Tartalom

dp24a 2. FP-előadás, 2024-09-09 **1**
Untitled 1
Elixir Map (szótár, dictionary) 2

dp24a 2. FP-előadás, 2024-09-09

Untitled

```
defmodule Fpea do

 @moduledoc """
 Példák az FP előadásokhoz

 @author "hanak@emt.bme.hu"
 @date   "$LastChangedDate: 2024-09-09 07:58:34 +0200 (Mon, 09 Sep 2024) $$"
 """

 # Korábban: DpBev

 # A fájl nevét csupa kisbetűvel kell írni, szóköz helyett aláhúzás-jellel (snake case)
 # A modul nevét egybe kell írni, a szavakat nagybetűvel kell kezdeni (camel case)

 @spec fac(n::integer) :: f::integer
 # f = n! (azaz f az n faktoriálisa)
 def fac(0), do: 1          # ha az n=0 mintaillesztés sikeres
 def fac(n), do: n * fac(n-1) # ha az n=0 mintaillesztés sikertelen

 @spec sum(xs::[integer]) :: s::integer
 # Az xs számlista összege s
 def sum([]), do: 0
 def sum(xs)  do x = hd(xs); rs = tl(xs); x + sum rs end

 @spec append(xs::[any], ys::[any]) :: rs::[any]
 # rs az xs lista ys elé fűzésével kapott lista
 def append([], ys), do: ys
 def append(xs, ys), do: [(hd xs) | (append (tl xs), ys)]

 @spec revapp(xs::[any], ys::[any]) :: rs::[any]
 # rs a megfordított xs lista ys elé fűzésével kapott lista
 def revapp([], ys), do: ys
 def revapp(xs, ys), do: revapp((tl xs), [(hd xs) | ys])

 # Korábban: DpGen

 @spec sum_of_squares(a::number, b::number) :: s::number
 def sum_of_squares(a,b), do: sqr(a) + sqr(b)
 defp sqr(a), do: a*a # p[rivate], i.e. local in this module

 def sum_of_sqrs_b5(a, b \\ 5), do: sqr(a) + sqr(b)

 def sum_of_sqrs_a6b5(a \\ 6, b \\ 5), do: sqr(a) + sqr(b)

 def head([]),     do: {:error, nil}
 def head([x|_xs]), do: {:ok, x}

 _tail = fn [] -> {:error, nil}; [_x|xs] -> {:ok, xs} end
 # cnt_a = fn [] -> 0; [_x|xs] -> (cnt_a xs) + 1 end

 def cnt_n([]), do: 0; def cnt_n([_x|xs]), do: (cnt_n xs) + 1

 def hello_1(msg, name), do: (IO.write msg; IO.puts ", mizújs, #{name}?")
```

```
 def hello_2(msg, name), do: (  # csoportosítás zárójelezéssel
  IO.write msg              # pontosvessző helyett új sorba
  IO.puts ", mizújs, #{name}?"
 )

 def hello_3(msg, name) do     # zárójel helyett do...end
  IO.write msg
  IO.puts ", mizújs, #{name}?"
 end

 # Már látott megoldás
 # def fac(0), do: 1          # ha az n=0 mintaillesztés sikeres
 # def fac(n), do: n * fac(n-1) # ha az n=0 mintaillesztés sikertelen

 def fac_1(0),          do: 1          # ha az n=0 mintaillesztés sikeres
 def fac_1(n) when n > 0, do: n * fac_1(n-1) # ha az n=0 mintaillesztés sikertelen

  def fac_2(0),                      do: 1          # ha az n=0 mintaillesztés sikeres
 def fac_2(n) when is_integer(n) and n > 0, do: n * fac_2(n-1) # ha az n=0 mintaillesztés sikertelen

end

fac = &Fpea.fac/1
fac.(5)
(&Fpea.fac/1).(5)
```

Elixir Map (szótár, dictionary)

Definiáljunk egy egyszerű szótárat! (Mi a szótár? Közvetlen elérésű, azaz kulccsal indexelhető adatstruktúra, két oszlopból álló "tömb").

Az => jel előtt álló kifejezés a kulcs, az utána álló az érték.

Egy szótárat könnyű listává átalakítani, van rá függvény a Map modulban, lásd https://hexdocs.pm/elixir/1.17.2/Map.html.

```
mymap = %{{1,1} => 2, {2,2} => 4, {3,3} => 6, {4,4} => 8}
Map.to_list mymap
```

Visszafelé, azaz listát szótárrá alakítani, erre könyvtári függvényt írni azért nehezebb, mert nem lehet általánosságban megmondani, hogy melyik komponens legyen a kulcs, melyik az érték. A lista elemei lehetnek párok (mint a fenti példában), de lehetnek több elemű ennesek, listák, sztringek stb. Ezért **nincs** Map.from_list vagy hasonló funkciójú könyvtári függvény.

```
defmodule MyDict1 do
 def to_dict([{key, value}|kvs]) do
  Map.merge(%{key => value}, to_dict(kvs))
 end
 def to_dict([]), do: %{} # Map.new()
end
myls = [{{1, 1}, 2}, {{2, 2}, 4}, {{3, 3}, 6}, {{4, 4}, 8}]
MyDict1.to_dict(myls)

defmodule MyDict2 do
 def to_dict([{value, key}|kvs]) do
  Map.merge(%{key => value}, to_dict(kvs))
 end
 def to_dict([]), do: %{} # Map.new()
end
myls = [{{1, 1}, 2}, {{2, 2}, 4}, {{3, 3}, 6}, {{4, 4}, 8}]
MyDict2.to_dict(myls)

defmodule MyDict3 do
 def to_dict([{{k1, k2}, value}|kvs]) do
  Map.merge(%{k1*k2 => value}, to_dict(kvs))
 end
 def to_dict([]), do: %{} # Map.new()
```

```
end
myls = [{{1, 1}, 2}, {{2, 2}, 4}, {{3, 3}, 6}, {{4, 4}, 8}]
MyDict3.to_dict(myls)

defmodule MyDict4 do
  # jobbrekurzív megoldás akkumulátorral
  def to_dict(kvs), do: to_dict(kvs, %{}) # Map.new())

  def to_dict([{{k1, k2}, value}|kvs], dict) do
    to_dict(kvs, Map.merge(%{k1*k2 => value}, dict))
  end
  def to_dict([], dict), do: dict
end
myls = [{{1, 1}, 2}, {{2, 2}, 4}, {{3, 3}, 6}, {{4, 4}, 8}]
MyDict3.to_dict(myls)
```

A szótárral (map-pel), a %{...} jelöléssel gyors elérésű adatstruktúrákat hozhatunk létre futási időben, azaz dinamikusan. Ha a szótár szerkezetét, a benne lévő kulcsokat már fordításkor ismerjük, és ezek nem változhatnak futás közben, akkor a defstrukt makróval statikus szótárakat készíthetünk. Az ilyen szótárakban lévő értékeket frissíthetjük, de új kulccsal nem bővíthetjük.