One Global Symbol Table for all Scopes

```
var a, b:int;

function main () {
   const a:real . . .
   {  var a, b:char;
      . . .
   }
   var b:real;
   . . .
}

function f (b:char):int {
   {  var c:int;
      . . .   ← Reached Here
   }
   var a:char;
   . . .
}
```

- ❖ Scope is recorded in each symbol
  - ✸ Distinguishes between symbols with same name
- ❖ Scope table keeps track of open scopes

Scope Table

level	2
[0]	1
[1]	2
[2]	2
[3]	0

	scope			
op	level	count	name	type
V	2	2	c	INT
A	1	2	b	CHAR
F	0	1	f	INT
V	1	1	b	REAL
V	2	1	b	CHAR
V	2	1	a	CHAR
C	1	1	a	REAL
F	0	1	main	NULL
V	0	1	b	INT
V	0	1	a	INT

Scope and Scope Table

- Scope: recorded in every symbol and consists of
  - Level: nested level of scope
  - Count: count of scope at a given level
- Scope Table:
  - Records scope counts at all levels
  - Which scopes are currently open
- Open scope: upon entry
  - Increment level number
  - Increment level count
- Close scope: upon exit
  - Decrement level number
  - Do not modify level count

Scope Table

level	2
[0]	1
[1]	2
[2]	2
[3]	0

	scope			
op	level	count	name	type
V	2	2	c	INT
A	1	2	b	CHAR
F	0	1	f	INT
V	1	1	b	REAL
V	2	1	b	CHAR
V	2	1	a	CHAR
C	1	1	a	REAL
F	0	1	main	NULL
V	0	1	b	INT
V	0	1	a	INT

Insert and Lookup a Symbol

- Insert:
  - Symbols are inserted in the order they appear in the source file
  - An inserted symbol is placed on top of the symbol stack
    - Symbol stack can be implementation using array or linked implementation
  - Store current level and level count in symbol
- Lookup:
  - Search a table for a given name
    - Enough to compare name pointers if name has a unique pointer
  - Must be in closest open scope
    - Symbol level ≤ current level
    - Level count in symbol must match the count in scope table
  - Return a pointer to found symbol

Scope Table

level	2
[0]	1
[1]	2
[2]	2
[3]	0

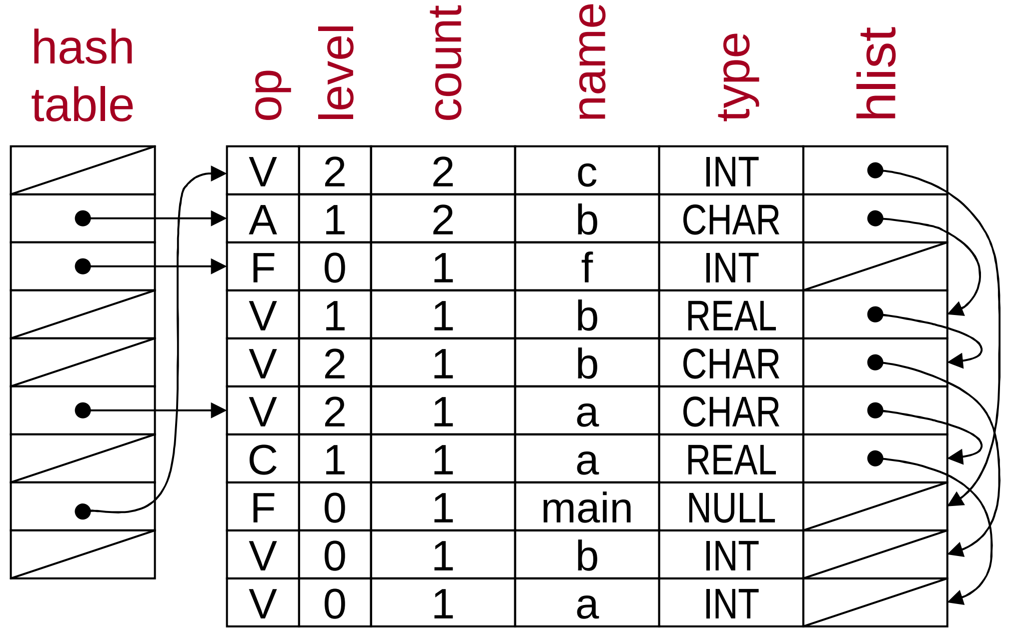
	scope			
op	level	count	name	type
V	2	2	c	INT
A	1	2	b	CHAR
F	0	1	f	INT
V	1	1	b	REAL
V	2	1	b	CHAR
V	2	1	a	CHAR
C	1	1	a	REAL
F	0	1	main	NULL
V	0	1	b	INT
V	0	1	a	INT

Speeding-Up Lookup with a Hash Table

- Hash table: an array of pointers is added
- A new field, *hlist*, is added to each symbol
  - To link symbols hashed to the same hash table index
- A name is hashed before insertion and before lookup
  - Enough to hash name pointer if name pointer is unique
- Insert:
  - At front of hash list
- Lookup:
  - Traverse one hash list
  - Return first match for a name in an open scope
  - Bypass symbols in closed scopes, or remove them from their hash list only

Scope Table

level	2
[0]	1
[1]	2
[2]	2
[3]	0



op	level	count	name	type	hlist
V	2	2	c	INT	●
A	1	2	b	CHAR	●
F	0	1	f	INT	
V	1	1	b	REAL	●
V	2	1	b	CHAR	●
V	2	1	a	CHAR	●
C	1	1	a	REAL	●
F	0	1	main	NULL	
V	0	1	b	INT	
V	0	1	a	INT	