

**DEFINITION**

- Global **name table** (= **spelling table** = **identifier table** = **lexeme table**) used to convert scanned identifier names into numeric references.
- Token stores reference to number in addition to or instead of string.
- ASTIdentifier structure will only be interested in number.
- Purpose:
  - Easily identify references to same identifier.
  - Minimize string comparisons to once throughout entire process.

**COST**

- Cost matters for interpreters
- Cost calculation:
 

Assume table of  $n$  records,  $m$  enquiries:

i.e. there are  $n$  names in program and they are used  $m$  times altogether

i.e.  $n$  additions,  $m$  enquiries  $m > n$

  - $S$  = average cost of 1 search
  - $A$  = cost of 1 addition
  - Cost =  $n A + n S + m S$   
(Each addition involves 1 search for duplicates.)

**POSSIBLE ORGANISATIONS****Linear List in Chronological Order (default)**

- Keep an table (array) of records, with pointer to last
  - Add to the end
  - Search from the end to beginning
- Cost
  - $S = n/2$  records when record is found
  - $S = n$  when record is not found (for additions)
  - $A = \text{constant } C$
  - Cost =  $nC + n^2 + m.n/2 = O(n(n+m)) = O(n.m)$

**Self-Organising List**

- In addition to linear list, provide linked list of table indices which moves last used index to front of list
- Costs are same order, but
  - Additional moving costs (constant with linked lists)
  - Real programs cluster usage of identifiers => real search savings

**Binary Search Tree in Alphabetical order**

- In addition to linear list, provide BST of table indices organizing names in alphabetical order
- Cost
  - $S = \log n$
  - $A = \log n$
  - $\text{Cost} = O((n+m) \log n) = O(m \log n)$   
In practise, useful if  $n > 50$

**Hash Table**

- In addition to linear list, provide hash table to organize indices (hash on string)
- Cost: Assume  $k =$  size of hash table
  - $S = O(n/k)$
  - $A = O(n/k)$
  - $\text{Cost} = O((n+m)n/k) = O(m.n/k)$
- To reduce Cost, make  $k$  big (around  $O(m)=100$  is good)