paris, {c, -2}}, {london, {c, 2}}]
```

Funciones de Orden Superior

- apply(Módulo, Función, ListaArgs) ó apply({Módulo, Función}, ListaArgs)
- Ejemplos:

```
> apply(dates, classify_day, [monday]).
weekDay
> apply(math, sqrt, [4]).
2.0
> apply({erlang, atom_to_list}, [abc]).
[97,98,00]
```

Otras FOS

lists:all/2

lists:any/2

lists:fold1/3

lists:foldr/3

lists:sort/2

• •••