Industrial Functional Programming [1]

Melinda Tóth, István Bozó

Dept. Programming Languages and Compilers
Eötvös Loránd University, Budapest, Hungary

[1] Supported by TÁMOP-4.1.2.A/1-11/1-2011-0052

Contents

Variables

- Start with an Upper Case Letter or underscore character
- Single assignment
- There is no need to declare variables
- Passed by value
- Variables are local to a function clause
- ??? Global variable ???

Pattern Matching

- Controlling the execution flow
- Assigning values to variables
- Extracting values from compound data structures
- `X = 2`
- `{A, A, X} = {1, 1,3}`
- `[Head | Tail] = [1,2,3]`

Erlang mode in Emacs

```
(add-to-list 'exec-path "/path_to_erlang/bin")
(add-to-list 'load-path "/path_to_erlang/lib/tools-2.6.5/emacs")
(require 'erlang-start)
```

Modules

- Name of the module is an atom
- File extension: “.erl”
- `-module(module_name).`
- Sequence of forms (attributes and function declarations)
- Every form is terminated by a period (.)

Elements of the module

- Function definitions
- Attributes – module, export, import, include, compile, behaviour, user_defined
- Macros
- Records
- Header files
- Comment - %

Attributes

- `-module(module_name).`
- `-export([fun1/0, fun2/1]).`
- `-import(lists, [max/1, map/2]).`
- `-include("file").`
- `-compile(export_all).`
- `-author('TothMelinda').`

An example module

```
-module(amod).

-export([f/0]).

-author(melinda).

-date('20.12.2012.').

f()->
    ok.
```

Compiling And Loading Modules

- `c(amod)`
- `l(amod)`
- `code:get_path(), code:add_patha/1`
- Editing the `.erlang` file
- `erlc amod.erl`
- `amod:f().`

Functions – ModName:FunName/Arity

```
name(Arg11, ..., Arg1N) [when Guard1] ->
    ExprList1;
name(Arg21, ..., Arg2N) [when Guard2] ->
    ExprList2;
...
name(ArgM1, ..., ArgMN) [when GuardM]->
    ExprListM.

Exprlist ::=  Expr1, Expr2, ... ExprK.

modname:funname(Par1, ..., Parn)
        funname(Par1, ..., Parn)
```

Simple functions

```
fact(0) -> 1;
fact(N) when N>0 ->
    N * fact(N-1).

fib(1) -> 1;
fib(2) -> 1;
fib(N) when N>2 ->
    fib(N-1)+fib(N-2).
```

Sequence of expressions

```
fib(1) -> 1;
fib(2) -> 1;
fib(N) when N>2 ->
    N1 = fib(N-1),
    N2 = fib(N-2),
    Result = N1 + N2,
    Result.
```

On the Next Lecture ...

- Built-in Functions
- Documentations
- Recursive Functions
- Lists