Nguyen, Thomas
Coen 171
Professor Vu
October 11th, 2016

Homework #4

Problem Set:

5.2 Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?

Advantages:

- Provides more programming flexibility (can execute same code on differing variable types)
- Simple and clean program
- Let the language deal with assigning the correct type to what you declare

Disadvantages:

- Unclear what the type of the variable you are passing around is
- Easily leads to unreliability and type errors
- Unsure of what restrictions there may be on a variable (what is the range of an int?)

5.4 Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.

Variables are not initially defined using a keyword, but rather they are dynamically defined their types and their values during runtime. This means that:

- The type of a variable is changing depending on what is being assigned/executed to that variable
- The value or expected data structure of a variable is only bound to the heap (defined) once it is assigned the values.

5.6 Consider the following JavaScript skeletal program

a) Assuming static scoping, in the following, which declaration of x is the correct one for a reference to x?

a. Sub1 refers to the “var x” that is declared INSIDE Sub1
b. Sub2 refers to the “var x” that is declared INSIDE Sub1
c. Sub3 refers to the “var x” that is declared OUTSIDE Sub1

b) Repeat part a, but assume dynamic scoping.

a. Sub1 refers to the “var x” that is declared INSIDE Sub1
b. Sub1 refers to the “var x” that is declared INSIDE Sub1
c. Sub1 refers to the “var x” that is declared INSIDE Sub1

5.8 Consider the following JavaScript program:
List all the variables, along with the program units where they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used.

In Sub1:

- X: declared in main
- Y: declared in Sub1
- Z: declared in Sub1
- A: declared in Sub1

In Sub2:

- X: declared in main
- Y: declared in Sub1
- Z: declared in Sub2
- A: declared in Sub2
- B: declared in Sub2

In Sub3:

- X: declared in Sub3
- Y: declared in Sub1
- Z: declared in Sub2
- A: declared in Sub3
- B: declared in Sub2
- W: declared in Sub3

5.9 Consider the following Python program:
List all the variables, along with the program units where they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used.

In Sub1:

- X: declared in main
- Y: declared in main
- Z: declared in main
- A: declared in Sub1

In Sub2:

- X: declared in main
- Y: declared in Sub1
- Z: declared in Sub1
- A: declared in Sub2
- W: declared in Sub2

In Sub3:

- X: declared in main
- Y: declared in Sub1
- Z: declared in Sub3
- A: declared in Sub3
- W: declared in Sub2

5.12 Consider the following program, written in JavaScript-like syntax:
Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

a) Main calls sub1; sub1 calls sub2; sub2 calls sub3.
   a. In Sub3: a, x, w
   b. In Sub2: b, z
   c. In Sub1: y

b) Main calls sub1; sub1 calls sub3.
   a. In Sub3: a, x, w
   b. In Sub1: y, z

c) Main calls sub2; sub2 calls sub3; sub3 calls sub1.
   a. In Sub1: a, y, z
   b. In Sub3: x, w
   c. In Sub2: b

d) Main calls sub3; sub3 calls sub1.
   a. In Sub1: a, y, z
   b. In Sub3: x, w

e) Main calls sub1; sub1 calls sub3; sub3 calls sub2.
   a. In Sub2: a, b, z
   b. In Sub3: x, w
   c. In Sub1: y

f) Main calls sub3; sub3 calls sub2; sub2 calls sub1.
   a. In Sub1: a, y, z
   b. In Sub2: b
   c. In Sub3: x, w

Programming Exercises:

5.1 Perl allows both static and a kind of dynamic scoping. Write a Perl program that uses both and clearly shows the difference in effect of the two. Explain clearly the difference between the dynamic scoping described in this chapter and that implemented in Perl.

```
#!/usr/bin/perl
use strict;
use vars qw ($foo); # declares $foo as package global in the current package.

$foo = "a";

print "\$foo = $foo\n"; # prints "global value"

# Whenever called, forces $foo to be variable from global frame
print "staticScope result '", staticScope(), "'\n"; # prints "a"

# Whenever called, allows $foo to be variable from caller's frame
print "dynamicScope result '", dynamicScope(), "'\n"; # prints "b"


sub staticScope {
   my $foo = "b";
   showfoo();
}

sub dynamicScope {
   local $foo = "c";
   showfoo(); # ALWAYS prints "local value"
}

sub showfoo {
   return $foo;
}


# ---- Dynamic Scoping in this chapter vs Dynamic Scoping in Perl ----
# Chapter:
#      - Dynamic scoping is based on the calling sequence of subprogram,
#        not on their spatial relationship to each other. Thus, scope can
#        only be determined at run time
#
# Perl:
#      - When you declare a variable to be "local", you save away the old value
#        of the global variable and assign it a new value for the duration
#        of the subroutine.
#
```

```
[Thomass-MacBook-Pro:Homework 4 thomasnguyen$ ./one.pl
$foo = a
staticScope result 'a'
dynamicScope result 'c'
```

5.4 Repeat Programming Exercise 3 with Python

```
def one ():
   a = 1
   print str(a)
   def two():
      b = 2
      print str(a) + ' ' + str(b)
      def three():
         c = 3
         print str(a) + ' ' + str(b) + ' ' + str(c)
      three()
   two()
one()
```

```
[Thomass-MacBook-Pro:Homework 4 thomasnguyen$ python four.py
1
1 2
1 2 3
```

5.7 Write three functions in C or C++: one that declares a large array statically, one that declares the same large array on the stack, and one that creates the same large array from the heap. Call each of the subprograms a large number of times (at least 100,000) and output the time required by each. Explain the results.

When allocating memory on the heap as opposed to allocating memory on the stack, we see a large decrease in the efficiency we are able to allocate memory. This is because allocating memory on the stack is just simply an update to the stack pointer, whereas allocating memory on the heap requires looking for a big enough memory space, splitting it, then managing the "book-keeping" such as functions like "free();"

```
#include <iostream>
#include <string>
#include <ctime>

using namespace std;

void staticArray() {
   char staticArr[100000];
}

void dynamicArray() {
   new char[100000];
```

```
}

void timeTest(string array) {
   string strA ("staticArray");
   string strB ("dynamicArray");

   clock_t start;
   double duration;

   for (int i = 0; i < 100000; i++) {
      if (array.compare(strA) == 0) {
         staticArray();
      } else if (array.compare(strB) == 0) {
         dynamicArray();
      } else {
         cout << "Error. Proper array name not passed" << endl;
         break;
      }
   }
   duration = (clock() - start) / (double)CLOCKS_PER_SEC;
   if (array.compare(strA) == 0) {
      cout << "Static Array Duration: ";
   } else if (array.compare(strB) == 0) {
      cout << "Dynamic Array Duration: ";
   }
   cout << duration << " seconds" << endl;
}

int main() {
   string str1 ("staticArray");
   string str2 ("dynamicArray");

   timeTest(str1);
   timeTest(str2);
}
```

```
[Thomass-MacBook-Pro:Homework 4 thomasnguyen$ g++ seven.cpp
[Thomass-MacBook-Pro:Homework 4 thomasnguyen$ ./a.out
 Static Array Duration: 0.006693 seconds
 Dynamic Array Duration: 0.039347 seconds
```