Erlang Academy

Лекция 2

Автор курса: Костюшкин Сергей

План

- Типы данных в Erlang
- Булева алгебра и операторы сравнения
- Сопоставление по образцу (pattern matching)
- Рекурсия
- Модули

Типы данных

Числа:

- Целое (Integer)
- С плавающей запятой (Float)

Структуры:

- Ассоциативный массив (Map)
- Кортеж (Tuple)
- Список (List)

Идентификаторы:

- Атом (Atom)
- Идентификатор процесса (Pid)
- Порт (Port)
- Ссылка (Reference)

Типы данных

Низкоуровневые:
Бинарные данные (Binary)

Функциональные:
Функция (Fun)

Псевдотипы/Макротипы:

- Строка (String), в действительности является списком
- Запись (Record), в действительности является кортежом
- Булево значение (Boolean), в действительности является атомом

Integer (Целое число)

```
1> 10.
10
2> -234.
-234
3> 16#AB10F.
700687
4> 2#110111010.
442
5> 5#113121410.
520230
16> io:format("Binary palindrome: ~11.2.0B~n", [2015]).
Binary palindrome: 11111011111
ok
7> $A + $Z.
155
8> $Я.
1071
9> 99999999999999999999999999999999999999999999999999999999999999999999999999 + 1.
100000000000000000000000000000000000000000000000000000000000000000000000000
10> X = 1.
1
```

Float (Число с плавающей запятой)

```
1> f().
ok
2> 17.368.
17.368
3> -56.654.
-56.654
4> 12.34E-10.
1.234e-9
5> 1.00000000009765625 + 0.00000000000000001.
1.0000000000976563
6> Y = 0.1 + 0.2.
0.30000000000000004
7> X = 1.
1
8> X + Y.
1.3
```

Tuple (Кортеж)

```
1> {1, 2, 3}.
{1,2,3}
2> {1, {2, 3}, 4}.
{1,{2,3},4}
3> {}.
{}
4> {1,2,3,4,5,6,7,8,9,10,11,12}.
{1,2,3,4,5,6,7,8,9,10,11,12}
5> {ok, [ ]}.
{ok,[]}
6> {error, bad_credentials}.
{error,bad_credentials}
```

List (Список)

```
1> [1,2,3].
[1,2,3]
2> [a,b,c,d].
[a,b,c,d]
3> [[1,2,3],4,5,[6,7]].
[[1,2,3],4,5,[6,7]]
```

Proplist (Список свойств)

```
1> [{1, 2}, {3, 4}].
[{1,2},{3,4}]
2> [{key1, value1}, {key2, value2}].
[{key1,value1},{key2,value2}]
```

Map (Асоциативный масив)

```
1> #{1 => 2, 3 => 4}.
#{1 => 2,3 => 4}
2> #{key1 => value1, key2 => value2}.
#{key1 => value1,key2 => value2}
```

String (Строка) / List (Список)

```
1> String = "test".
"test"
2> String = [$t, $e, $s, $t].
"test"
3> String = [116, 101, 115, 116].
"test"
4> "Привет, мир!".
[1055,1088,1080,1074,1077,1090,44,32,1084,1080,1088,33]
```

Binary (Бинарные данные)

```
1> <<1,2,3,0,255>>.
<<1,2,3,0,255>>
2> Bin2 = <<"Some Text">>.
<<"Some Text">>
3> Bin2 == <<83, 111, 109, 101, 32, 84, 101, 120, 116>>.
true
4> <<"Несе Галя воду, коромисло гнеться..."/utf8>>.
<<208,157,208,181,209,129,208,181,32,208,147,208,176,208,
 187,209,143,32,208,178,208,190,208,180,209,131,44,32,
 208,...>>
5> unicode:characters_to_binary("Якая прыгожая ў цябе дачка").
<<208,175,208,186,208,176,209,143,32,208,191,209,128,209,
 139,208,179,208,190,208,182,208,176,209,143,32,209,158,
 32,...>>
6> unicode:characters_to_list(<<"Něco čerstvé"/utf8>>).
[78,283,99,111,32,269,101,114,115,116,118,233]
```

Atom (Атом)

```
1> abcdef.
abcdef
2> start_with_a_lower_case_letter.
start_with_a_lower_case_letter
3> 'Blanks can be quoted'.
'Blanks can be quoted'
4> 'Anything inside quotes \n\012'.
'Anything inside quotes \n\n'
5> ok.
ok
6> list_to_atom("Some Text").
'Some Text'
7> list_to_atom("Ой лишенько, яка халепа!").
** exception error: bad argument
    in function  list_to_atom/1
       called as list_to_atom([1054,1081,32,1083,1080,1096,1077,1085,1100,
                               1082,1086,44,32,1103,1082,1072,32,1093,1072,
                               1083,1077,1087,1072,33])
```

Атом в других языках

Ruby: Symbol

Javascript: Symbol.for(..)

SQL: Enum

Boolean (Булево значение) / Атом

```
1> true.
true
2> false.
false
3> 1 > 2.
false
4> 5 > 2.
true
```

Function (Функция)

```
1> F1 = fun(X) -> X + 1 end.
#Fun<erl_eval.6.50752066>
2> F1(5).
6
3> Map = fun lists:map/2.
#Fun<lists.map.2>
4> Map(F1, [1,2,3]).
[2,3,4]
5> M = lists.
lists
6> F = seq.
seq
7> M:F(1,10).
[1,2,3,4,5,6,7,8,9,10]
```

Pid (Идентификатор процесса)

```
1> spawn(fun() -> 1+1 end).
<0.35.0>
```

Port (Порт)

```
1> P = open_port({spawn, "tar -xzf -"}, [exit_status, binary]).
#Port<0.535>
```

Reference (ссылка)

```
1> make_ref().
#Ref<0.0.3.33>
```

Reference (ссылка)

Javascript: Symbol

Булевы логические операторы

- and (Логическое И)
- or (Логическое ИЛИ)
- xor (Логическое вычитание)
- not (Логическое НЕ)

- andalso (Ленивое И)
- orelse (Ленивое ИЛИ)

Бинарные логические операторы

- band (Логическое И)
- bor (Логическое ИЛИ)
- bxor (Логическое вычитание)
- bnot (Логическое НЕ)

Операторы сравнения

Операторы сравнения:

<	меньше
>	больше
=<	меньше или равно
>=	больше или равно
==	равно
/=	не равно
=:=	эквивалентно
=/=	не эквивалентно

Приоритеты типов:

number < atom < reference < fun < port < pid < tuple < map < list < bit string < binary

Сопоставление по образцу

Кортеж

X = {1, 2, 3}.

{1, A, B} = X.

A. %% 2

B. %% 3

Сопоставление по образцу

Список

```
L = [1, 2, 3].
[H | T] = L.
H. %% 1
T.  %% [2, 3]
```

Сопоставление по образцу

```
N = 10.                          %% Удастся - свяжет N с 10
{B, C, D} = {10, foo, bar}.      %% Удастся - свяжет B с 10,
                                 %%          C с foo, D с bar
{A, A, Z} = {abc, abc, foo}.     %% Удастся - свяжет A с abc,
                                 %%          Z с foo
{A, A, Y} = {abc, def, 123}.     %% Свалится

[K, L, M] = [1, 2, 3].           %% Удастся - свяжет K с 1,
                                 %%          L с 2, M с 3
[K, L, M, P] = [1, 2, 3].        %% Свалится
```

Сопоставление по образцу

Варианты сопоставления для списка

L = [1, 2, 3].

[1, 2, 3] = L.

[1 | [2, 3]] = L.

[1, 2 | [3]] = L.

[1, 2, 3 | [ ]] = L.

[1 | [2 | [3]]] = L.

[1 | [2 | [3 | [ ]]]] = L.

Сопоставление по образцу

```
[A,B|C] = [1,2,3,4,5,6,7]. %% Удастся - A = 1, B = 2,
                           %%           C = [3,4,5,6,7]
[H|T] = [1,2,3,4].         %% Удастся - H = 1, T = [2,3,4]
[H2|T2] = [abc].           %% Удастся - H2 = abc, T2 = []
[H3|T3] = [ ].             %% Свалится

{Atom,_, [N|_],{N}} = {abc,23,[22,x],{22}}.  %% Удастся
                                             %% Atom = abc
                                             %% N = 22

%% Последняя операция комбинирует в себе до 12
операций из языков без поддержки сопоставления по
образцу
```

Пример модуля

Файл: demo.erl

Содержимое:

```
-module(demo).
-export([double_num/1]).

double_num(X) ->
    times(X, 2).

times(X, N) ->
    X * N.
```

Рекурсия

```
double([H|T]) ->
    NewH = 2*H,
    NewT = double(T),
    [NewH|NewT];
double([ ]) ->
    [ ].
%% Компактная форма того же кода
double([H|T]) ->
    [2*H|double(T)];
double([ ]) ->
    [ ].
```

Хвостовая рекурсия

```
member(H, [H|_]) ->
    true;
member(H, [_|T]) ->
    member(H, T);
member(_, []) ->
    false.
```

Что почитать?

Палиндром

Системы счисления

Стандарт представления чисел с плавающей запятой