Announcements

• Assignment 2 due tonight at 11:55pm
  - Submit individually
• Assignment 3 out tonight
  - PL-Detective questions
    • I select the semantics of MYSTERY and you guess them.
    • You have a limited number of trials.
    • Experimentation is not a proof but gives some evidence, so explanation of your reasoning is critical

Feedback

• Please feel free to tell/e-mail me any concerns that come up
  - Recitations?
    - HW Question/Session
    - Last time is copying
    - Group exercises
    - “Extra HW”

Feedback from HW1

• Time spent: mean 1.26 hrs, median 1 hr
• Easy but first homework

Next Unit

• More “doing” projects?
  1. Start functional prog sooner
  2. Wait till end

Review
Variables

- Attributes (Sebesta)
  - Name: what it's called
  - Address: where it is stored
  - Type: possible values
  - Value: data / contents
  - Lifetime: while it exists / data available
  - Scope: when name is available (code hints)

Aliasing

int* x; int* y; ...
// have cells pointed to by x and y

Example: Static and Dynamic Typing

- static: type bound to variable at compile-time
- dynamic: type bound to variable at run-time

Example: Static and Dynamic Typing

x = 3;
y = x + 1;
x = [7, 9, 2];
y = x + 1

Ok with static or dynamic typing?
Example: Static and Dynamic Typing

- Static typing includes dynamic typing
- With this view, what is wrong with dynamic typing?
- Dynamic typing forces the overhead of tag checking always

Example: Static and Dynamic Typing

- Can we have
  \[ x = 3; x = [10.2, 5, "bob"]; \]

Type Inference (ML)

- Static and dynamic typing is orthogonal to explicit declarations

Storage Binding

- Storage binding determines lifetime
  - Why?
  - Kinds of storage binding
    - static
    - stack dynamic
    - explicit heap dynamic
    - implicit heap dynamic

Storage Binding: Static

- What is it? Example? Advantages?
  - Set memory address throughout execution
    - Example: globals
    - may be more efficient (save space)
    - history sensitive (for each defining)
    - can’t access \( x \) - scope
    - void \( g() \) 

\[ \text{fun times} 10: x = 10.0 * x \]
\[ \text{fun times}(x::\text{int}) :: \text{int} = 10.0 * x \]
\[ \text{fun times}(x::\text{int}, y::\text{int}) :: \text{int} = \text{times}(\text{int}, \text{int}, \text{int}) \]
\[ \text{fun square}(x::\text{real}) :: \text{real} = x * x \]
\[ \text{fun square}(x::\text{real}) :: \text{real} = x * x \]
Storage Binding: Stack Dynamic

• What is it? Example? Advantages?

Suppose function $f$ has local $x$, $g$ has $y$, and $h$ has $z$. Function $f$ calls $g$ and $g$ calls $h$. What is the lifetime of each variable?

\[ f(x) \rightarrow g(y) \rightarrow h(z) \]

Storage Binding: Explicit Heap Dynamic

• What is it? Example? Advantages?

Storage Binding: Implicit Heap Dynamic

• What is it? Example? Advantages?

When are things freed?

Automatic deallocation

⇒ using garbage collection
End of Binding
On to Scope

Static scoping: Let’s extend arithmetic with a binding construct

\[ e ::= n \mid e_1 + e_2 \mid e_1 * e_2 \mid \text{let } x = e_1 \text{ in } e_2 \mid x \]

- **let** binds a *new* variable \( x \) to \( e_1 \)
  for use in \( e_2 \)
  - variable \( x \) is *bound* in the body \( e_2 \)
  - the *scope* of \( x \) is the body \( e_2 \)

What is meant by “new”?

- Every instance of **let** binds a variable that uniquely identified with it.

Example expressions

\[ \text{let } x = 1 \text{ in } (x + x) \]
\[ \text{let } x = 2 \text{ in } (x + x + x) \]
\[ \text{let } x = 3 \text{ in } (x + x + x + x) \]

How do resolve variables?

- **Lexical scope rule:**
  - A variable is bound by the nearest enclosing binding
  - Operationally, how?

Preview

- What is scope?
  - *“where in source code text can you access a variable?”*

- What is static scoping?
  - Determine which statements can access by looking at the source code

- What is dynamic scoping?
  - Determine the calling sequence of functions
Can we do a program transformation to make it clear where a variable is bound?

- How?
  - Rename bound variables
    - Bound names don't matter

Rename bound variables

- We identify terms up to renaming ($\alpha$-conversion)

Higher-Order Abstract Syntax:
All terms that differ only in names of bound variables are considered equivalent.

Examples

Dynamic Scoping

- "Scope depends on the calling sequence of functions"

- Within a function, is there any difference with static scoping?

Exercise: Which $x$ is used?

PROCEDURE f() =
  VAR x : INTEGER;
PROCEDURE g() =
  VAR x : INTEGER; BEGIN ... END;
PROCEDURE h() = BEGIN ... END;

BEGIN
  h(); g();
END

BEGIN
  g(); h();
END
Summary: Dynamic Scoping

• No variable is local making programs difficult to debug
  - e.g., in LISP, TCL, TeX

• Generally, considered a bug in PL implementation
  - (but watch out, very easy to do!)

For Next Time

• Reading
• Online discussion forum
  - ≥1 substantive question, comment, or answer each week
• Homework assignment 3
  - Start early!