`Micro` Overview and Symbol Tables

CS316 Spring 2022

Example Micro Program

- Refer to the grammar in PA2 to know the programming constructs fully.

  - MicroProgram

Beyond Syntactic Analysis

- Until now:
  - `INT x:=2;` ≡ `INT x:=5;`
  - `x:=y+100` ≡ `x:=y+1000000000000000;`
  - `INT x, y;`

    ```
    y := (2.3 * 6);        ✓
    x := ( main );         ✓
    ```

- Now on:

  ...toward meaningful, executable programs

Symbol Table

- A *symbol table* maintains
  - Symbolic names
  - Attributes of a name
    - E.g. type, scope, accessibility
- Used to manage declarations of symbols and their correct usage

Symbol Table – Names

For the sample program shown below identify all names (note: this is not a valid micro program)

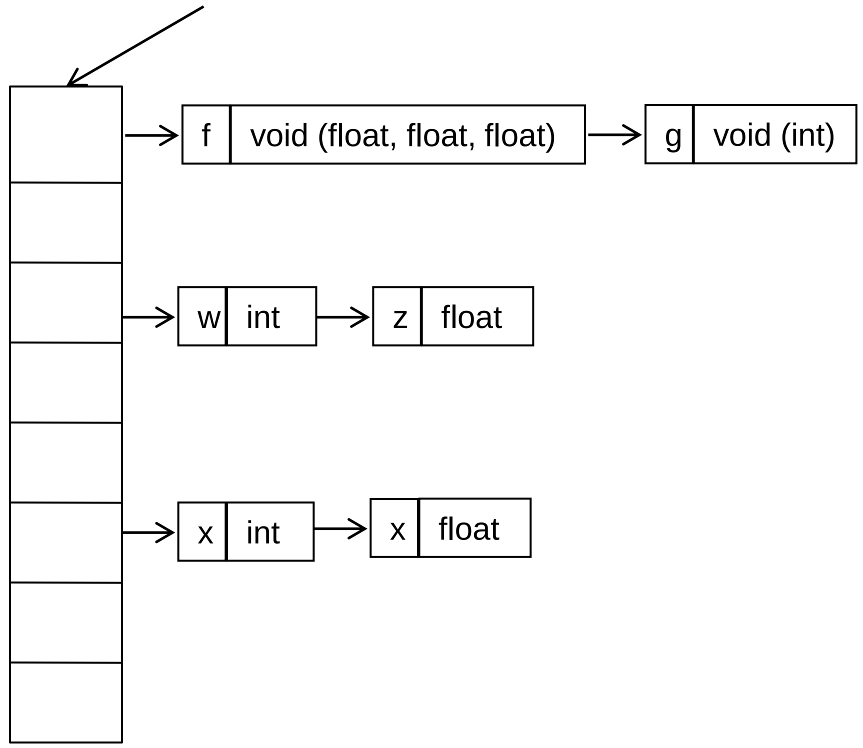
```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
    INT w, x;
    {
       FLOAT x, z;
       f(x, w, z);
    }
    g(x);
}

END
```

Symbol Table Implementation – High-level Requirements

- Should accommodate:
  - Efficient retrieval of names
  - Frequent insertion and deletion of names
- Should consider *scopes*

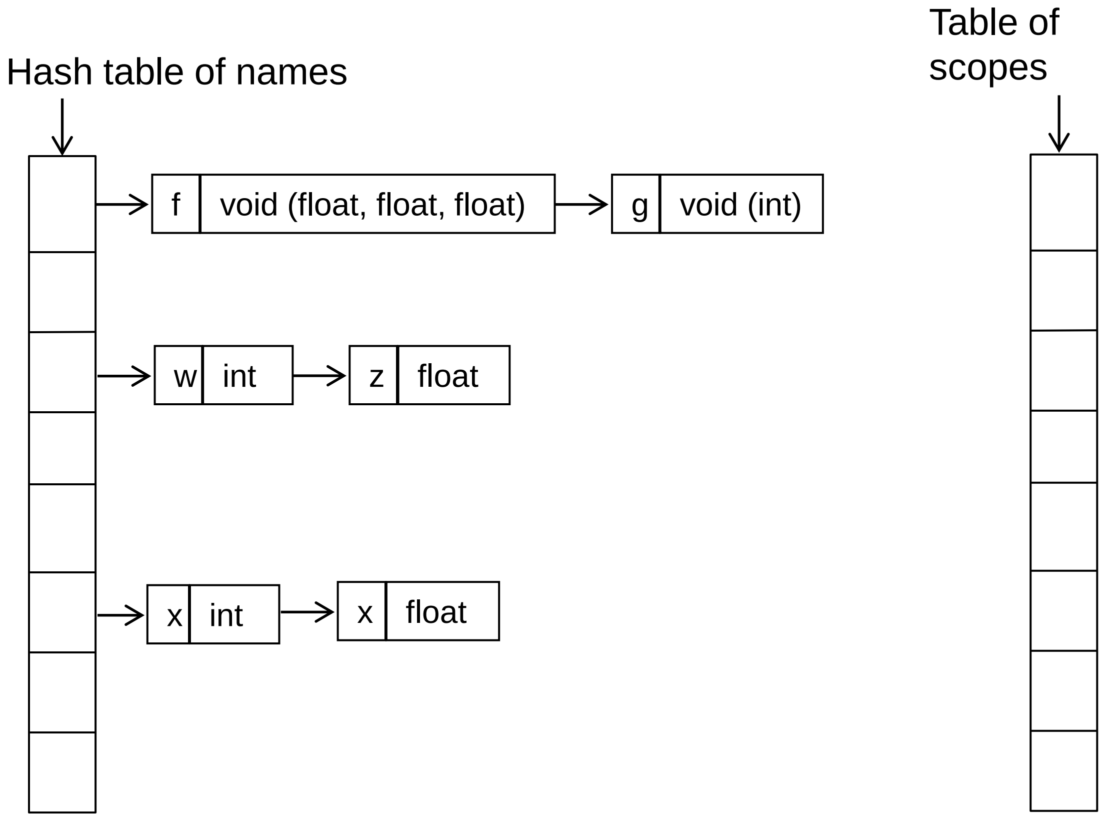
Symbol Table – an implementation

Hash table of names



```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
     INT w, x;
     {
         FLOAT x, z;
         f(x, w, z);
     }
     g(x);
}

END
```

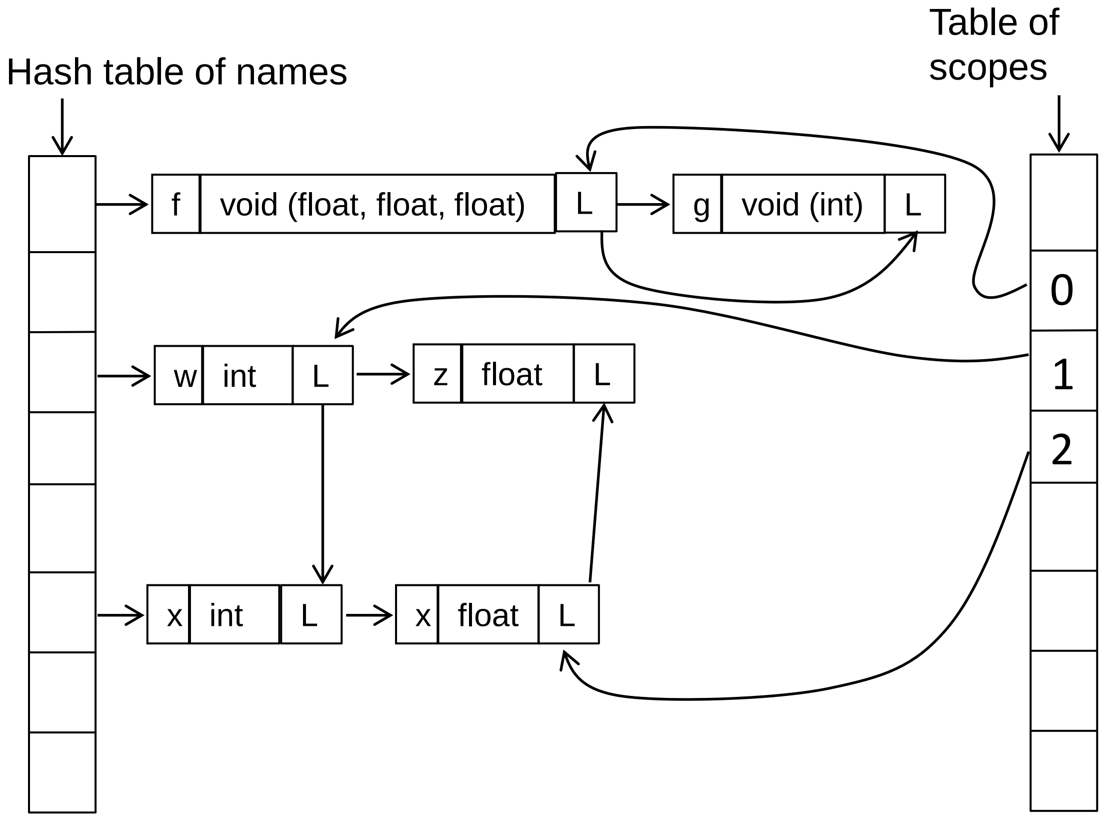
Symbol Table – an implementation



```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
        INT w, x;
        {
              FLOAT x, z;
              f(x, w, z);
        }
        g(x);
}

END
```

- be aware of current scope
- Be aware of all active scopes
- Chain names by their scope-levels

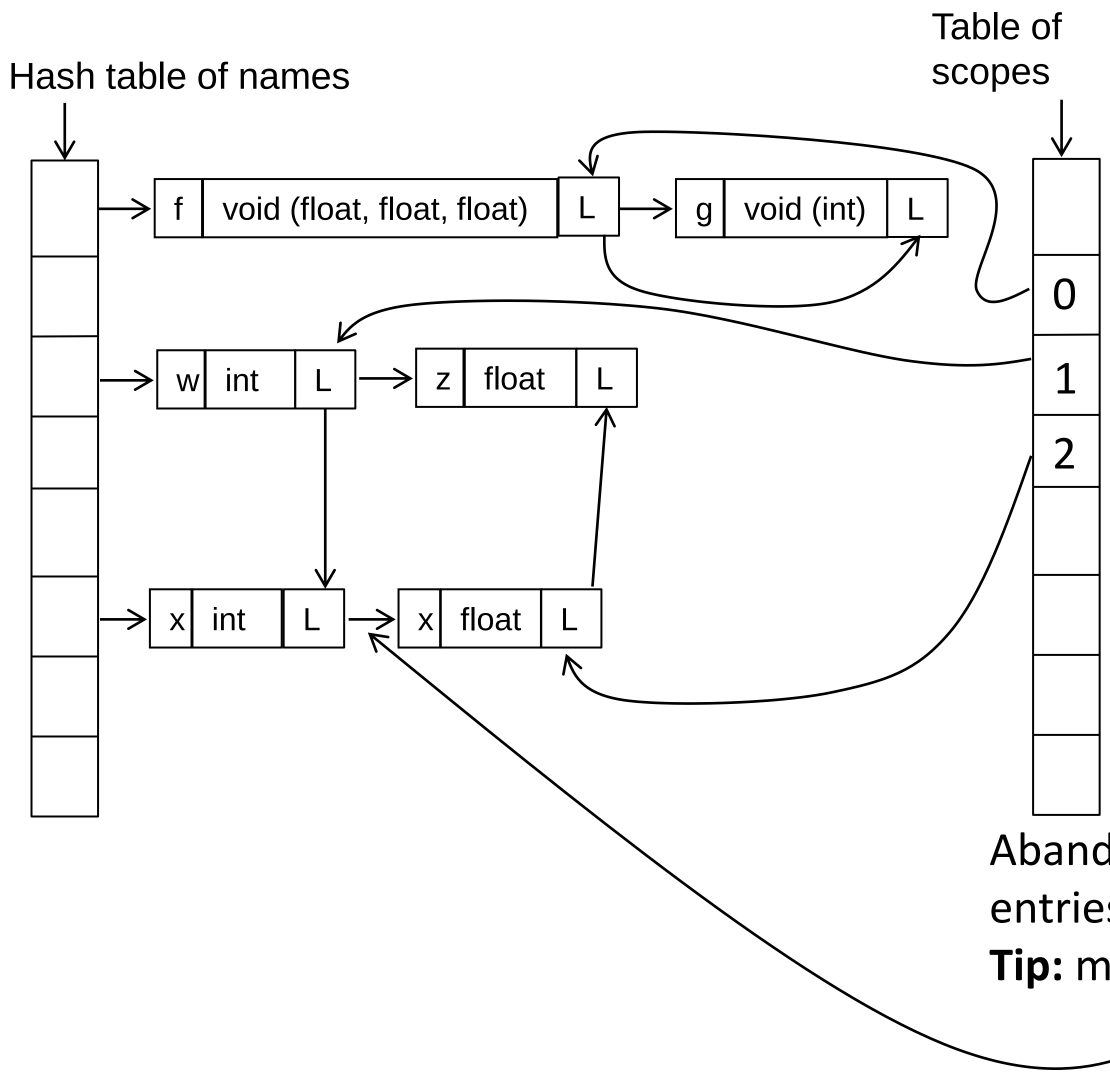
Symbol Table – an implementation



```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
        INT w, x;
        {
            FLOAT x, z;
            f(x, w, z);
        }
        g(x);
}

END
```

- Chain names by their scope-levels

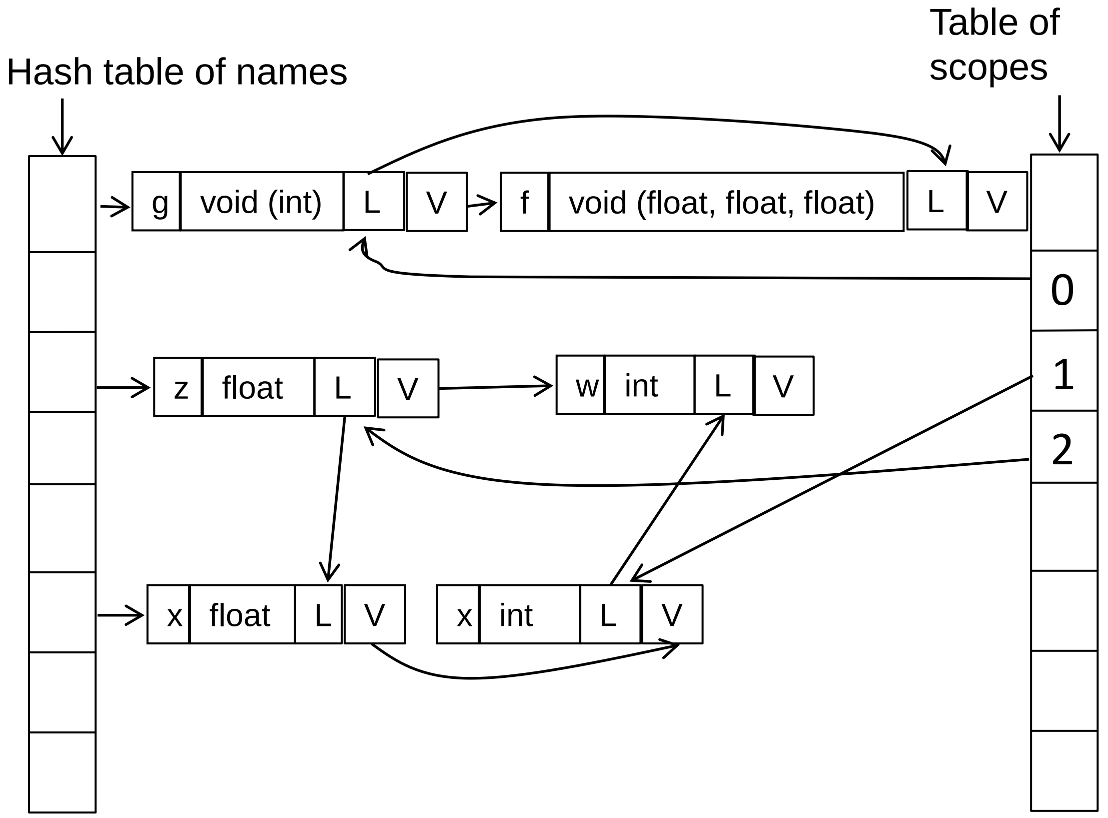
Symbol Table – an implementation



```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
     INT w, x;
     {
         FLOAT x, z;
         f(x, w, z);
     }
     g(x);
}

END
```

Abandon scope: remove references to all entries that exists at the current level.
Tip: make the hash chain a <u>doubly linked list</u>.

Symbol Table – an implementation

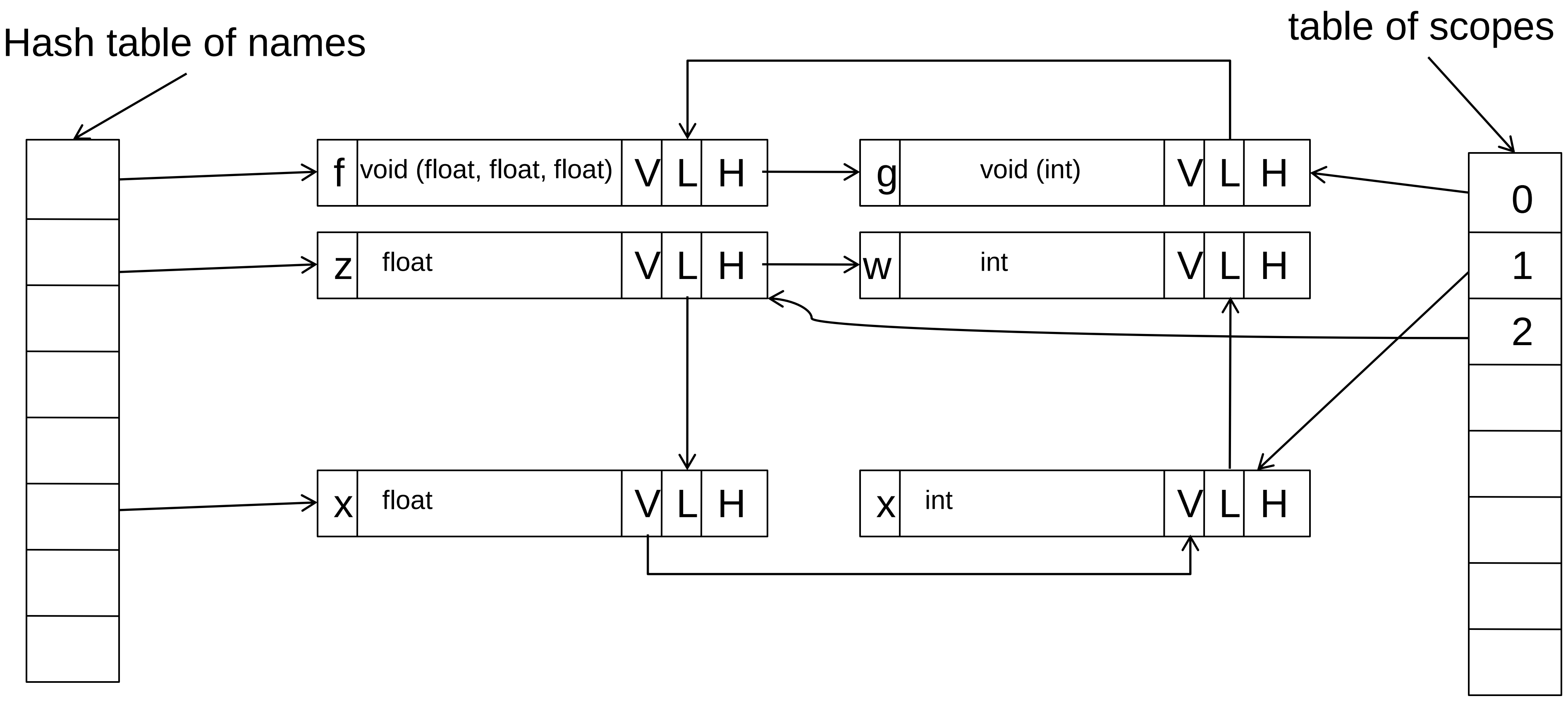


```
PROGRAM scope_test
BEGIN
#global declarations
FUNCTION void f(float, float, float)
FUNCTION void g(int)
{
        INT w, x;
        {
              FLOAT x, z;
              f(x, w, z);
        }
        g(x);
}

END
```

What if I want to access the integer x here?
Tip: maintain an ordered stack for each symbol name appearing in the current scope.

Notice the order of objects: “insert at the front of the list”

Symbol Table – an implementation


Hash table of names
table of scopes
f
void (float, float, float)
V L H
g
void (int)
V L H
z
float
V L H
w
int
V L H
x
float
V L H
x
int
V L H
0
1
2