Names

A name is a series of characters used to denote an entity in a program, e.g. variable, type, function, and etc..

/**
 * File: power.c
 * Author: Ying Li
 * Date: 07/20/2015
 */

#include <stdio.h>

/* power: increase base to nth power; n >= 0 */
int power (int base, int n) {
    int i, power;
    power = 1;
    for (i = 1; i <= n; i++) {
        power = power * base;
    }
    return power;
}

/* test power function */
int main (int arg, char *argv[]) {
    int i;
    for (i = 0; i < 10; i++) {
        printf("%d %d %d \n", i, power(2, i), power(-3, i));
    }
    return 0;
}
Syntactic Issues

✦ Name ↔ Identifier: determined by lexical rules

• Case sensitivity

<table>
<thead>
<tr>
<th>Case-insensitive</th>
<th>Pascal, Ada, SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-Sensitive</td>
<td>C, Java, Python</td>
</tr>
<tr>
<td>Mixed case sensitivity</td>
<td>PHP</td>
</tr>
</tbody>
</table>

• Use of special characters

<table>
<thead>
<tr>
<th>Underscore ( _)</th>
<th>C, Java, Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyphen (-)</td>
<td>Cobol</td>
</tr>
</tbody>
</table>

✦ Predefined identifiers

• Reserved: keywords/reserved words

• Allowed to redefine: tradeoff between the size of keywords and readability
Semantic Issues

- **Binding**: an association between an entity and a symbol
  - *Static binding*: a binding is *static* if the association occurs before run-time
  - *Dynamic binding*: a binding is *dynamic* if the association occurs at run-time

- **Visibility**: redeclaration within a scope of a name may hide other instances of the same name
  - *Overloading*: permits different instances of the same function or operator name to be resolved based on *different signatures* (the number or types of its arguments)

- **Lifetime**: the lifetime of a variable name refers to the time interval during which the variable is allocated memory
Variables

A **variable** is a *binding* of a **name** to a **memory address**.

- Attributes of variables: type, value, lifetime, and scope.
- Binding a name to a memory address requires four pieces of information:
  - Name
  - Address: uniquely identifies the actual memory location
  - Type: even without explicit variable types, compiler or interpreter must internally handle typing
  - Value