

Тема 4 - ЕФП

1. Двоични файлове, низове и символни списъци

```
iex(31)> string="hello"  
"hello"  
iex(32)> is_binary string  
true
```

1.1. UTF-8 и Unicode

```
iex(37)> string="hello"  
"hello"  
iex(38)> byte_size string  
5  
iex(39)> String.length string  
5
```

```
iex(40)> ?a  
97  
iex(41)> ?W  
87  
iex(42)> String.codepoints("hello")  
["h", "e", "l", "l", "o"]
```

1.2 Двоични файлове (и бит string) Binaries (and bitstrings)

```
iex(43)> <<0, 1, 2, 3>>  
<<0, 1, 2, 3>>  
iex(44)> byte_size<<0, 1, 2, 3>>  
4  
iex(45)> String.valid?(<<239, 191, 191>>)  
true
```

```
iex(49)> <<256>>
<<0>>
iex(50)> <<256::size(16)>>
<<1, 0>>
iex(51)> <<256::utf8>>
"-A"
iex(52)> <<256::utf8,0>>
<<196, 128, 0>>
iex(53)> <<1::size(1)>>
<<1::size(1)>>
iex(54)> <<2::size(1)>>
<<0::size(1)>>
iex(55)> is_binary(<<1::size(1)>>)
false
```

```
iex(59)> is_bitstring(<<1::size(1)>>)
true
iex(60)> bit_size(<<1::size(1)>>)
1
iex(61)> <<0, 1, x>>=<<0, 1, 2>>
<<0, 1, 2>>
iex(62)> x
2
iex(63)> <<0, 1, x>>=<<0, 1, 2, 3>>
** (MatchError) no match of right hand side value: <<0, 1, 2, 3>>
```

```
iex(65)> "he"<>rest="hello"
"hello"
iex(66)> rest
"llo"
```

1.3. Char lists

```
iex(67)> 'hello'
'hello'
iex(68)> is_list'hello'
true
iex(69)> 'hello'
'hello'
```

```
iex(73)> to_string 'hello'
"hello"
iex(74)> to_string :hello
"hello"
iex(75)> to_string 1
"1"
```

2. Keywords, maps n dicts

2.1. Keywords lists

```
iex(84)> list=[{:a, 1}, {:b, 2}]
[a: 1, b: 2]
iex(85)> list=[a: 1, b: 2]
[a: 1, b: 2]
iex(86)> list==[a: 1, b: 2]
true
```

```
iex(90)> list ++ [c: 3]
[a: 1, b: 2, c: 3]
iex(91)> [a: 0] ++ list
[a: 0, a: 1, b: 2]
```

```
iex(92)> new_list=[a: 0] ++ list
[a: 0, a: 1, b: 2]
iex(93)> [a: 0] ++ list
[a: 0, a: 1, b: 2]
```

```
iex(99)> new_list=[a: 0] ++ list
[a: 0, a: 1, b: 2]
iex(100)> new_list[:a]
0
```

```
iex(103)> if false, do: :this, else: :that
:that
iex(104)> if(false, [do: :this, else: :that])
:that
```

```
iex(1)> [a: a]=[a: 1]
[a: 1]
iex(2)> a
1
iex(3)> [a: a]=[a: 1, b: 2]
** (MatchError) no match of right hand side value: [a: 1, b: 2]
```

2.2. Maps

```
iex(16)> map=%{:a =>1, 2 => :b}
%{2 => :b, :a => 1}
iex(17)> map[:a]
1
iex(18)> map[2]
:b
iex(19)> map[:c]
nil
```

```
iex(22)> map=%{1 => 1, 1 => 2}
warning: key 1 will be overridden in map
iex:22

%{1 => 2}
iex(23)> map=%{a: 1, b: 2}
%{a: 1, b: 2}
iex(24)> %{a: 1, b: 2}
%{a: 1, b: 2}
```

```
iex(27)> Map.get(%{:a =>1, 2 => :b}, :a)
1
iex(28)> Map.to_list(%{:a=>1, 2=>:b})
[{2, :b}, {:a, 1}]
```

```
iex(28)> Map.to_list(%{:a=>1, 2=>:b})
[{2, :b}, {:a, 1}]
iex(29)> map=%{:a=>1, 2=>:b}
%{2 => :b, :a => 1}
iex(30)> map.a
1
iex(31)> map.c
** (KeyError) key :c not found in: %{2 => :b, :a => 1}

iex(31)> %{map|:a=>2}
%{2 => :b, :a => 2}
```

```
iex(32)> %{map|:a =>2}
%{2 => :b, :a => 2}
iex(33)> %{map|:a =>3}
%{2 => :b, :a => 3}
```

2.3. Dicts

```
iex(34)> keyword=[]
[]
iex(35)> map=%{}
%{}
```

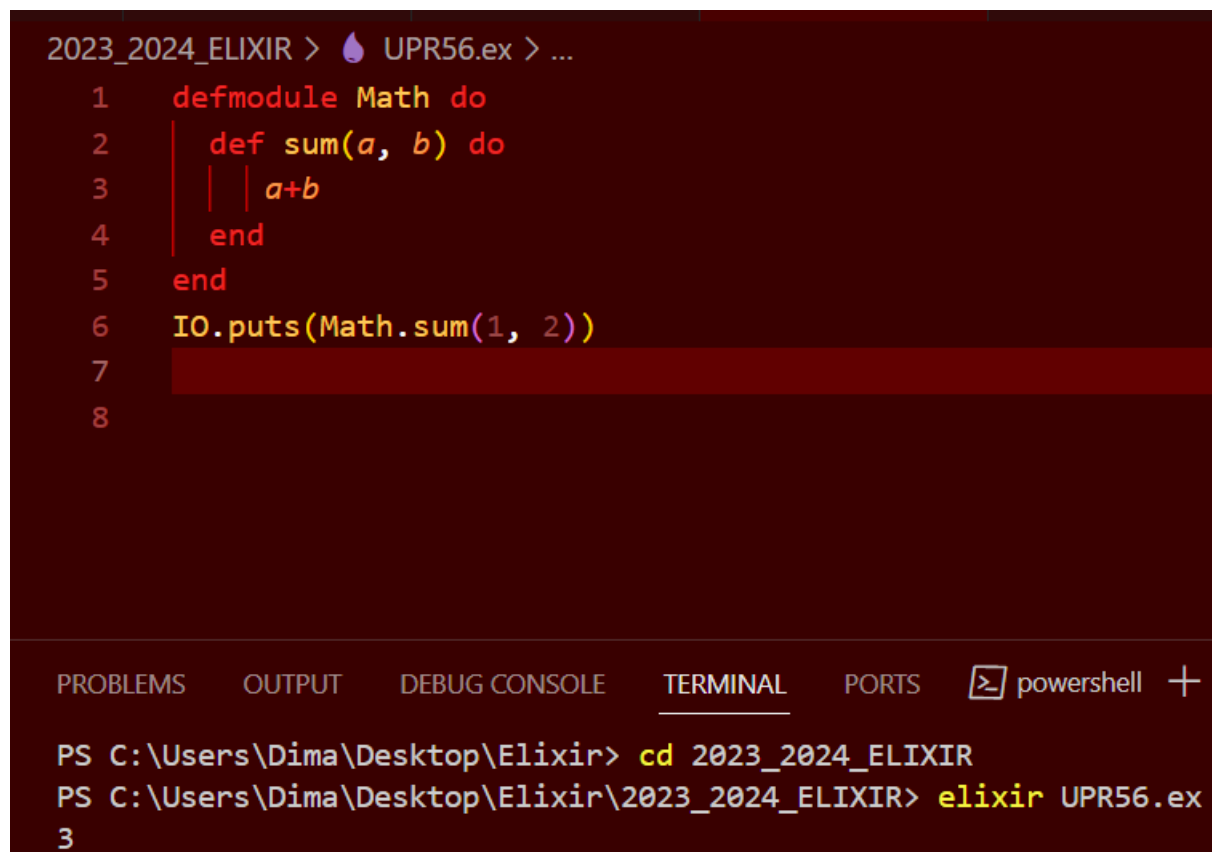
3. Modules

```
iex(44)> String.length"hello"
5
```

```
iex(1)> defmodule Math do
...(1)>   def sum(a, b) do
...(1)>     a+b
...(1)>   end
...(1)> end
{:module, Math,
 <<70, 79, 82, 49, 0, 0, 5, 16, 66, 69, 65, 77, 65, 116, 85, 56, 0, 0, 0, 1
36,
   0, 0, 0, 15, 11, 69, 108, 105, 120, 105, 114, 46, 77, 97, 116, 104, 8, 9
5,
   95, 105, 110, 102, 111, 95, 95, 10, 97, ...>>, {:sum, 2}}
iex(2)> Math.sum(1, 2)
3
```

3.1. Компиляция (Compilation)

3.2. Scripted mode



The screenshot shows an IDE with a dark theme. The top part displays a code editor with Elixir code in a file named `UPR56.ex`. The code defines a module `Math` with a function `sum` and calls it. The bottom part shows a terminal window with PowerShell commands.

```
2023_2024_ELIXIR > UPR56.ex > ...  
1  defmodule Math do  
2    |  def sum(a, b) do  
3    |    |  a+b  
4    |  end  
5  end  
6  IO.puts(Math.sum(1, 2))  
7  
8  
  
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  powershell +  
  
PS C:\Users\Dima\Desktop\Elixir> cd 2023_2024_ELIXIR  
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR56.ex  
3
```

3.3. Named functions

```
2023_2024_ELIXIR > UPR55.ex > ...
1  defmodule Math do
2    def sum(a, b) do
3      do_sum(a, b)
4    end
5
6    defp do_sum(a, b) do
7      a+b
8    end
9  end
10 IO.puts(Math.sum(1, 2))
11 IO.puts(Math.do_sum(1, 2))
12
```

PROBLEMS OUTPUT TERMINAL ... powershell + v [] [] ... ^ X

```
# 1
[1, 2, 3]

UPR50.ex:3: Math.zero?/1
UPR50.ex:13: (file)
(elixir 1.13.2) lib/code.ex:1183: Code.require_file/2
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR55.ex
3
** (UndefinedFunctionError) function Math.do_sum/2 is undefined or private
Math.do_sum(1, 2)
UPR55.ex:11: (file)
(elixir 1.13.2) lib/code.ex:1183: Code.require_file/2
```

3.4. Function capturing

```
2023_2024_ELIXIR > UPR50.ex > ...
1
2  defmodule Math do
3    def zero?(0) do
4      true
5    end
6
7    def zero?(x) when is_number(x) do
8      false
9    end
10  end
11  IO.puts(Math.zero?(0))  #=> true
12  IO.puts(Math.zero?(1))  #=> false
13  IO.puts(Math.zero?([1,2,3]))
14

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  powershell + v

PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR50.ex
true
false
** (FunctionClauseError) no function clause matching in Math.zero?/1

    The following arguments were given to Math.zero?/1:

        # 1
        [1, 2, 3]

UPR50.ex:3: Math.zero?/1
UPR50.ex:13: (file)
(elixir 1.13.2) lib/code.ex:1183: Code.require_file/2
```

3.5 Default arguments

2023_2024_ELIXIR > UPR52.ex > {} Concat > def join(a, b, sep)

```
1  defmodule Concat do
2    def join(a, b \\ nil, sep \\ "")
3
4    def join(a, b, _sep) when is_nil(b) do
5      a
6    end
7
8    def join(a, b, sep) do
9      a<>sep<>b
10   end
11 end
12
13 IO.puts(Concat.join("Hello","world")) #=> Hello world
14 IO.puts(Concat.join("Hello","world","_")) #=> Hello_world
15 IO.puts(Concat.join("Hello")) #=> Hello
16
```



PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

 powershell + v 

PS C:\Users\Dima\Desktop\Elixir> cd 2023_2024_ELIXIR

PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR52.ex

Hello world

Hello_world

Hello

```
2023_2024_ELIXIR > UPR53.ex > ...
1  defmodule Concat do
2    def join(a, b) do
3      IO.puts " ***First join"
4      a<>b
5    end
6
7    def join(a, b, sep \\ "") do
8      IO.puts " ***Second join"
9      a<>sep<>b
10   end
11 end
12 IO.puts(Concat.join("Hello","world", "_"))
13 IO.puts(Concat.join("Hello","world"))
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + -

```
Hello
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR53.ex
warning: this clause for join/2 cannot match because a previous clause at 1
ine 2 always matches
UPR53.ex:7

***Second join
Hello_world
***First join
Helloworld
```

4. Recursion

4.1. Цикли чрез рекурсия (Loops through recursion)

```
2023_2024_ELIXIR > UPR57.ex > ...
1  defmodule Recursion do
2    def print_multiple_times(msg, n) when n<=1 do
3      IO.puts msg
4    end
5    def print_multiple_times(msg, n) do
6      IO.puts msg print_multiple_times(msg, n-1)
7    end
8  end
9
10 IO.puts(Recursion.print_multiple_times("Hello", 3))
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + -

```
PS C:\Users\Dima\Desktop\Elixir> cd 2023_2024_ELIXIR
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR57.ex
** (CompileError) UPR57.ex:6: undefined function msg/1 (expected Recursion
to define such a function or for it to be imported, but none are available)
```

4.2. Алгоритми за „намаляване“ и „картографиране“ (“Reduce” and “map” algorithms)

2023_2024_ELIXIR > UPR51.ex > ...

```
1  defmodule Math do
2  def sum_list([head|tail], accumulator) do
3  sum_list(tail, head+accumulator)
4  end
5
6  def sum_list([], accumulator) do
7  accumulator
8  end
9  end
10 IO.puts(Math.sum_list([1,2,3],0))
11
```

PROBLEMS

OUTPUT

TERMINAL

...



powershell



PS C:\Users\Dima\Desktop\Elixir> CD 2023_2024_ELIXIR

PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR51.ex

6

```
2023_2024_ELIXIR > UPR51.ex > ...  
1  defmodule Math do  
2  def sum_list([head|tail], accumulator) do  
3  sum_list(tail, head+accumulator)  
4  end  
5  
6  def sum_list([], accumulator) do  
7  accumulator  
8  end  
9  end  
10 IO.puts(Math.sum_list([1,2,3],0))  
11 IO.puts(Math.sum_list([2,3],1))  
12 IO.puts(Math.sum_list([3],3))  
13 IO.puts(Math.sum_list([],6))  
14  
PROBLEMS OUTPUT TERMINAL ... powershell + v [ ] [ ] ... ^  
  
PS C:\Users\Dima\Desktop\Elixir> CD 2023_2024_ELIXIR  
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR51.ex  
6  
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR51.ex  
6  
6  
6  
6
```

```

2023_2024_ELIXIR > UPR58.ex > {} Math
1  defmodule Math do
2    def double_each([head|tail]) do
3      [head*2|double_each(tail)]
4    end
5    def double_each([]) do
6      []
7    end
8  end
9
10 IO.puts(Math.double_each([1, 2, 3]))
11

```

PROBLEMS OUTPUT TERMINAL ... powershell + - [] [] ... ^

```

PS C:\Users\Dima\Desktop\Elixir> cd 2023_2024_ELIXIR
PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> elixir UPR58.ex
0♦♦

```

```

iex(1)> Enum.reduce([1, 2, 3], 0, fn(x, acc)->x+acc end)
6
iex(2)> Enum.map([1, 2, 3], fn(x)->x*2 end)
[2, 4, 6]
iex(3)> Enum.reduce([1, 2, 3], 0, &+/2)
6

```

```

iex(4)> Enum.map([1, 2, 3], &(&1*2))
[2, 4, 6]

```

5. Enumerables and Stream

5.1. Enumerables

```

PS C:\Users\Dima\Desktop\Elixir\2023_2024_ELIXIR> iex.bat
Interactive Elixir (1.13.2) - press Ctrl+C to exit (type h()
ENTER for help)
iex(1)> Enum.map([1, 2, 3], fn x->x*2 end)
[2, 4, 6]
iex(2)> Enum.map(%{1=> 2, 3=>4}, fn {k, v}->k*v end)
[2, 12]
iex(3)> Enum.map(1..3, fn x->x*2 end)
[2, 4, 6]
iex(4)> Enum.reduce(1..3, 0, &+/2)
6

```

5.2. Eager vs Lazy

```
iex(5)> odd?=&(rem(&1,2)!=0)
```

```

warning: found identifier "odd?", ending with "?", followed
by =. It is unclear if you mean "odd ?=" or "odd? =". Please
add a space before or after ? to remove the ambiguity
iex:5:1

```

```
#Function<44.65746770/1 in :erl_eval.expr/5>
```

5.2.1. The pipe operator

```

iex(6)> Enum.filter(1..3, odd?)
[1, 3]
iex(7)> 1..100_000|>Enum.map(&(&1*3))|>Enum.filter(odd?)|>Enum.sum
7500000000

```

```

iex(8)> Enum.sum(Enum.filter(Enum.map(1..100_000, &(&1*3)), odd?))
7500000000

```

5.3. Streams

```

iex(9)> 1..100_000|>Stream.map(&(&1*3))|>Stream.filter(odd?)|>Enum.sum
7500000000
iex(10)> 1..100_000|>Stream.map(&(&1*3))
#Stream<[enum: 1..100000, funs: [#Function<48.50989570/1 in Stream.map/2>]]>

```

```

iex(11)> stream=Stream.cycle([1, 2, 3])
#Function<62.50989570/2 in Stream.unfold/2>
iex(12)> Enum.take(stream, 10)
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1]

```

```

iex(13)> stream=Stream.unfold("hello", &String.next_codepoint/1)
#Function<62.50989570/2 in Stream.unfold/2>
iex(14)> Enum.take(stream, 3)
["h", "e", "l"]

```

```

iex(3)> map=%{:a =>1, 2 => :b}
%{2 => :b, :a => 1}
iex(4)> map[:a]
1
iex(5)> map[:c]
nil

```

```

iex(16)> Enum.sum(Enum.filter(Enum.map(1..100_000, &(&1*3)), odd?))
7500000000
iex(17)> 1..100_000|>Stream.map(&(&1*3))|>Stream.filter(odd?)|>Enum.sum
7500000000

```

```

iex(26)> stream = Stream.cycle([1, 2, 3])
#Function<62.50989570/2 in Stream.unfold/2>
iex(27)> Enum.take(stream, 10)
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1]

```

```

iex(28)> stream = Stream.unfold("hello", &String.next_codepoint/1)
#Function<62.50989570/2 in Stream.unfold/2>
iex(29)> Enum.take(stream, 3)
["h", "e", "l"]

```

```

iex(30)> stream = File.stream!("path/to/file")
%File.Stream{
  line_or_bytes: :line,
  modes: [:raw, :read_ahead, :binary],
  path: "path/to/file",
  raw: true
}
iex(31)> Enum.take(stream, 10)
** (File.Error) could not stream "path/to/file": no such file or directory
(elixir 1.13.2) lib/file/stream.ex:83: anonymous fn/3 in Enumerable.File.Stream.reduce/3
(elixir 1.13.2) lib/stream.ex:1517: anonymous fn/5 in Stream.resource/3
(elixir 1.13.2) lib/enum.ex:3296: Enum.take/2

```


Задачи:

1 зад. Да се напише функция **perfect_n**, която проверява дали дадено естествено число е съвършено. Съвършени числа са тези, за които сумата на всичките им делители (без самото число) е равна на същото число:

Пример:

за perfect (6)	Резултат: 1+2+3=6
за perfect (28)	Резултат: 1+2+4+7+14=28
за perfect (33550336)	Резултат: 1+2+...+16775168=33550336

2 зад. Символен низ е съставен единствено от малки латински букви. Да се напише програма, която намира и извежда на екрана броя на срещанията на всяка от буквите на низа.

3 зад. Да се напише програма, която определя дали дадено положително цяло число с четен брой цифри на сумата от първите му n цифри е равна на сумата от последните му n цифри ($n = \text{брой_на_цифрите_на_числото} / 2$)

4 зад. Да се напише функция, която намира всички числа в даден интервал от положителни числа в десетична бройна система. Резултатът от тази функция е списък от прости числа, които прочетени и в двете посоки са прости числа. Списъкът да се подреди в нарастващ ред и от него да се изключат числата палиндроми.

Пример:

13 17 31 37 71 73

13 е просто число и прочетено наобратно е 31, което също е просто число.

В интервал от 2 до 100 числата са: 13 17 31 37 71 73 79 97

В интервала от 9900 до 10000 числата са: 9923 9931 9941 9967

5. Да се напише функция **increasing_n**, която проверява дали цифрите на дадено естествено число са в нарастващ ред, четени отляво-надясно:

Пример:

за increasing (12489) **Резултат:** Да в нарастващ ред са!

за increasing (4456) **Резултат:** Не не са в нарастващ ред!

Задължителна домашна работа

1. Да се дефинира функция **duplicates**. **list1** и **list2** са списъци от числа. Функцията построява списък от тези числа в **list1**, които се срещат повече от веднъж в **list2**.

Пример:

list1=(1 2 3)

list2=(1 2 1 3 2))

Резултат: duplicates (1 2)

2 зад. Дадено е цяло число x . Да се напише програма, която установява има ли в редица от цели числа $a_0, a_1, a_2, \dots, a_{n-1}$ ($1 \leq n \leq 100$) елемент, равен на x .

3 зад. Да се дефинира функция, която намира най-големия общ делител на две неотрицателни числа, поне едно от които е различно от 0.