Programming 2
Tutorial 6

Imperative Programming

Nedim Šišić
2023

Exercises – 1

Define a variable counter initialized to 0:

```
    let counter = ref 0;; .
```

Write a *next_val* function, with no parameters*, which in each call **increases** the value of counter by 1 and **returns** the new value.

(*you can declare the function like "let next_val _ = ..." or "let next_val () = ...")

Exercises – 1

Define a variable counter initialized to 0:

```
let counter = ref 0;; .
```

Write a *next_val* function, with no parameters*, which in each call **increases** the value of counter by 1 and **returns** the new value.
(*you can declare the function like "let next_val _ = ..." or "let next_val () = ...")

```
let counter = ref 0;;
let next_val _ =
  counter := (!counter) + 1;
  !counter;;
```

Exercises – 2

Write a *next_val2* function, with no parameters, which in each call **increases** the value of a variable *counter2* by 1 and **returns** the new value, with *counter2* **accessible** only inside the scope of *next_val*.

Exercises – 2

Write a *next_val2* function, with no parameters, which in each call **increases** the value of a variable *counter2* by 1 and **returns** the new value, with *counter2* **accessible** only inside the scope of *next_val*.

```
let next_val_2 =
  let counter_2 = ref 0 in
    fun () ->
      incr counter_2;
      !counter_2;;
```

Exercises – 3

a) Use the for loop to write a function that prints all divisors of a given natural number.

b) Solve the problem by using the while loop instead.

Exercises – 3

a)

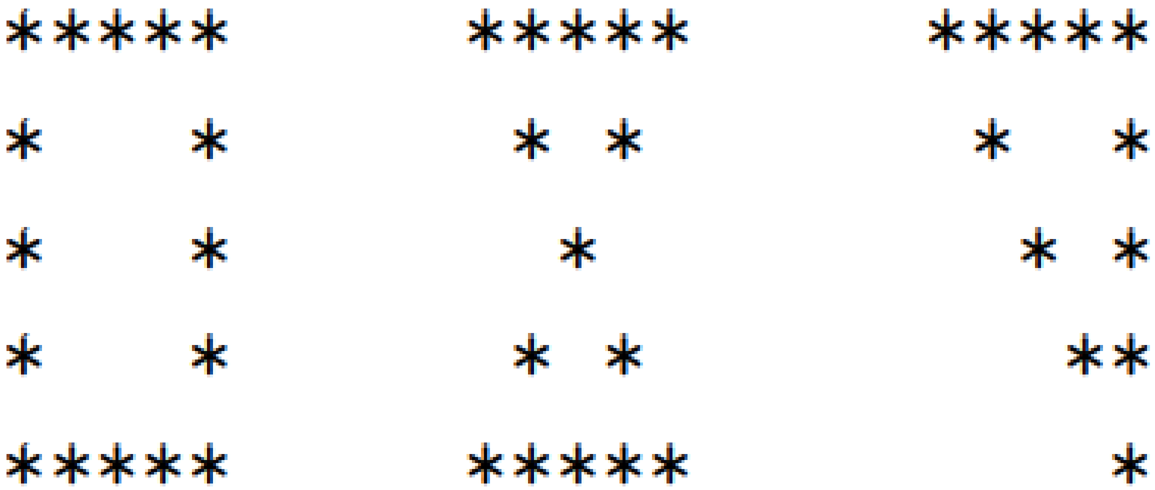
```
let divisors n =
  for i = 1 to n do
    if n mod i == 0 then (
      print_int i;
      print_string " "
    )
done;;
```

b)

```
let divisors2 n =
  let i = ref 1 in
    while !i <= n do
      if (n mod !i = 0) then
        (
          print_int !i;
          print_string "\n“
        );
      incr i
    done;;
```

Exercises – 3

c) For each of the following shapes (square, sandHour, upperTriangle) write a function that takes a parameter n and draws the given shape (examples are for n = 5).



```
*****      *****      *****
*   *       * *        *  *
*   *        *          * *
*   *       * *          **
*****      *****          *
```

Exercises – 3

c)
```
let square n =
for i=0 to n-1 do
  for j=0 to n-1 do
    if (i=0 || i=n-1 || j=0 || j=n-1) then
      print_string "*"
    else
      print_string " "
  done;
  print_string "\n“
done;;
```

square:

(i=0 || i=n-1 || j=0 || j=n-1)

sandHour:

(i = 0 || i = n-1 || i = j || i = n - 1 - j)

upperTriangle

(i = 0 || i = j || j = n - 1)

Exercises – 4

Write a function that returns an array containing the integers from 1 to n.

```
let array1toN n =
  let a = Array.make n 1 in
    for i = 1 to n do
      a.(i - 1) <- I
    done;
  a;;
```

Exercises – 5

Create an array *a_1* containing integers 1 and 2, and an array `a_2 = [| a_1; a_1; a_1|]`. Change the first element of the first element of *a_2*. What is now the value of a_2?

Arrays and references

```
let x = 0;;
    val x : int = 0
let y = [|x; x; x|];;
    val y : int array = [|0; 0; 0|]
y.(0) <- 1;;
    - : unit = <unknown constructor>
y;;
    - : int array = [|1; 0; 0|]
x;;
    - : int = 0
```

Arrays and references

```
let a_1 = [|1; 2|];;
let a_2 = [|a_1; a_1; a_1|];;
    val a_2 : int array array = [|[|1; 2|]; [|1; 2|]; [|1; 2|]|]
a_2.(0).(0) <- 5;;
a_2;;
    - : int array array = [|[|5; 2|]; [|5; 2|]; [|5; 2|]|]
a_1;;
    - : int array = [|5; 2|]
a_1.(1) <- 3;;
a_2;;
    - : int array array = [|[|5; 3|]; [|5; 3|]; [|5; 3|]|]
```

A change to one of *a_1* and the three arrays in *a_2* changes them all?!

Arrays and references

```
let a = Array.make 2 1;;
    val a : int array = [|1; 1|]
let b = a;;
    val b : int array = [|1; 1|]
b.(0) <- 5;;
    - : unit = <unknown constructor>
b;;
    - : int array = [|5; 1|]
a;;
    - : int array = [|5; 1|]
```

Here, variables *a* and *b* correspond to the same array in memory.

In the previous example, variable *a_1* and the three arrays in *a_2* all correspond to the same array.

Exercises – 7

Write a function that applies a given function *f* over the diagonal elements of matrix *m.*

```
let diag f m =
  for i = 0 to ((Array.length m) - 1) do
    m.(i).(i) <- f m.(i).(i)
  done;
m;;
```

Exercises – 8

Let *m_1* and *m_2* be matrices of integers and matrices, respectively:

```
let m_1 = Array.make_matrix 2 2 5;;

let m_2 = Array.make_matrix 2 2 (Array.make_matrix 2 2 3);;
```

a) What is the value of *m_1* when you change the element at position (0, 0) to 1?

b) What is the value of *m_2* when you change the element at position (0, 0) to (Array.make_matrix 2 2 1)?

c) Without doing b), what is the value of *m_2* when you change the element at position (0, 0, 0, 0) to 1?

Exercises – 9

Write a function that multiplies a matrix *m* with a vector *v*.

Example:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

Exercises – 9

```
let mult_mat_vec m v =
  let vec = Array.make (Array.length v) 0 in
    for i = 0 to ((Array.length m) - 1) do
      for j = 0 to ((Array.length m.(i)) - 1) do
        vec.(i) <- vec.(i) + m.(i).(j)*v.(j)
      done
    done;
let vec;;
```