Some problems are in my notebook under 2/11/2019

1. Consider the following pseudo-code:

procedure foo(X):

Pass by reference:

X = a reference to y, so anything on x is the same to y

X = 1

Pass by value:

X = value(Y) 🡪 will print 1

X := X + 1

print X

end procedure

Pass by value:

Y is still 0

var Y = 0

Pass by reference:

Y = 0 🡪 Y = 1

foo(Y)

print Y

What is printed if parameters are passed by value? What is printed if they

are passed by reference?

2. When a procedure returns something, is it passed by value or reference?

It is passed by value because the procedure occurs so the procedure will affect the outer variable by returning it, because else, it will not affect the variable like we saw in pass by value above.

3. In Fortran 77, local variables are allocated statically (i.e., they live for the complete program execution).

In Java and C, they are allocated on the stack (i.e., their lifetimes are tied to their procedure's execution).

In Python and Haskell, they are allocated on the heap. What accounts for these differences?

4. Write pseudo-code for a procedure that will not work properly if local

variables are statically allocated but would work if local variables are

allocated on the stack or the heap.

5. In the following pseudo-code, variables and parameters are explicitly

declared with types, so that "var x : integer" means that x stores an integer

value. Assuming static scoping, indicate the visible variables and their types

at each of the lines marked with "-- (n)"

procedure P(A : integer):

var X : real

procedure Q(B : real):

var X : integer

-- (1)

end procedure

procedure R(A : integer, B : integer):

var Y : real

procedure S(A : real):

var X : string

-- (2)

end procedure

end procedure

-- (3)

end procedure

6. Write pseudo-code for a procedure that finds the minimum element of a

(non-empty) list, using the traverse procedure discussed in class (shown below).

procedure traverse(list, op):

while list is not empty:

op(list.data)

list := list.next

end while

end procedure

Answer:

Procedure Check item

If item < min

Min : = item

End if

End procedure

7. What would the following pseudo-code print, if we assume static scoping?

What would it print, if we assume dynamic scoping? Would it be possible to

simulate the behavior of dynamic scoping in a language with static scoping?

var x = 0

procedure tick():

x := x + 1

end procedure

procedure foo():

tick()

print x

end procedure

procedure bar():

var x = 10

foo()

end procedure

print x

foo()

bar()

print x

Call by value v call by reference??

* Memory location of the data procedure is key (see ex 1)
* Static scoping: just looking at the variable’s scope, finding the upper-closest definition and value, no need to draw the run-time stack.
* Dynamic scoping: draw the run-time stack structure, find the variable’s latest definition and value in the nearest location.