CS 430
Spring 2015

Mike Lam, Professor

Variables

Variables

- What is a variable?

Variables

- Variable: an abstraction of memory cells
  - Most languages have variables
  - However, they are NOT essential for computation!
- Six main attributes/properties:
  - Name
  - Address
  - Value
  - Type
  - Lifetime
  - Scope

Binding

- *Binding*: association between an attribute and an entity
  - Begins at *binding time*
  - *Static* bindings begin before the program is executed and do not change during execution
  - *Dynamic* bindings may begin or change during execution
- We will discuss bindings for five of the major variable attributes
  - Lifetime bindings are usually a combination of other attributes

Name

- Name – string of characters that serves as an identifier
  - Case sensitive?
  - Reserved words?
  - Special characters with meanings? ($ and @ in Ruby)
  - Standards or conventions?
- Keyword vs. reserved word
  - Keyword: string of characters with special meaning
  - Reserved word: string of characters that cannot be used as a variable name
- Name bindings are usually static
  - Do all variables have a name?

Address

- Address: location of a variable
  - Sometimes called *l-value*
- *Aliases*: multiple variables with an identical address
- Address bindings may be static or dynamic

Value

- Value: contents of the memory associated with a variable
  - Sometimes called *r-value*
- Value bindings are usually dynamic
  - Otherwise, they wouldn't be "variable"
  - Important exception: *constants*

Type

- Type: range of values a variable can store
  - And the operations that can be applied to it
- Common types:
  - Integer
  - Floating-point
  - Character-based
  - Union or composite structure (*object*)
- Implicit vs. explicit
- Static vs. dynamic
  - "Duck" typing
- Type inference

Lifetime

- Lifetime: duration of address binding
- Common lifetimes:
  - Static
  - Stack dynamic
  - Explicit heap dynamic
  - Implicit heap dynamic
- Allocation: explicit or implicit?
- Deallocation: explicit or implicit?

Scope

- Scope: program range where a variable is visible
- Local vs. global
- Static vs. dynamic
  - Code structure vs. call structure
  - Dynamic scoping is very rare (Perl example)
- Impact of block structures
  - Ancestor scopes
- Often related to lifetime

Referencing Environment

- Referencing environment: all variables and constructs visible from a program statement
  - Local scope plus ancestor scopes

Case studies

- Cases
  - Ruby class instance variables
  - Java "static final" class variables
  - C++ "new"-allocated object
  - C loop index variable
- Questions
  - What is the name, address, value, type, lifetime, and scope? Are they static or dynamic?