Lena Kemmelmeier

CS 326 - Programming Languages, Concepts and Implementation

Homework 3
Due: February 29th, 2024

CS326 – Programming Languages, Concepts and Implementation

Homework 3

(Due February 29)

1. (20 pts) Consider the following fragment of code in C:

```
{ int a, b, c;
      ...
      { int d, e;
            ...
            { int f;
                  ...
            }
            ...
      }
      ...
      { int g, h, i;
            ...
      }
      ...
}
```

Assume that each integer variable occupies four bytes. How much total space is required for the variables in this code? Justify your answer.

The total space required for the variables in this code is 24 bytes. Integers a, b, and c are live throughout this entire code section (12 bytes needed here) – they are in the outermost scope. Integers d and e are in a nested scope one level below the outermost scope (meaning we need an additional 8 bytes), and nested within this scope (two levels below the outermost one), is integer f (here we need 4 more bytes). We no longer need d, e, and f once we leave these two scopes, meaning this space can be used for something else later. When we enter the other nested scope (which was also one level below the outermost scope), we store g, h, and i (requiring 12 bytes). Because we no longer need the space for d, e, and f, we only need the 12 bytes for a, b, and c on top of the 12 bytes we already had (overlap).

2. (20 pts) Consider the following pseudocode, where procedures Q and R are nested inside procedure P:

```
procedure P (A, B: real)
      X: real
```

```
procedure Q (B, C: real)
      Y: real
      ...

procedure R (A, C: real)
      Z: real
      ... // (*)
...
```

Assuming static scope, what is the referencing environment (i.e., what names are known, and what do they refer to) at the location marked by (*)?

At the location marked by the asterisk, the referencing environment is made up of Z (local variable of R), R (procedure), A (one of R's parameters), C (one of R's parameters), Q (procedure, excluding its parameters/local variables), X (local variable of P), P (procedure), and B (one of P's parameters).

3. (30 pts) Consider the following pseudocode:

```
1. procedure main
2.    a: integer:= 1
3.    b: integer:= 2

4.    procedure middle
5.          b: integer:= a

6.          procedure inner
7.                print a, b

8.          a: integer:= 3

9.          // body of middle
10.         inner()
11.         print a, b

12.   // body of main
13.   middle()
14.   print a, b
```

(a) Suppose this was code for a language with the declaration-order rules of C (but with nested subroutines) - that is, names **must be declared before use**, and the scope of a name extends **from its declaration through the end of the block.** At each `print` statement, indicate which declarations of `a` and `b` are in the referencing environment. What does the program print (or will the compiler identify static semantic errors)?

- **For the print statement on line 7, a will have the value it's given on line 2 (1) and b will have the value it's given on line 5 (1).**
- **For the print statement on line 11, a will have the value it's given on line 8 (3) and b has the value from line 5 (1).**
- **For the print statement on line 14, a will have the value it's given on line 2 (1) and b has the value from line 3 (2).**
- **The program will print the following: 1 1 3 1 1 2.**

(b) Repeat the exercise for the declaration-order rules of C# (**names must be declared before use**, but the scope of a name is the **entire block** in which it is declared).

- **For the print statement on line 7, a has not been declared… this is also an issue with line 5 (trying to use a before declaration)**
- **This compiler <u>will</u> identify a standard semantic error for lines 5 and 7. The program will not print anything because a is trying to be used before it has been declared.**

(c) Repeat the exercise for the declaration-order rules Modula-3 (names can be declared in any order, and their scope is the **entire block** in which they are declared).

- **For the print statement on line 7, a will have the value it's given on line 8 (3) and b will have the value it's given on line 5 (3).**
- **For the print statement on line 11, a will have the value it's given on line 8 (3) and b has the value from line 5 (3).**
- **For the print statement on line 14, a will have the value it's given on line 2 (1) and b has the value from line 3 (2).**
- **The program will print the following: 3 3 3 3 1 2.**

4. (30 pts) Consider the following pseudocode:

```
x: integer:= 1
y: integer:= 2

procedure add
      x := x + y

procedure second (P: procedure)
      x: integer := 2
      P()

procedure first
      y: integer:= 3
      second(add)

// main program
first()
write integer(x)
```

(a) What does this program print if the language uses static scoping?

The program prints 3 (in add, x is 1 and y is 2, so x is 3).

(b) What does it print if the language uses dynamic scoping with deep binding?
The program prints 4 (in add, x is 2 and y is 2, so x is 4).

(c) What does it print if the language uses dynamic scoping with shallow binding?
The program prints 1 (value of the outer x).