Scope and Persistence (Existence) of Variables and Identifiers

SFWR ENG 2B03

2003

Robert L. Baber

A Program and its Subcomponents

```
program A
var x (global)
x := 1
print “A”, x
call B
call B
…
end program A
```

```
procedure B
    var x (local)
    x := 2
    print “B”, x
    call C
    x := 3
end procedure B
```

```
procedure C
    procedure D
    …
    end procedure D
    var y (own, state)
    ...
    print “C”, x, y
    ...
end procedure C
```

Scope of the Variables in Program A

Variable	Scope
x (in A)	A, C, D (not B)
x (in B)	B only (neither A, C nor D)
y (in C)	C (also D in some programming languages)

In A, C and D, the name x refers to the variable declared in A. In B, the name x refers to the variable declared in B.

In C (and possibly D), the name y refers to the variable declared in C. Elsewhere, y is undefined.

Persistence of the Variables in Program A

Variable	Persistence (existence)
x (in A)	A, B, C, D (everywhere)
x (in B)	B, C, D (not A)
y (in C)	A, B, C, D (everywhere)

The variable x declared in A exists throughout program execution.

The variable x declared in B exists only while B is active (being executed).

The variable y declared in C exists effectively throughout program execution.

Scope and Persistence of the Variables x

While C and D are executing, both variables named x exist. Their values are maintained. The variable x declared in B continues to exist because B is still active; B has called C (which presumably calls D).

References to x in C and D are references to the (global) variable x declared in A.

Creation and Release of the Variables x

The variable x declared in A is created when the execution of A begins. This variable is released (eliminated, ceases to exist) when the execution of the program A terminates.

Similarly, the variable x declared in B is created when B starts to execute (is called). It continues to exist during execution of B's call to C. This variable x is released (eliminated, ceases to exist) when the execution of B terminates.

Creation and Release of the Variable y

The state variable y declared in C is created when the execution of C begins *the first time* (with some programming language implementations, even earlier). This variable is released (eliminated, ceases to exist) when the execution of the *entire program* A terminates (not before).

In the earlier literature, such variables were also sometimes called “own” variables.

EquivalentProgram, Variables Renamed

```
program A
var xa (global)
xa := 1
print “A”, xa
call B
call B
…
end program A
```

```
procedure B
    var xb (local)
    xb := 2
    print “B”, xb
    call C
    xb := 3
end procedure B
```

```
procedure C
    procedure D
    …
    end procedure D
    var y (own, state)
    ...
    print “C”, xa, y
    ...
end procedure C
```

Persistence of Variables

Global and state variables, in particular their values, exist throughout program execution, whether or not they can be referenced from all parts of the program.

Local variables cease to exist when control returns outside their declared scope. Their values disappear, are no longer maintained.

Persistence of State vs. Local Variables

The values of *state* variables

- are *maintained* between calls to the routine or module in which they are declared, i.e.,
- still exist between a return and the next call.

The values of *local* variables

- are *lost* between calls to the routine in which they are declared, i.e.,
- do *not exist* between a return and next call.

Scope ≠ Range of Persistence (Existence)

If a statement within the scope of a variable is being executed, the variable must, clearly, exist.

The reverse is not necessarily true. State variables are *local in scope* but *global in persistence* (existence).

Loosely speaking, therefore,

scope $\subseteq$ range of persistence

Summary

- The set of statements in a program which may contain references to a variable is the *scope* of that variable.
- If a variable and its value exist while a statement in a program is being executed, that statement is in the range of persistence (existence) of that variable.
- Scope and persistence are not necessarily the same.