Programming Languages

Reference
- Non-local reference
  - Defining scope
- Local reference
- No forward reference
- Obsolete binding

Symbol Table
- Dictionary
- Algorithm for Nested Scoping Implementation
  - When entering a new scope, push a new, empty dictionary on the symbol table stack.
  - When exiting a scope, pop the top dictionary off the symbol table stack.
  - When a name is bound to a value, push the entry onto the current dictionary.
  - Give a name reference, look in the current dictionary.
    - If the name reference has a binding, return the appropriate value.
    - If the name reference is not found, repeat the process with the next dictionary in the stack.
    - If there are no more dictionaries, return failure and report a lookup error.

Example
- Build a symbol table the the following code. Consider both function and variable.
- `<name, type, line number>, type: int, func`
- See next page
```c
1 int h, i;
2
3 void B (int w) {
4     int j, k;
5     i = 2 * w;
6
7     if (i > 0) {
8         int i = 5;
9         w = w + i;
10     }
11
12     w = w + i;
13 }
14
15 void A (int x, int y) {
16     float i, j;
17     B (h);
18     i = 3;
19 }
20
21 int main () {
22     int a, b;
23     h = 5;
24     a = 3;
25     b = 2;
26     A (a, b);
27     B (h);
28 }
```