Symbol Tables --

- Stand alone symbol tables
- Part of AST?
- Keeps track of names, but need a way to talk about types

Name and Attribute "database" (Dictionary?)
- basis for a lot of semantic information
- "interface"
  - create (st)
  - enter (st, name, attributes)
  - STE = lookup (st, name)
  - destroy (st)

- Implementations
  - Linked list, array (unordered)
  - ordered list
  - binary search tree
  - hash table
  - C++ STL map ...

Block Structured Symbol Tables

Languages: Algol, Pascal, Modula-2, ATL/1

Program A;
  type Y = integer;

  Procedure P (X : Y);
  Var Y : Boolean;
  Begin
    ....
  End P;

begin
  ....
End A.

Scope 1: Y is type integer
Scope 2: Y is variable of type boolean
Procedure P’s declaration? Y a type or a variable?
(Most languages consider it to be a type here.)
Issues in block structured symbol tables

Scope: multiple, enclosed scopes. (Depth max?)

Current Scope, What causes scope
- procedure
- functions
- structs / records
- { .... } in C
- Explicit namespace constructs

Identifier Search Rules
- scope <-- current scope
- while scope is legal do
  - x = lookup (scope, name)
  - if (name is found) break;
  - scope <-- next scope out
Implementation for ATL

Global table: array of tables, one for each level
One kv_tree for each level

-1: "Global"
0: Program
1: procedure level 1
2: procedure level 2
3...

Maintain "current_level" variable (Maximum level -- 15)

One possible interface: (More complete one later)
- type = make_type (kind, attributes): creates a "type descriptor"
- id = make_id (name, attributes ....): Creates an id record
- new_scope() -- Increments the current_level, initializes new kv_tree
- end_scope() -- decrements the current_level, possibly cleans up
- int Enter_id(id): Enters the id in the current level’s kv_tree, may generate an error
- id = Find_id(name): Uses proper search rules to find an name in the scopes.
- Debugging: print_type(type), print_id(id) -- should write these as same time
Extra symbol tables

- Type descriptor -- describes a type
  - basic: integer, float, boolean / may need a size (not for hc)
  - array: low and high and element type (1 dim) / size of array

Type rec = Record
  a, b : real;
  c, d : integer;
  rec : char;
end;

- Symbol table as part of record type descriptor
  - mixing of type descriptors and id records
  - use this scope only when working with this record (not in ATL/1!)
Import/Export in Symbol Tables

Ada packages, C++ namespaces, Modula 2 modules, ATL/2, ...

- Import into current scope?
- How to get imported symbols?
- Separate compilation?
  - Representation
  - Importing/Exporting only part of the names?

Altered Search Rules
- With statement (pascal, modula-2)