

# Symbol Tables and Scope Checking

# Symbol Tables

- A **symbol table** is a mapping from a name to the thing that name refers to.
- As we run our semantic analysis, continuously update the symbol table with information about what is in scope.
- Questions:
  - What does this look like in practice?
  - What operations need to be defined on it?
  - How do we implement it?

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y)  {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y)  {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x, y, z);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:     {
9:         int y = x;
10:        {
11:            printf("%d,%d,%d\n", x, y, z);
12:        }
13:        printf("%d,%d,%d\n", x, y, z);
14:    }
15:    printf("%d,%d,%d\n", x, y, z);
16: }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x, y, z);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d,%d,%d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d,%d,%d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d,%d,%d\n", x, y, z);
14:        }
15:        printf("%d,%d,%d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x, y, z);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x, y, z);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5
y	9

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x, y, z);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2
x	5
z	5

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1
x	2
y	2

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1

# Symbol Tables: The Intuition

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:     printf("%d, %d, %d\n", x@2, y@2, z@1);
4:     {
5:         int x, z;
6:         z = y@2;
7:         x = z@5;
8:         {
9:             int y = x@5;
10:            {
11:                printf("%d, %d, %d\n", x@5, y@9, z@5);
12:            }
13:            printf("%d, %d, %d\n", x@5, y@9, z@5);
14:        }
15:        printf("%d, %d, %d\n", x@5, y@2, z@5);
16:    }
17: }
```

Symbol Table	
x	0
z	1

# Symbol Table Operations

- Typically implemented as a **stack of tables**.
- Each table corresponds to a particular scope.
- Stack allows for easy “enter” and “exit” operations.
- Symbol table operations are
  - **Push scope**: Enter a new scope.
  - **Pop scope**: Leave a scope, discarding all declarations in it.
  - **Insert symbol**: Add a new entry to the current scope.
  - **Lookup symbol**: Find what a name corresponds to.

# Using a Symbol Table

- To process a portion of the program that creates a scope (block statements, function calls, classes, etc.)
  - Enter a new scope.
  - Add all variable declarations to the symbol table.
  - Process the body of the block/function/class.
  - Exit the scope.
- Much of semantic analysis is defined in terms of recursive AST traversals like this.

# Another View of Symbol Tables

# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```

# Another View of Symbol Tables

Root Scope

```
0: int x;
1: int y;
2: int MyFunction(int x, int y)
3: {
4:     int w, z;
5:     {
6:         int y;
7:     }
8:     {
9:         int w;
10:    }
11: }
```

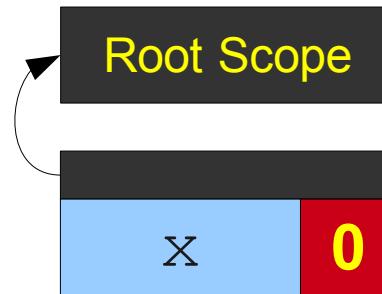
# Another View of Symbol Tables

Root Scope

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```

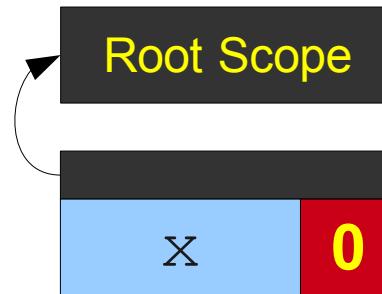
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



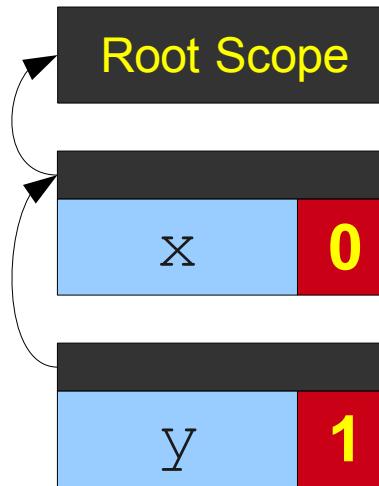
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



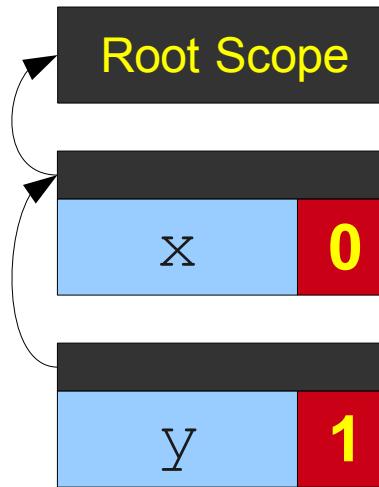
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



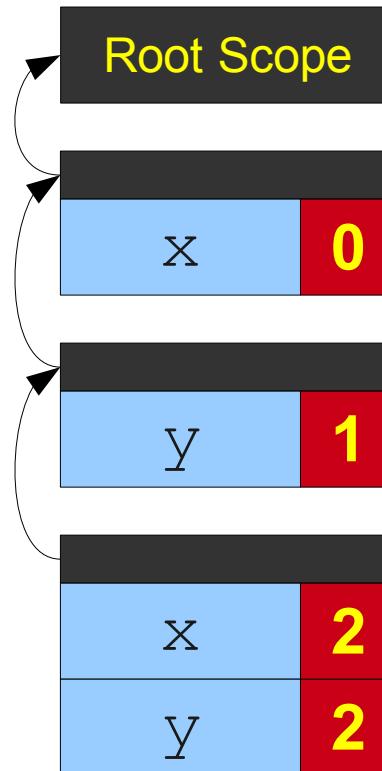
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



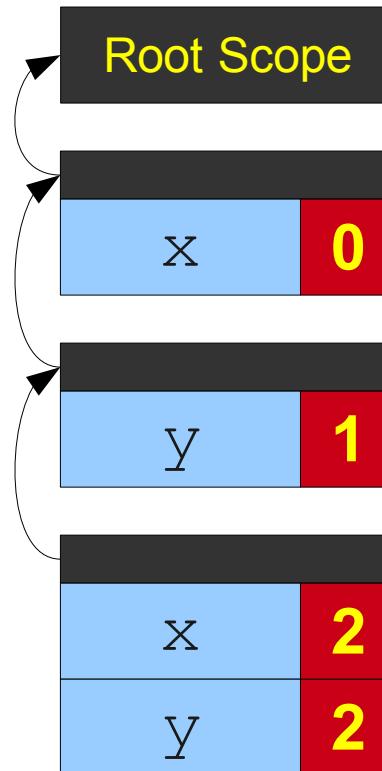
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



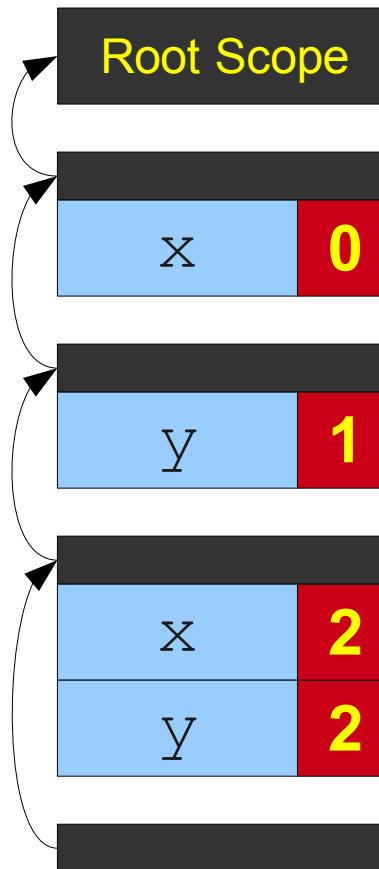
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



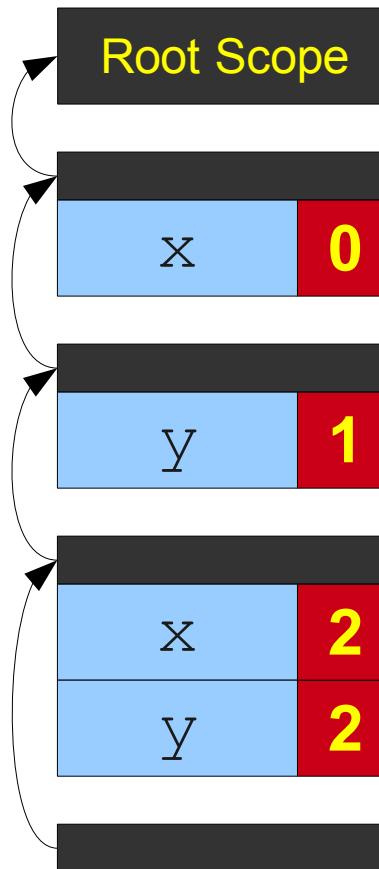
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



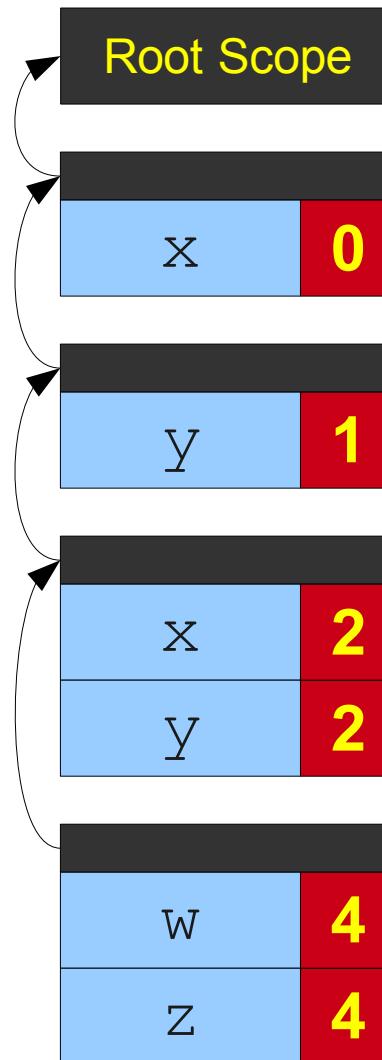
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



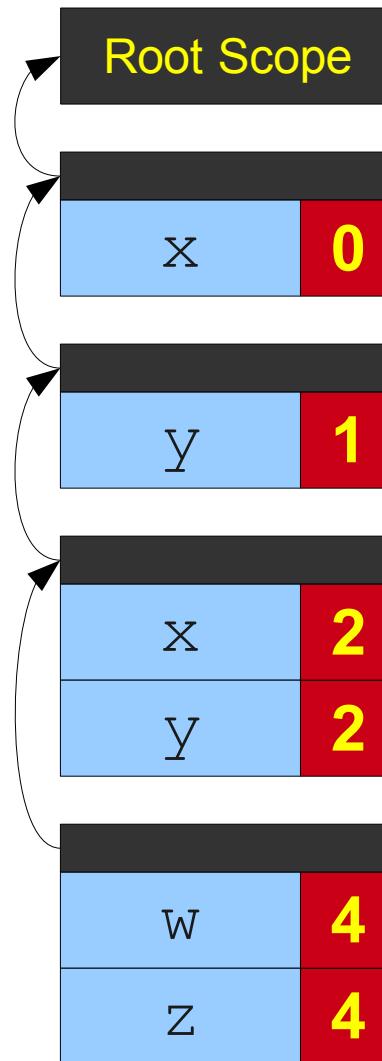
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



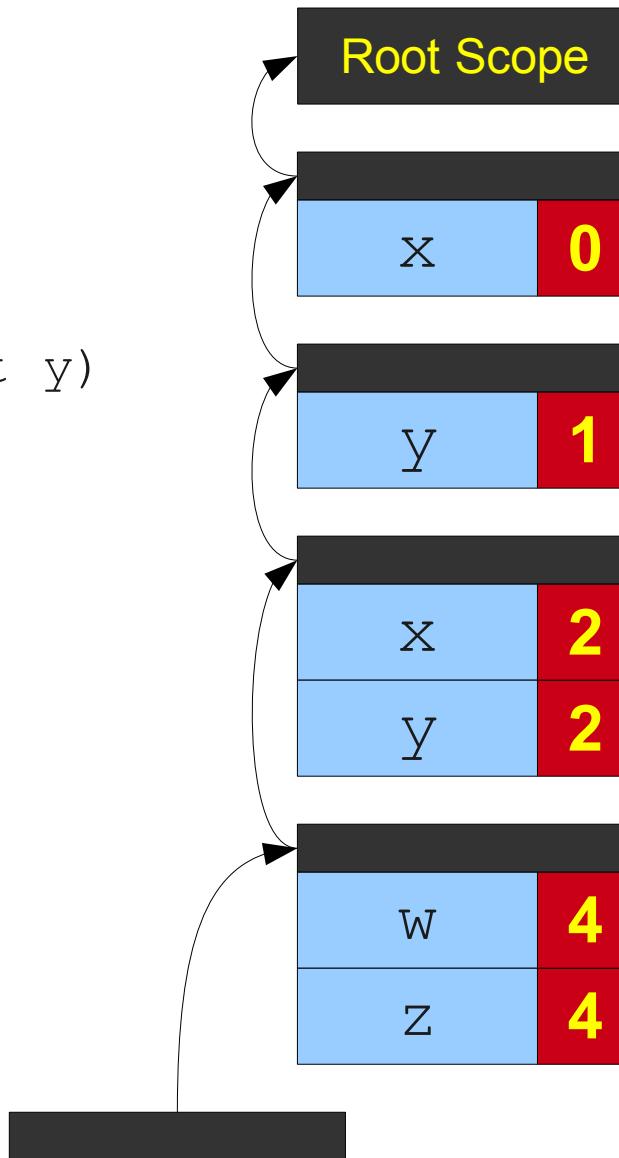
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



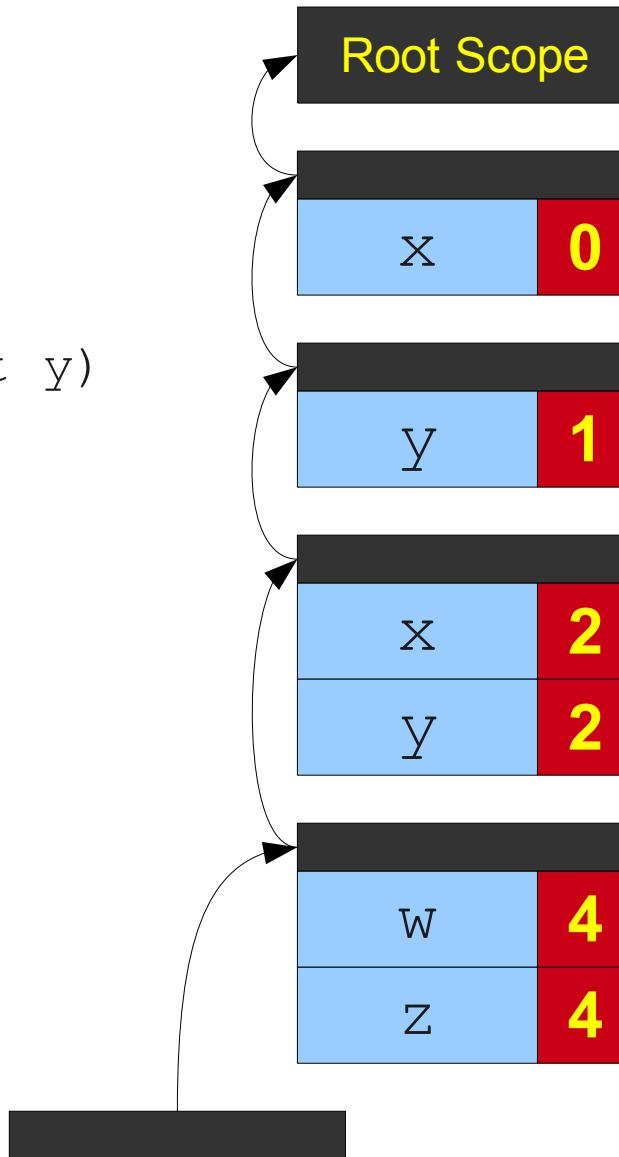
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



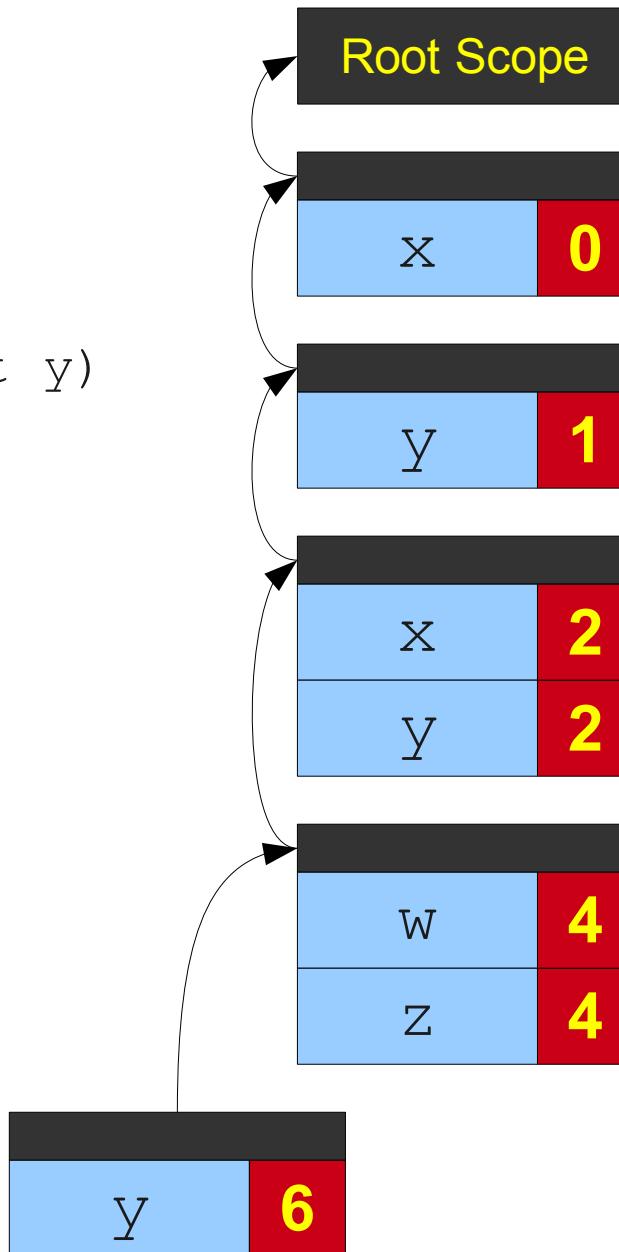
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



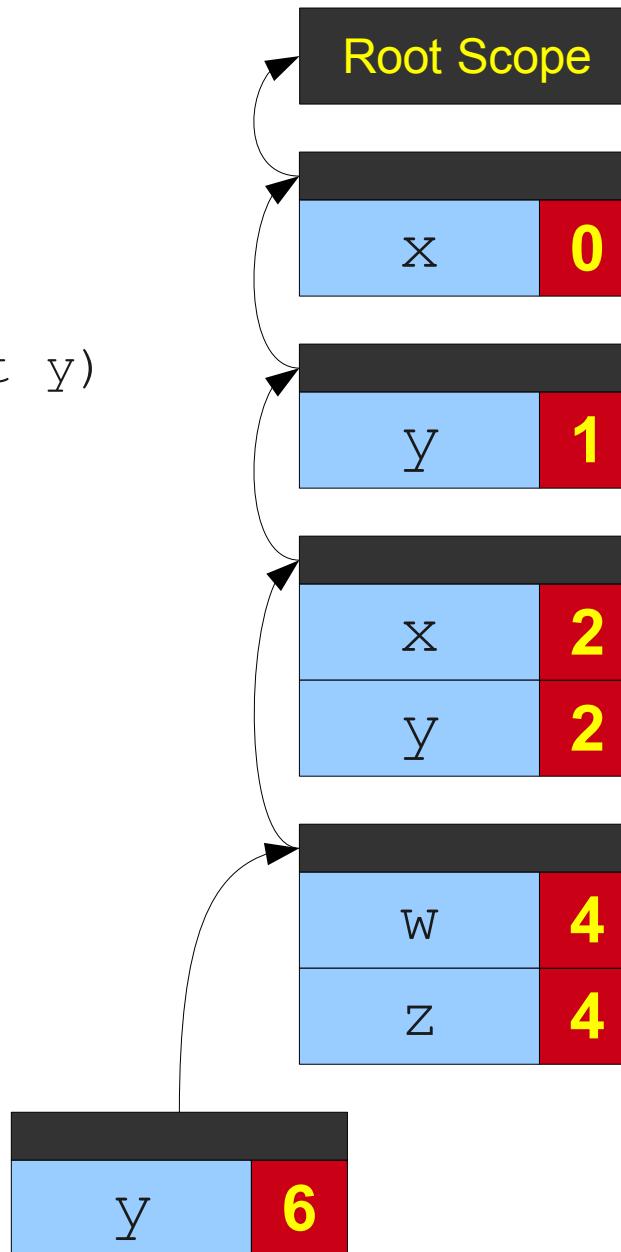
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



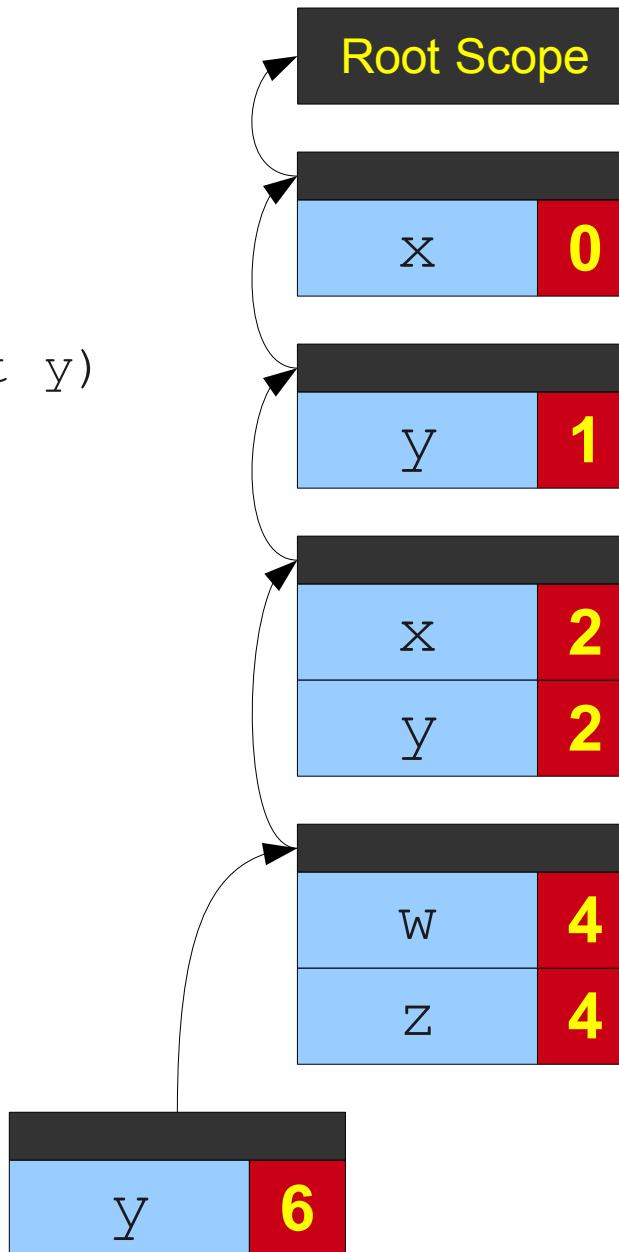
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



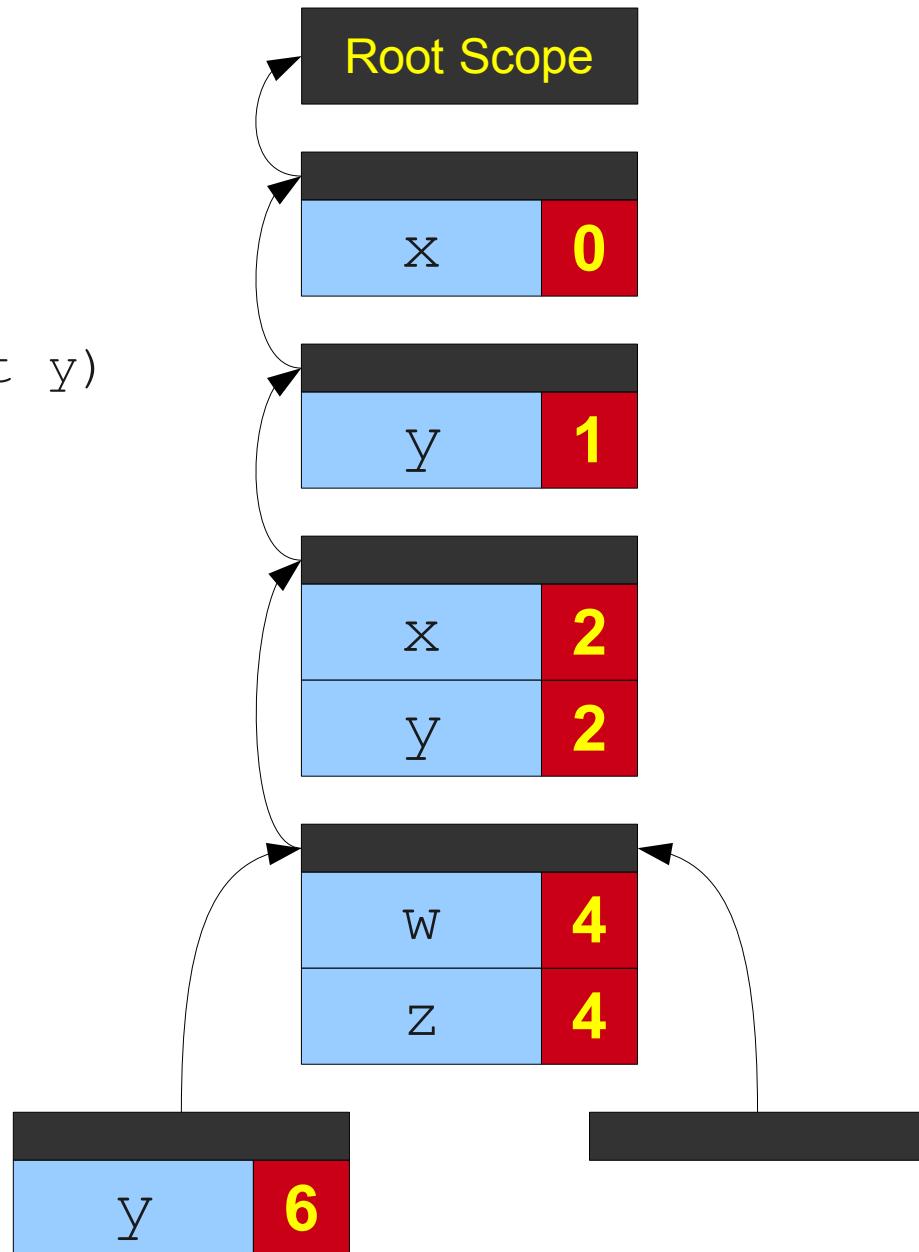
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



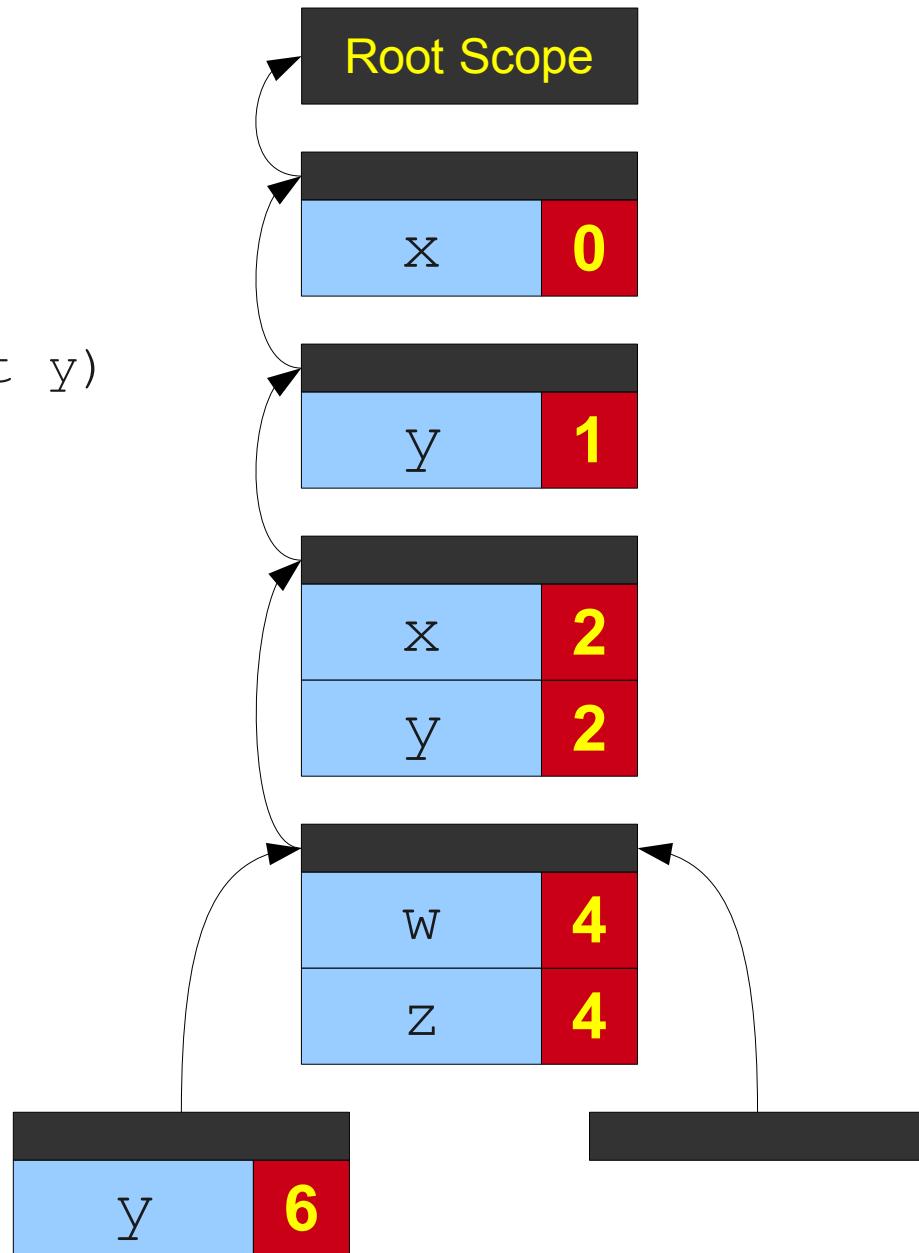
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



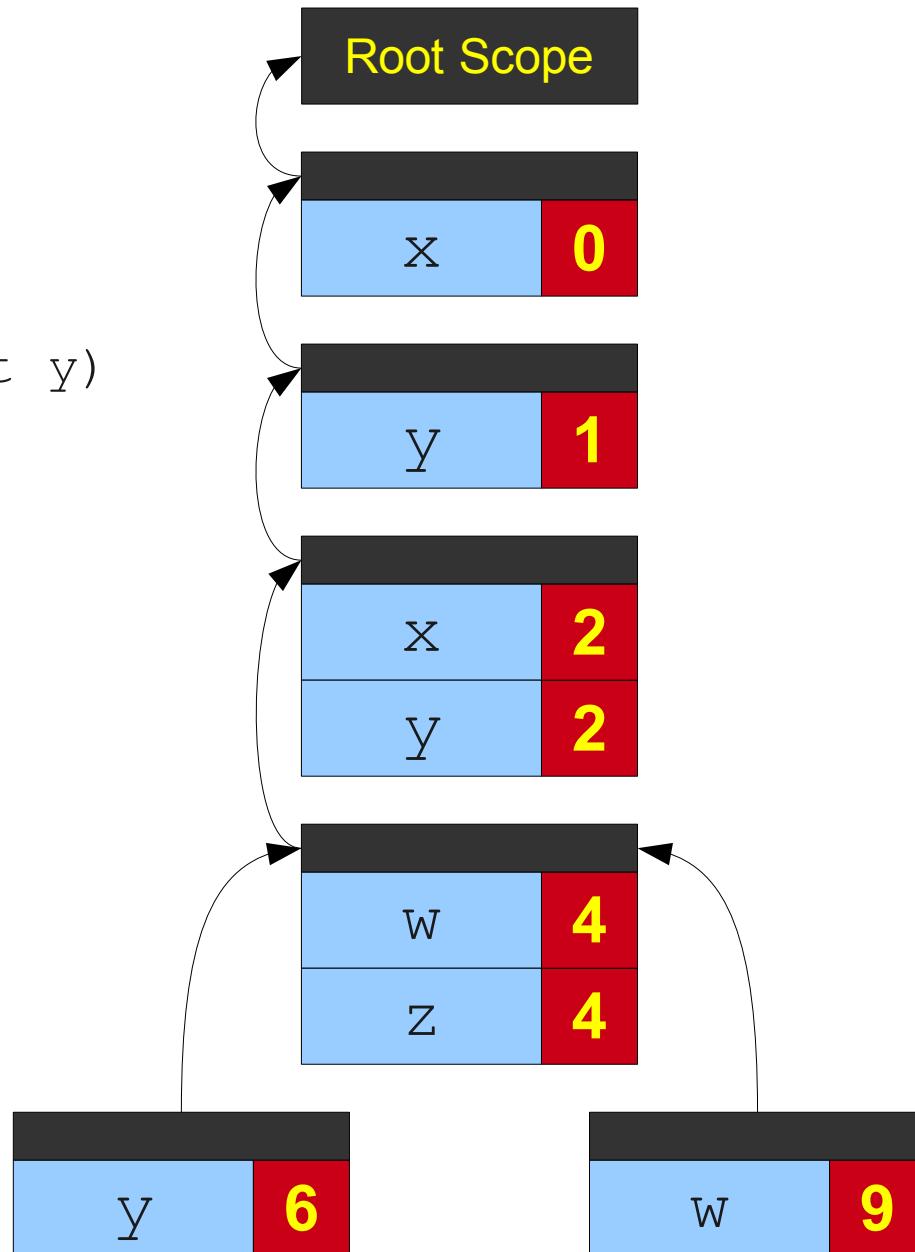
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



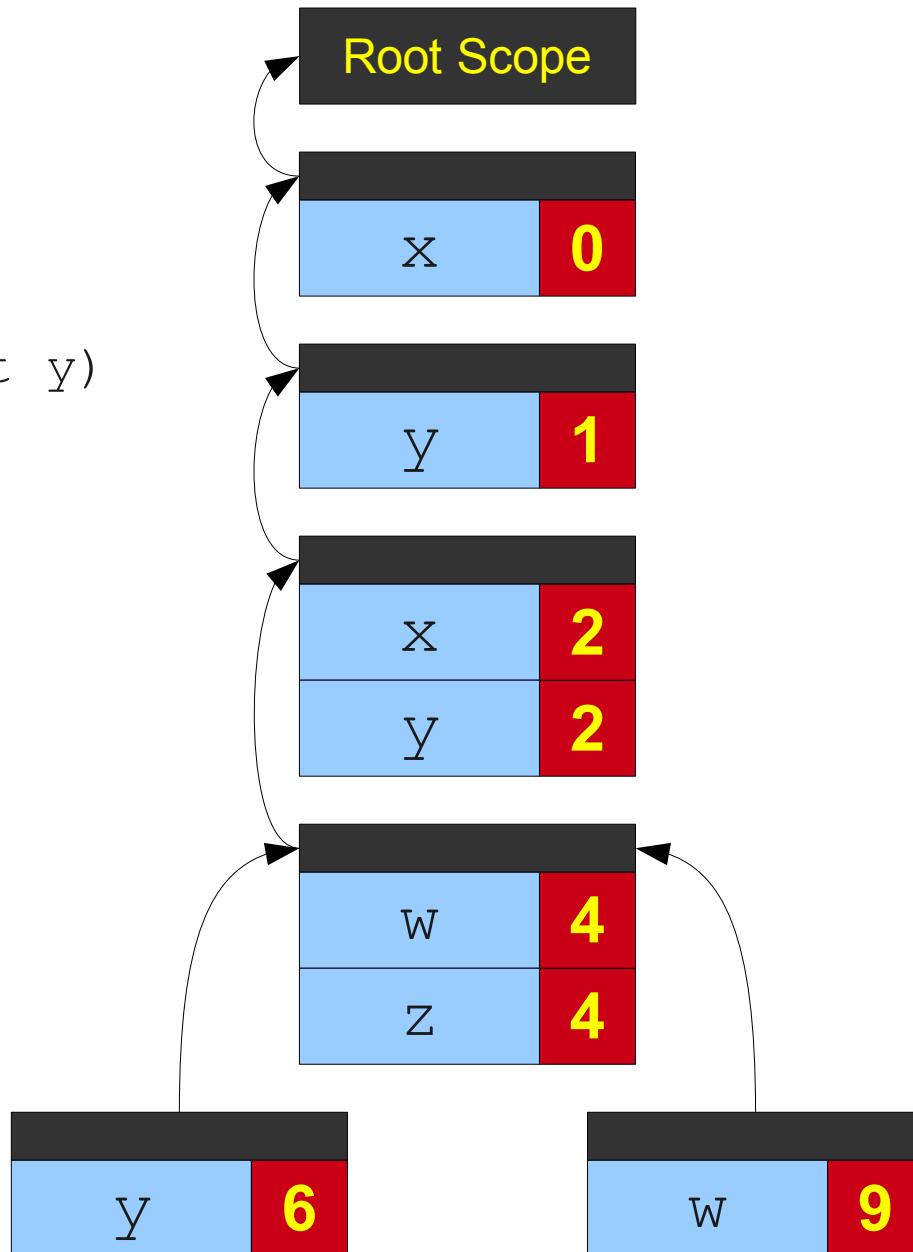
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



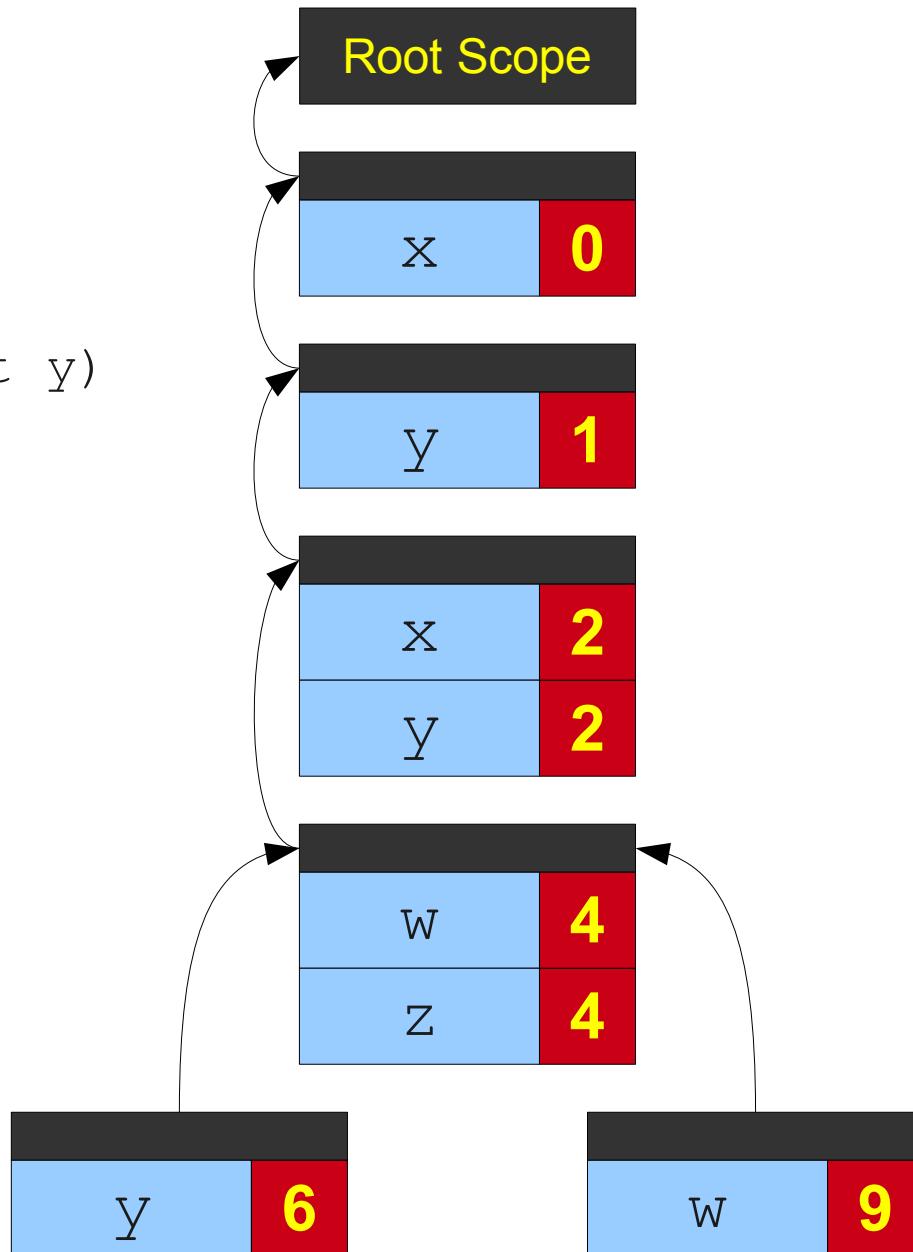
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



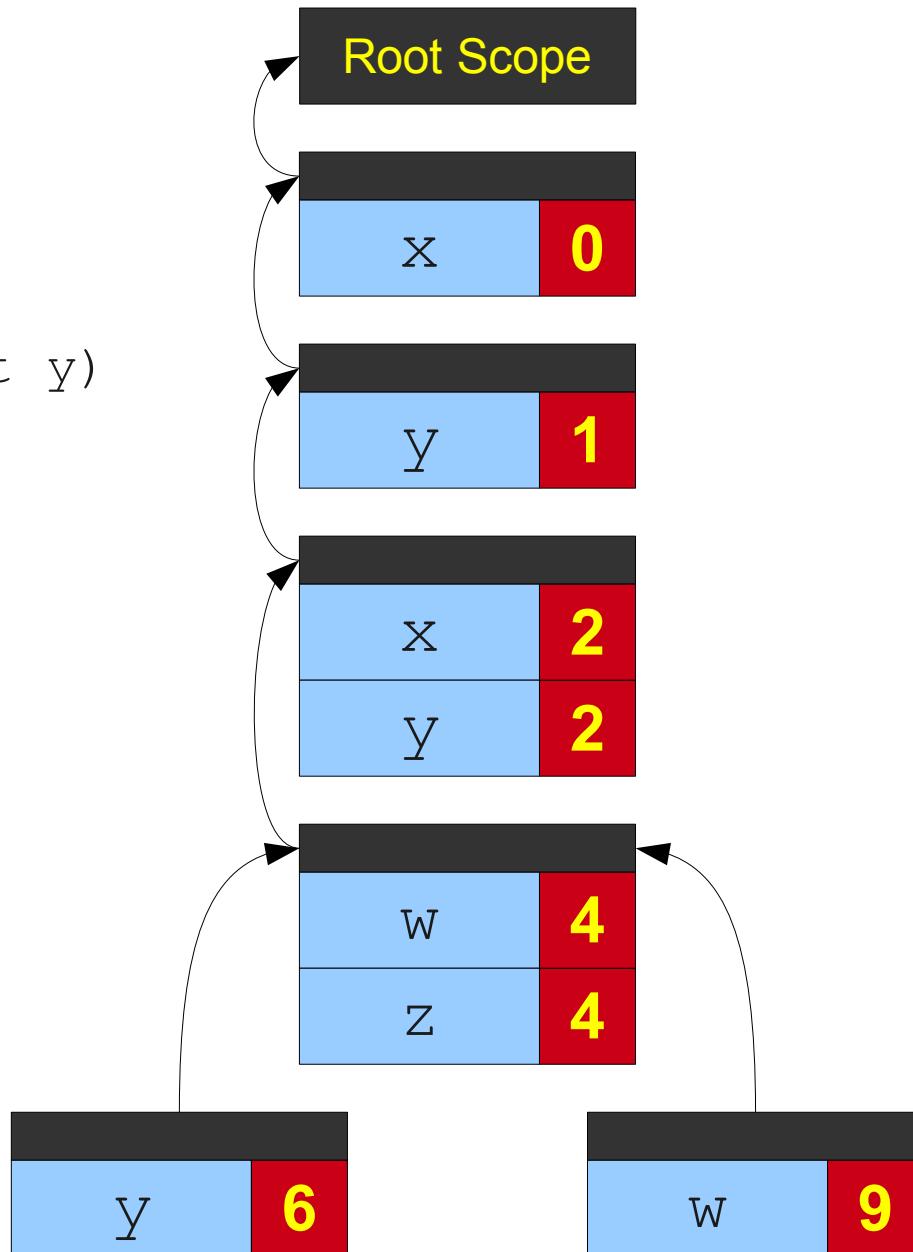
# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



# Another View of Symbol Tables

```
0: int x;  
1: int y;  
2: int MyFunction(int x, int y)  
3: {  
4:     int w, z;  
5:     {  
6:         int y;  
7:     }  
8:     {  
9:         int w;  
10:    }  
11: }
```



# Spaghetti Stacks

- Treat the symbol table as a linked structure of scopes.
- Each scope stores a pointer to its parents, but not vice-versa.
- From any point in the program, symbol table appears to be a stack.
- This is called a **spaghetti stack**.

# Why Two Interpretations?

- Spaghetti stack more accurately captures the scoping structure.
- Spaghetti stack is a **static** structure; explicit stack is a **dynamic** structure.
- Explicit stack is an optimization of a spaghetti stack; more on that later.