Nick Venenga

Chapter 5

Review Questions

9. Define static binding and dynamic binding.
   - Static binding happens at compile time and dynamic binding happens at runtime.
10. What are the advantages and disadvantages of implicit declarations?
   - Implicit declarations facilitate ad-hoc variable declaration and can come in handy for shorter programs and prototyping where quick development or a one time use is much more important than maintainability. When code needs to be maintained by a large group of people and the code base is significant, explicit declaration has advantages. It is easier for other programmers to quickly see and understand data types. Also polymorphism and abstraction can be shown more clearly since you will see mixed types. In addition it is possible explicit declaration can decrease compile time--especially for larger programs with lots of data types where the compiler must spend more time determining the correct type.

Problem Set

9. Consider the following Python program:
   - sub1: (main) x; (sub1) a, y, z
   - sub2: (main) x, y, z; (sub2) a, w
   - sub3: (main) y; (sub2) a, w, x; (sub3) b, z

Chapter 6

a. Review Questions

6. What are the advantages of user-defined enumeration types?
   - User defined enumerations improve manageability and readability. They allow a programmer to specify a logical relation of data that is easy for the computer to understand (integer basis) while also allowing for easy reading by the programmer (word representation in code).

31. Define union, free union, and discriminated union.
   - A union is a data structure that can store different types of data depending on the context/requirement. For example, a particular union could hold an integer under certain conditions or a floating point under other conditions. A free union doesn't specify ("tag") what kind of data the union is holding. A discriminated union has a tag that specifies what type of data it's holding.

41. Describe the lazy and eager approaches to reclaiming garbage.
   - The eager approach to reclaiming garbage will reclaim memory as soon as all references to an object are destroyed or out of scope. This method minimizes memory consumption by quickly removing items from memory but increases CPU usage since a check has to be done every time scope is changed or references destroyed.

- The lazy approach checks for unreferenced pieces of memory when certain conditions are met. For instance, the garbage collector could wait until the system is idle or memory is being demanded and there isn't enough unallocated space to fulfill the request. This method leaves memory in use longer but reduces the CPU demands of the eager approach.
- ** From a security perspective, the eager approach offers benefits since certain risks can be mitigated when items are removed from memory quicker. For instance, credit card data may need to be left in memory unencrypted for the shortest possible amount of time.

45. Define strongly typed.

- A strongly typed language is one where data types are enforced at compile time. This means that type errors are always caught during compilation versus a weakly typed language where type errors can propagate during runtime.

b. Problem Set

14. Write a short discussion of what was lost and what was gained in Java's designers' decision to not include the pointers of C++.

- Java lost low-level optimization and fine-grained control over memory and data structures by not including C++ pointers. Certain optimizations can be obtained by manually managing memory when the programmer knows ahead of time exactly how the memory is going to be used. Java gained the ability to largely avoid memory leaks. Pointers make memory management more complicated and increase the risk of leaks since memory allocated to the heap must be explicitly deallocated or it is left there (a memory leak). Java reclaims some functionality of pointers by the use of references for all data of the Object type (pass by reference).

19. Compare the string manipulation capabilities of the class libraries of C++, Java, and C#.

- The standard library C++ has the least amount of string capabilities. To achieve more advanced manipulation, the programmer must create code themselves use a third-party library with additional functionality. For example, C++ doesn't have buffering built into strings and doesn't have a mutable form of strings. C# and Java both have significantly larger string functionality. For instance, they implement some form of buffering for building strings based on input. They also have capabilities such as StringBuilder which is a mutable form of string.

c. Programming Exercise

5. Write a simple program in C++ to investigate the safety of its enumeration types. Include at least 10 different operations on enumeration types to determine what incorrect or just silly things are legal. Now, write a C# program that does the same things and run it to determine how many of the incorrect or silly things are legal. Compare your results.

- Can't have different enums with same values -> O.K. in C#
  - Test {Item}; Test2 {Item}; //compile fail
- Enum values pollute global namespace -> O.K. in C# (you have to access values through enum object)

    - Test {Success, Exit}; //these are both accessible as “Success” and “Test”
- Local variables override enum values -> O.K. in C# since enums don’t pollute global namespace
    - Test {Item}; int Item = 10; //override
- Multiple values with same underlying value -> C# works the same
    - Test {Item1 = 1, Item2, Item3 = 2}; //1,2,2
- Type mixing possible
    - Test {Int = “int”, String = 1};
- Ad-hoc adding enum values -> Not allowed in C# (or in Visual C++)
    - Test{Item}; Test Item2;
- Enum pointers point to enum values but not ints even though enum values are ints. O.K. C# doesn’t allow pointers by default
    - Test{Item}; Test* item = new Test(); *item = Item; //is this a feature?

Enumerations are int type so can be used in math. Arithmetic, re-assignment, etc. Iteration over enum elements. Two enums with same element names

Chapter 7

a. Review Questions

15. What is referential transparency?
    - Referential transparency means functions with the same value can be interchanged for one another. It’s important in functional program and plays in with side effects. If a function doesn’t have side effects, it will always leave the machine in the same state as after every call. Therefor it can be moved around (i.e. swapped with another function) and the results will be the same.
16. What are the advantages of referential transparency?
    - Referential transparency makes functions more predictable. This increases readability and eases a programmer’s ability to understand how code is working. This also applies mathematical concepts which can make programs adhere to mathematical rules that may aid in development or understanding. In addition, it is easier to debug and prove correct a program with referential transparency since each function only relies on its arguments and can be checked independently of the rest of the program.
18. What is short-circuit evaluation?
    - A short circuit evaluation happens when there are chained expressions and the programming language stops evaluation of the expression as soon as it enters a state when it must evaluate to a certain condition. For instance, true or (anything) will always evaluate to true so the program return true without evaluating (anything).

b. Problem Sets

Show the order of evaluation of the following expressions:

a. a * b - 1 + c	$((a * b)^1 -1)^2 +c)^3$
b. a * (b - 1) / c mod d	$(((a * (b - 1)^1)^2 / c)^3 \bmod d)^4$
c. (a - b) / c & (d * e / a - 3)	$(((a - b)^1 / c)^5 \,\&\, (((d * e)^2 / a)^3 - 3)^4)^6$
d. -a or c = d and e	$((-a)^1$ or $((c = d)^2$ and $e)^3)^4$
e. a > b xor c or d <= 17	$(((a > b)^1$ xor $c)^3$ or $(d <= 17)^2)^4$
f. -a + b	$(-(a + b)^1)^2$

c. Programming Exercises (code attached)

time ./rand
real 0m0.043s

time mono rand.exe
real 0m0.073s

time java Random
real 0m0.104s

time python3 rand.py
real 0m0.239s

time scheme48 -h 40960000 < rand.scm
real 0m5.423s

issues with prolog but slowest I suspect

- Of course these aren't completely accurate since Java has the overhead of loading the virtual machine where C++ and C# don't--but that still effects time.
- Unsurprisingly interpreted are slower. I suspect prolog is pretty inline with scheme

Reflection (code attached)

151118542
(0,0)
John Doe 4

Java/Matrix (code attached) -- I didn't implement just write the method. It should be pretty trivial once the method is written though

0
8
3
9
-1
0
500499
44964
0
39500500
800019999