CSCI 334: Principles of Programming Languages

Lecture 16: Scope

Instructor: Dan Barowy

Williams

Outline

What is scope?

What are the kinds?

Why is it important?

How do they work?

Scope

Recall that a **variable** is a named placeholder for a value in an expression. **Scope** is a set of rules that determines **what value** is returned when a variable is used in an expression.

Kinds

There are two main kinds of scope.

- **Lexical** scope
- **Dynamic** scope

Both definitions depend on a notion of **time**.

- Lexical scope depends on **compile time**.
- Dynamic scope depends on **run time**.

Importance

Scope rules are used to determine:

- Which **values** are returned.
- When **garbage collection** is run.

Scope rules can have an impact on whether programmers write **buggy** programs. Here are some languages with **confusing** scope rules:

- JavaScript
- R
- LISP (the original)

Dynamic scope

Dynamic scope is a rule that finds the **most recent value** of a given variable in a program's execution (i.e, at **run time**).

Lexical scope

Lexical scope is a rule that uses the **lexically closest value** of a variable at the time the use was defined (i.e., at **compile time**).

Kinds

"Confusing" languages either have a flawed/complicated version of lexical scope (e.g., R, JavaScript) or use dynamic scope (the original LISP).

Example

```
x = 10;

def f()
{
   print x."\n";
}

def g()
{
   x = 20;

   return f();
}

g();
```

What gets printed out?

Perl Examples

```
local $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   local $x = 20;

   return f();
}

g();
```

Dynamic scope
(`local` keyword)

```
my $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   my $x = 20;

   return f();
}

g();
```

Lexical scope
(`my` keyword)

Perl Examples

(let's try these)

How do they work?

(whiteboard)

Recap & Next Class

This lecture:

Scope

Next lecture:

Object-oriented programming