Erlang - den funktionella delen

Johan Montelius
KTH

Datastrukturer

- Literaler
  - atoms: foo, bar, ...
  - numbers: 123, 1.23, ...
  - nil: []
  - bool: true, false
- Sammansatta (compound)
  - tuples: {foo, 12, {bar, zot}}
  - lists: [], [foo, 12, bar, zot]

Variabler

- lexikalt omfång
  - proceduren
- otypat
  - ges ett värde när den skapas
- Syntax
  - X, Foo, BarZot, _

Tilldelning och mönstermatchning

- mönstermatchning:
  - <mönster> = <uttryck>
- ett mönster kan vara en variabel
  - Foo = 5
  - Bar = {foo, zot, 42}
- eller en sammansatt struktur
  - {A, B} = {4, 5}
  - {A, [_, B | T]} = {41, [foo, bar, zot]}

mönstermatchning kan misslyckas

- mönstermatchning kan vägleda en exekvering:
  - {person, Name, Number} = {dog, pluto}

Listor

- tomma listan (nil): []
- en cons-cell: [H | T]
- en lista med ett element:
  - [a | []]
  - [a]
- en lista med två element:
  - [a | [b | []]]
  - [a, b]

mönsternmatchning och listor

- [H | T] = [a]
- [H | T] = [a, b]
- [H | T] = [a, b, c]
- [H | T] = [a, [b, c]]
- [X, Y | T] = [a, b, c]
- H = a, T = [e,f], L = [H | T]
- H = a, T = b, L = [H | T]

Funktionsdefinition


```
area(X,Y) ->
     X*Y.
```

if-uttryck


```
fac(N) ->
     if
          N == 0 -> 1;
          N > 0 -> N*fac(N-1)
     end.
```

case-uttryck


```
sum(L) ->
      case L of
            [] ->
                  0;
            [H|T] ->
                  H + sum(T)
      end.
```

Implicit case-uttryck


```
sum([]) ->
      0;
sum([H|T]) ->
      H + sum(T).
```

mönstrermatchning


```
member(X,L) ->
      case L of
            [] ->
                  no;
            [X|_] ->
                  yes;
            [_|T] ->
                  member(X, T)
      end.
```

svansrekursion – inte


```
length(L) ->
      case L of
            [] ->
                  0;
            [_|T] ->
                  N = length(T),
                  1 + N
      end.
```

svansrekursion – ja


```
length(L) ->
     length(L, 0).

length(L, N) ->
     case L of
          [] ->
               N;
          [_|T] ->
               length(T, N+1)
     end.
```

moduler


```
-module(lst).
-export([reverse/1]).

reverse(L) ->
     reverse(L,[]).

reverse(L, A) ->
     case L of
          [] ->
               A;
          [H|T] ->
               reverse(T,[H|A])
     end.
```

moduler


```
-module(test).
-export([palindrome/1]).

palindrome(X) ->
      case lst:reverse(X) of
            X -> yes;
            _ -> no
      end.
```

programmeringsomgivning

- En interaktiv kommandotolk (Erlang shell)
- Kompilera, ladda och exekvera anrop.
- Kan köras helt oberoende eller inne i SDE (Eclipse, Emacs)
- Det finns en debugger men används inte så ofta.
- Använd loggutskrifter för att se vad som gick fel.