CLEAN PROGRAMMING LANGUAGE DOCUMENTATION

Nicolas Nino Lozano

INTRODUCTION

Welcome to this intend of a Clean Programming Language Documentation, tailored for aspiring students venturing into the world of coding. This concise guide provides a beginner-friendly exploration of Clean, emphasizing clarity, simplicity, and good coding practices. Unlock the foundations of Clean programming and embark on a journey towards efficient and readable code.

What is Clean? it is a lazy functional programming language (a expression will not be evaluated if is not needed) which means is different from other programming languages which are eager(opposite).

Contents

HELLO WORLD

TO START A HELLO WORLD IS IMPORTANT TO HAVE THE BASIC STRUCTURE OF

THE CODE WHICH IS THE FOLLOWING ONE:

module NameOfTheFile (name of the module or file.Don't use funny simbols)

import StdEnv (use the most common library)

Start= "Hello World" (Start your main function with the string value you want to show)

TYPES OF DATA

AS ANY OTHER LANGUAGES THERE ARE TYPES A LOT OF TYPES OF DATA. IN

CLEAN WE HAVE:

- Int
- String
- Char
- Bool

OPERATORS

AS WELL WITH THE OPERATORS WE HAVE:

- <
- <=
- >=
- <>NOT EQUAL
- == EQUAL
- && AND
- || OR
- // COMMENTS ONE LINE / ... /
- not ()

HIGHER ORDER FUNCTIONS

Higher order functions where f is a function and [x:xs] a list

- map f [x:xs] = do f for all of the elements of the list
  map can also be used with a lamda expression like this

map (\x = 3*x) x

- filter f [x:xs] = choose those elements from a list which satisfy f which should return
  a bool value
  filter isEven[2 ,4 ,6 ,7 ,8 ,9]// [2, 4, 6, 8]
- takeWhile f [x:xs] = take numbers until the function is satisfied by and element of the list
  takeWhile isEven[2 ,4 ,6 ,7 ,8 ,9]// [2, 4, 6]
- dropWhile f [x:xs] = drop number until the functions is not satisfied anymore
  dropWhile isEven[2 ,4 ,6 ,7 ,8 ,9]// [7, 8, 9]
- foldr / foldl f z [x:xs] = operate all the elements of lists according to funcion. Usually
  is curryied function
  Start=foldr(+) 10 [1 , 2, 3]// 16
- until f1 f2 x = operate f2 until f1 is satisfies
  Start=until((<)10) ((+)2) 0// 12
- iterate f1 x = operate f1 in x infinitely
  Start=iterate inc1// infinite list [1..]

LIST COMPREHENSIONS

USING & AND COMMA : Be careful when using comma or & for list comprehension, because comma will create as many tuples as possible and & just create tuples by taking the position

Start = [(x,y) \\ x <- [1. .2], y <- [4..6]] // [(1,4),(1,5),(1,6),(2,4),(2,5),
(2,6)]

&

Start = [(x,y) \\ x <- [1. .2]& y <- [4..6]] // [(1,4),(2,5)]

TAKING CERTAIN PARAMETERS: Just take the values that satify the condition after the pipe

[(x,y) \\ x <— [1..5], y 4 <— [l..x] | isEven x] // [(2, 1), (2, 2),(4,1),
(4,2),(4,3),(4,4)]

REMARKABLE BUILT-IN FUNCTIONS

HERE YOU HAVE SOME RELEVANT FUNCTIONS FOR EACH TYPE OF DATA WHERE

X Y Z ARE ANY NUMBER, [x: xs]\ is a list:

FOR NUMBERS

Return INT:

- sqrt x = square root of X | X should be a Real value
- abs x = absolute value of x
- sign x/-x = 1/-1 | Returns the sign of a number
- gcd x y = z | Returns greatest common divisor
- exp x = e^x | Returns euler into the power of x
- Return boolean:
- isEven/ isOdd x = True/False | Check its type

Return list:

- repeat x = [x,x,x..] | *returns a list full of x
- repeat n x = [x..] | repeat n times x

FOR LISTS (x= [x:xs])

Return List:

- init x = Everything but the last one
- sort x = x | *Everything is sorted in ascending order
- is Member y x = True/False | *Check if y is member of x
- flatten [[x]] = [x] | *Combine sublists of a lists into one list
- Return Int:
- length [x: xs] = *length of [x: xs]
- prod [x: xs] = a | *is product of each element of x
- sum [x: xs] = a | *is the sum of every element of x
- last [x: xs] = a | is the last element of a list

Return tuples:

- splitAt x [x,y,z] = split the list from the x element
- e.g splitAt 3 [1,2,3,4] = ([1,2,3],[4])
- e.g splitAt 3 ['hello'] = (['h','e','l'],['l','o']) Interesting way to get a list of char from a String
- zip [x:xs] [y:ys] = [(x,y)..] | Do a list of tuples from two lists by putting the the first element in the first element of the tuple and the same with the second

FOR CHAR

Return bool:

- isUpper/isLower = True/False | Checks what type of char is

Return Char:

- toLower/toUpper c = C turns lowercase c to uppercase viceverse

FOR TUPLES

Return tuples:

- fst (a,b) = a | returns the first element of the tuple
- snd (a,b) = a | returns the second element of the tuple

RETURNS INT

- size string = x | prints the size of the length of string

RETURNS ANY

- (sortBy (\a b = a.something > b.something)