Scoping

Principles of Programming Languages

Colorado School of Mines

https://lambda.mines.edu

Learning Group Activity

Start working on the Tic-Tac-Toe activity with your group. I'll come around and answer any questions you have about the activity, and give you a check for "glancing over it".

Save your Exam Questions

We'll have an exam review session during the second half of class.

Scope

Note

Materials from today (with the exception of the review session) are on Exam 2, not Exam 1.

- A **scope** refers to the period of which a variable is visible.
- Languages limit the scope of variables to be just within the function to help avoid the risk of collision in two identically named variables. But it raises the question: *what does it mean to be within a function?*
  - In **static scoping**[1], the variable is visible when it is within the same function *structure-wise*.
  - In **dynamic scoping**, the variable is visible when it is within the same function *environment-wise*.

[1] Also called "lexical scoping".

Static Scoping: Example

```
procedure A (
    x ← 0
    procedure B (
        ▷ x is visible within B
        y ← x + 1
    )
    procedure C (
        ▷ x is visible within C, but not y
        x ← x − 2
    )
    B()
    C()
)
```

Any downsides? What happens when we add first class functions?

Closures

When we add first class functions to static scoping, the function must carry around an environment of the variables it depends on.

```
/* A Lehmer Linear Congruential Generator in JavaScript */
function prng(seed) {
    var x = seed;
    function next() {
        x = (16807 * x) % 2147483647;
        return x;
    }
    return next;
}
```

What other uses could you use closures for?

Dynamic Scoping: Example

In dynamic scoping, the calling environment determines the visible variables.

```
procedure A (
    ▷ x from caller
    print x
    x ←
x + 5
)
```

```
procedure B (
    x ←
0
    A()
    print x
)
```

```
procedure C (
    x ←
2
    A()
    print x
)
```

Advantages: easy first-class functions without need to carry calling environment

Scoping in the Wild

- **Static:** C, C++, Pascal, Python
- **Dynamic:** C preprocessor macros, Early Lisps, Emacs Lisp, Bash
- **Allows both:** Common Lisp, Scheme, Perl

Exam Review Time

- Any questions on exam material?
- We will go through exam topics, and do a quick review otherwise.