Symbol tables

A symbol table associates values or attributes (e.g., types and values) with names.

What should be in a symbol table?
- variable and procedure names
- literal constants and strings

What information might compiler need?
- textual name
- data type
- declaring procedure
- lexical level of declaration
- if array, number and size of dimensions
- if procedure, number and type of parameters

Two major functions for a symbol table:

\[
\text{addentry(ID, type, other attributes...)} \rightarrow \text{access-code}
\]

\[
\text{lookupentry(access-code)} \rightarrow \{\text{attributes}\}
\]

Implementation

Major goals are (1) small table size and (2) quick retrieval of attributes

Implementation strategies:

- **Direct search.** Each name has a unique address. (e.g., original Basic. Names limited to under 300: A$, A, A0..A9 for each letter.)
  - Fast retrieval, but limited to small table size
- **Linear search.** Add names in order, and search in order - Slow retrieval (N=2 on average), Small table size, Easy to program.
- **Binary search.** Names are sorted in order - Search time O(log n), small table size, but addentry operation slow.
- **Hash search.** Create function of name that mimics direct search. For a good function, each name has a retrieval time of 1 and table can be relatively small. Works best with table size under 2N.
**Hash table collisions**

- Typical hash functions: Table size is a prime and divide name (as an integer) by table size. Remainder of 0 to Tablesize-1 is location in table containing entry.
- Problem with hash tables are when hashing algorithm is not perfect (as it must never be): collisions.
- Issues:
  - Same hash function must be used for all program
  - Possible name space much larger than table size
  - Two names can hash to same value
- Collisions:
  - After search see if found correct entry. If not, resolve collision
  - Storage strategies: Move to next item, Move k away, Have hash bucket or hash chain.

**Scope issues**

How to handle nested lexical scoping?
- When we ask about a name, we want the closest lexical declaration

One solution
- Use one symbol table per scope
- Tables chained to enclosing scopes
- Insert names in table for current scope
- Name lookup starts in current table if needed, checks enclosing scopes in order

Another solution
- Have one symbol table
- Include scope information as part of name (e.g., X declared in procedure P stored as P.X)
- Look up names by first checking for LocalProcedure.X then EnclosingProcedure.X ...
- There are many variations of this approach