Maria Deslis
csci400 - spring2013 - Bahn
hw10

1. [2] What is the difference between a "keyword" and a "reserved word"?

+ A keyword is a word of a programming language that is special only in certain contexts.
+ A reserved word is a special word of a programming language that cannot be used as a name.

2. [2] What is the difference between an "lvalue" and an "rvalue"?

+ The address of a variable is sometimes called its l-value.
+ A variable's value is sometimes called its r-value because it is what is required when the name appears in the ride side of an assignment statement.

3. [2] What are the four categories of storage bindings for scalar variables based upon lifetime?
4. [4] For each category in #3, give describe when and how the variable is bound to its storage?
5. [4] For each category in #3, give two advantages and two disadvantages.
6. [4] For each category in #3, give an example of a variable in C that has that storage binding. If that type of binding is not supported in C, give an example of a language that does support it.

+Static Variables

-Bound to memory cells before program execution begins and remains bound to those same memory cells until program execution terminates

-Advantages:

i. Efficiency; all addressing of static variables can be direct whereas other kinds of variables often require indirect addressing, which is slower.

ii. No runtime overhead is incurred for allocation and deallocation of static variables, although this time is often negligible

-Disadvantages:

i. Static binding to storage is reduced to flexibility; in particular, a language that has only static vars cannot support recursive subprograms.

ii. Storage cannot be shared among variables

- Example: (C/C++)

```
char lexical() {
    static int position = -1;
    position = position + 1;
    return text[position];
}
```

+Stack-Dynamic Variables

-Storage bindings are created when their declaration statements are elaborated, but whose types are statically bound.

-Advantages:

i. To be useful, recursive subprograms require some form of dynamic local storage so that each active copy of the recursive subprogram has its own version of the local variables.

ii. Conserves storage

-Disadvantages:

i. Overhead of allocation and deallocation

ii. Subprograms cannot be history sensitive

iii. Inefficient references/Indirect addressing

-Example: (Java)

```
void Recursion(int n) {
    if (n > 0) {
        n--;
        Recursion(n);
        cout << n << " ";
        }
}

void main() {
        Recursion(3);
}
```

+Explicit Heap-Dynamic Variables

-Nameless (abstract) memory cells that are allocated and deallocated by explicit run0-time instructions written by the programmer. These variables which are allocated from and deallocated to the heap can only be referenced through pointer or reference variables.

-Advantages:

i. Allows easy implementation of dynamic structures

ii. Capable of error detection by the compiler

-Disadvantages:

i. Pointers and references used to implement dynamic structures are more difficult to use and not quite as efficient as other variable types

ii. Highly disorganized because of the unpredictability of its use

-Example: (C/C++)

```
int * intnode;           // Creates a pointer
intnode = new int;   // allocates an int cell
...
delete intnode;      // deallocates the cell to which intnode points
```

+Implicit Heap-Dynamic Variables

-Bound to heap storage only when they are assigned values

-Advantages:

i. Highest degree of flexibility, allowing highly generic code to be written

ii. All attributes are bound every time they are assigned

-Disadvantages:

i. If implicit heap-dynamic variables is the run-time overhead of maintaining all the dynamic attributes, which could include array subscript types and ranges, etc.

ii. Loss of some error detection by the compiler

-Example: (Javascript)

```
list = [74, 84, 86, 90, 71];
```

This Javascript array is only brought into being when its assigned values. It then gets placed in the heap as a collection of those values. It would bind those values to that variable. It would not necessarily tell you what type they were. That part will be dependent on other language semantics.

7. [2] Static scoping is known by what other name?

+ Static scoping is also sometimes called lexical scoping

8. [4] What are some of the disadvantages of dynamic scoping?

i. During the time span beginning when a subprogram begins its execution and ending when that execution ends, the local variables of the subprogram are all visible to any other executing subprogram, regardless of its textual proximity or how execution got to the currently executing program. There is no way to protect local variables from this accessibility. Subprograms that have not yet completed their executions. Thus results in less reliable programming than that of static scoping.

ii. The inability to type check references to non-locals statically. This problem results from the inability to statically find the declaration for a variable referenced as a nonlocal

iii. Also makes programs much more difficult to read, because the calling sequence of subprograms must be known to determine the meaning of references to nonlocal variables.

iv. Access to nonlocal variables in dynamic-scoped languages take far longer than accesses to non-locals when static scoping is used.

9. [4] What does a “referencing environment” refer to?

+ The referencing environment of a statement is the collection of all names that are visible in the statement.

10. [2] In C, what types of variables are implicitly initialized?

+ Variables with static storage duration are implicitly initialized with NULL. The initial value of all other variables is undefined.

Type	**Implicit Initial Value**
byte, short, int, long	0
boolean	false
char	'\0000'
float	+0.0f
double	+0.0
object reference (including String)	null