Symbol Table Management

Prof. James L. Frankel
Harvard University

Version of 6:38 PM 26-Sep-2023

Symbol Tables in Our Implementation

- A symbol table is a data structure that associates each identifier with its type and other information

- Contains all user-declared identifiers
- Associated with each identifier is a type data structure
  - The type data structure is of the form shown in the Type Checking slides

- Every reference to an identifier in the parse tree should be replaced with a pointer to the appropriate entry in the applicable symbol table

Symbol Table for each Overloading Class

- All possible C overloading classes are shown in Table 4-2 in §4.2.4 on page 78

- For our language, we have:
  - Statement labels (different from case labels)
  - Other names

- For a full C implementation, there are three more overloading classes:
  - Preprocessor macro names
  - Structure, union, and enumeration tags
    - These always follows the reserved words **struct**, **union**, or **enum**
  - Component names (referred to as “members” in Standard C)
    - These have a name space within each specific struct or union
    - Their declarations are always within a struct or union and they may only be used following either `.` or **`->`**

Symbol Table at each Scope

- Symbol tables for *statement labels* exist at each procedure scope
- Symbol tables for *other names* exist at:
  - File scope
  - Each procedure scope (same as the outermost block scope within a function)
  - Each block scope

- Please note that formal parameter names in a function definition are in the same symbol table as the other names in the outermost block of a function (see §4.2.2)
  - "In Standard C, the scope of formal parameter declarations in a function definition is the same as the scope of identifiers declared at the beginning of the block that forms the function body."

Symbol Table Inward Linkage

- A global pointer should exist that points to the *file-scope* other-names symbol table
- All global variables and procedures should be present in the *file-scope* other-names symbol table

- Associated with each non-function global variable is a type data structure
- Associated with each function global variable is:
    - a type data structure that specifies the type of the function's return value
    - a type data structure that specifies the number of parameters and type of each parameter
    - a pointer to the function's statement-label symbol table
    - a pointer to the function's outermost-block other-names symbol table (which contains the names and types of all of the function's parameters and the names and types of all of the function's outermost block variables)
    - a pointer to the AST for the body of the function

- Associated with each block (with the exception of the outermost block in a function) is a *block-scope* other-names symbol table
- Associated with each procedure is a *procedure-scope* other-names symbol table
    - Each entry in a block-scope or procedure-scope symbol table has an identifier name and a type

Symbol Table Outward Linkage

- Each block-scope symbol table should have a pointer to the innermost enclosing block-scope's or function-scope's symbol table
  - This allows searching for the declaration of a referenced identifier in the innermost block scope or function scope
  - After searching through block and function scopes, a final search through the file-scope symbol table would be conducted

- The file-scope symbol table is not linked through the outward pointers because the instructions needed to access file-scope variables are different from the instructions needed to access block-scope and function-scope variables

Example of Symbol Tables

- Go over symbolTables.c