CSCI 334:
Principles of Programming Languages

Lecture 19: Scope & Testing

Instructor: Dan Barowy

Williams

Outline

What is scope?

What are the kinds?

Why is it important?

How do they work?

Unit testing

Your to-dos

1. No quiz this week.
2. Project checkpoint #2, **due Sunday 12/4**
3. If you found checkpoint #1 difficult, come see me! Office hours today 2:30-3:30pm; Fri 11-1pm.

Final project timeline

1. Minimal working version, **due Sun 11/27**
2. Draft language specification, **due Sun 12/4**
3. Mostly working version, **due Sun 12/11**
4. Project + video presentation, **due Sun 12/18**

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.

If your language does not have **functions** (or **blocks**, like **for loops**), scope rules are mostly irrelevant.

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 **surprising** scope rules:

- JavaScript
- R
- LISP (the original)
- Bash
- Mathematica

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

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

Want to be a front-end developer? You should probably know these rules:

https://stackoverflow.com/a/500459/480764

Perl Examples

```
local $x = 10;

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

sub g
{
   local $x = 20;

   f();
}

g();
```

```
my $x = 10;

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

sub g
{
   my $x = 20;

   f();
}

g();
```

What is printed?
Which one is dynamic and which one is lexical?

Perl Examples

```
local $x = 10;

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

sub g
{
   local $x = 20;

   f();
}

g();
```

Dynamic scope
(`local` keyword)

```
my $x = 10;

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

sub g
{
   my $x = 20;

   f();
}

g();
```

Lexical scope
(`my` keyword)

How do they work?

(whiteboard)

Unit testing

Unit testing is a quality-assurance method designed to find bugs before software ships. A **unit test** consists of **test code** written to exercise the functionality of a **unit** of code **in isolation**. For example, in functional code, a unit is often thought of as a **module**, **function**, or **primitive** operation.

Note that unit testing is usually **not sufficient** to determine the correctness of code!

Popular Unit Test Frameworks

Java: JUnit

.NET: MsTest or NUnit

Python: unittest

Ruby: rspec or cucumber

Tons more!

`https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks`

(demo)

Recap & Next Class

This lecture:

- Scope
- Unit testing

Next lecture:

- Type inference