Symbol Table

Course : 2CS701 – Compiler Construction

Prof Monika Shah

Nirma University

Ref : Ch.7 Compilers Principles, Techniques, and Tools by Alfred Aho, Ravi Sethi, and Jeffrey Ullman

Glimpse

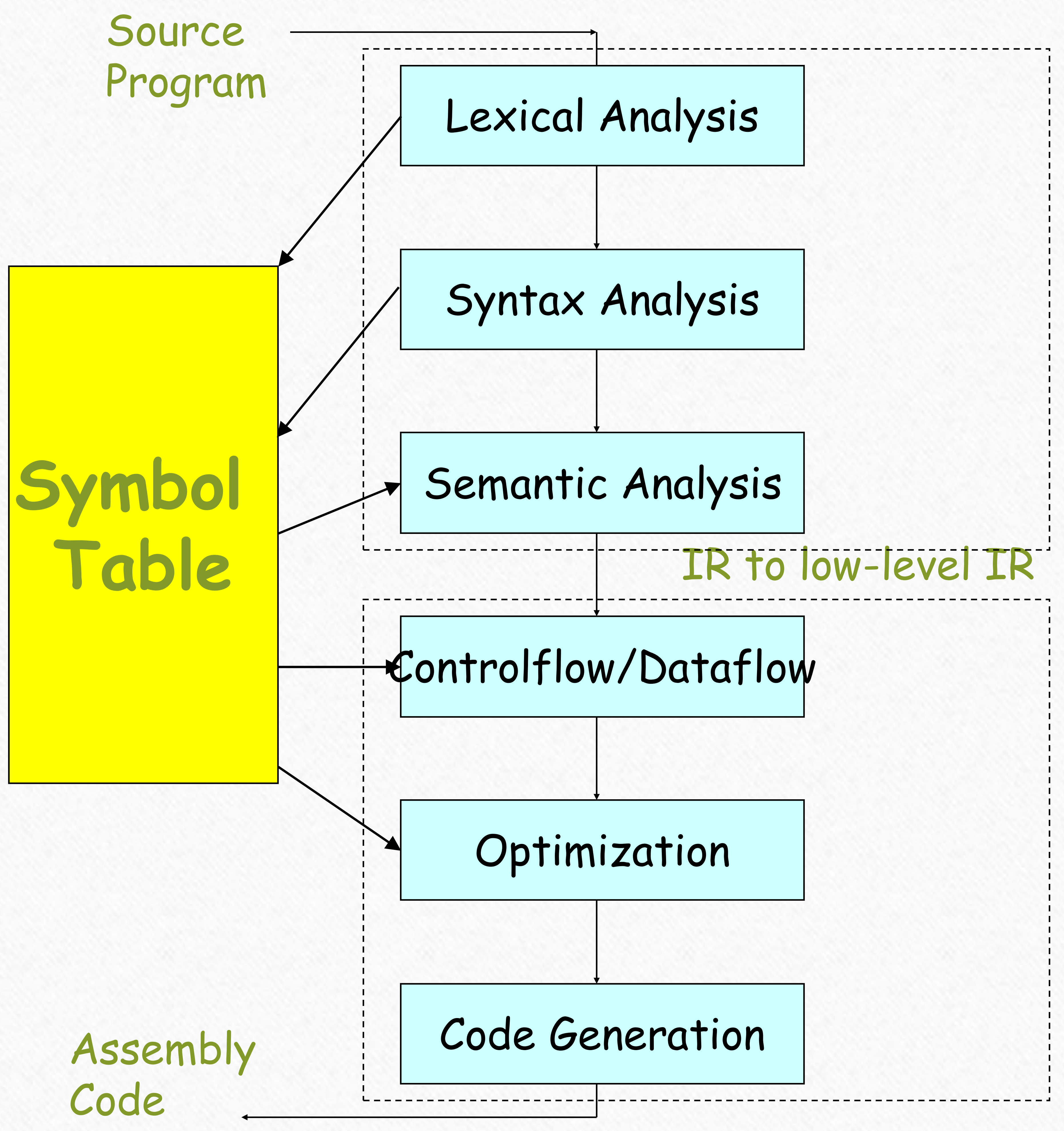
- Introduction to Symbol Table
- Information stored in Symbol Table
- Usage of Symbol Table in various compiler phases
- Operations in Symbol Table
- Issues in Symbol Table Design
- Implementation of Symbol Table
    - One Table for All Symbols
    - Hierarchical structure of Symbol Tables for different scope

Symbol Table

- Essential Data Structure for Compiler
- Stores Information about symbols
    - Type of Symbols : Variables, Procedures, Functions, Constants, Labels, Structures etc.
- Dynamic storage allocation
- Updated by Lexical Analyzer and Parser
- Used by later phase like Semantic Code Analyzer, Code Generator

Information in Symbol Table

Name	Type	Location	Scope	Data Type	Others
Name of Identifier or Pointer to String in String Table	Variable, Procedure, Label, Constant, etc. Variable Type: Primitive , Derived,	Offset within the program where variable is defined			Array limit, fields of records, parameter, return values etc.



- Insert Symbol in Symbol time when occurred first time
- Return pointer to the symbol to Parser

Front end

- Update Datatype of variable, functions, etc..
- Update type of symbol
- Errors : Re-declaration, Un-declared, Prototype etc.

- Type checking
- Verify data type of operands for each operator
- Verify data types function parameters

Back end

- Two or more temporary variables can be combined if they are of same type

- Memory storage size depends on data type of variables

Machine independent asm to machine dependent

Operations in Symbol Table

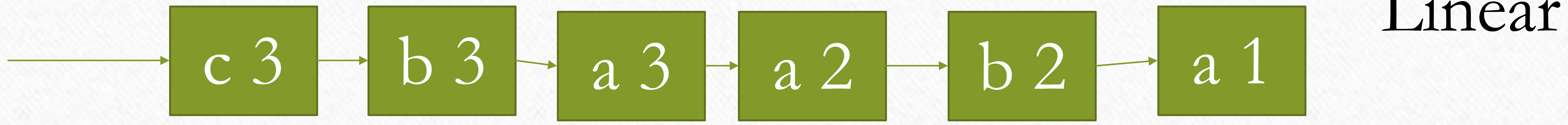
- Lookup
- Insert
- Modify
- Delete

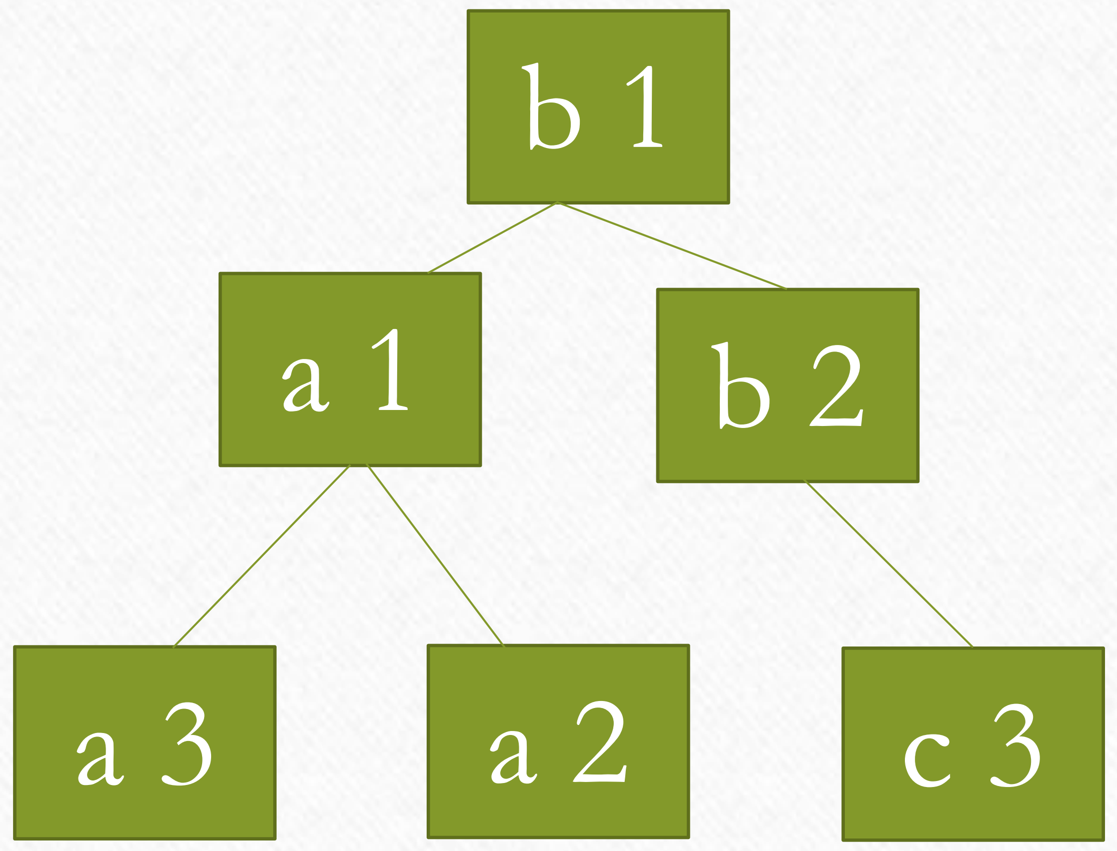
Issues in Symbol Table design

- Selection of Formats : Linear , List, Tree etc.
- Access Method : Linear Search, Binary Search, Hashing, Tree Search etc.
- Location of Storage : Primary Memory (Generally), Secondary
- Scope Issues : Inner block can access Outer block symbols , but not opposite

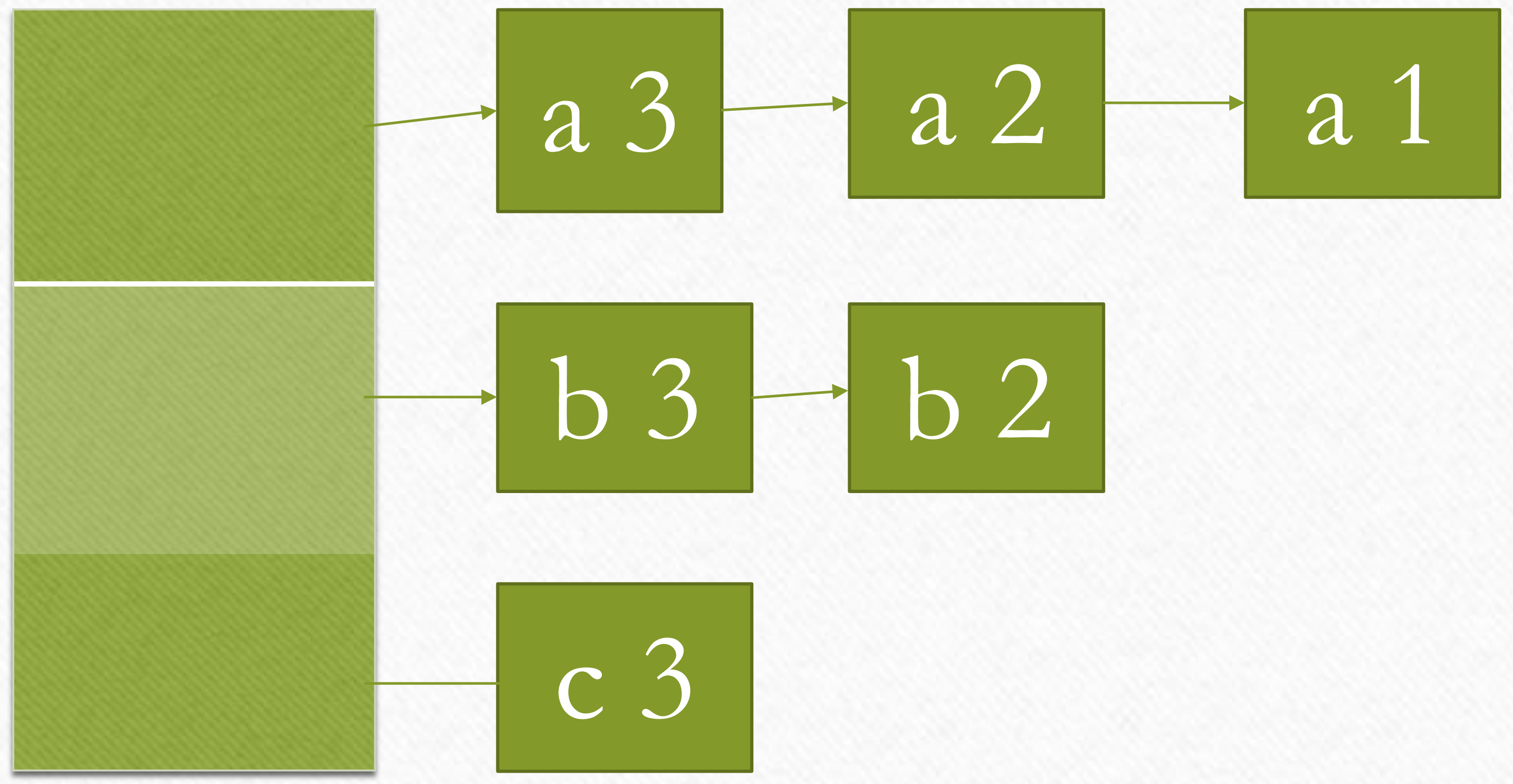
One Table for All Scopes

```
void main
{
int a;
{
int b,a;
{
int  a,c;
}
}
}
```





Tree



Hashing

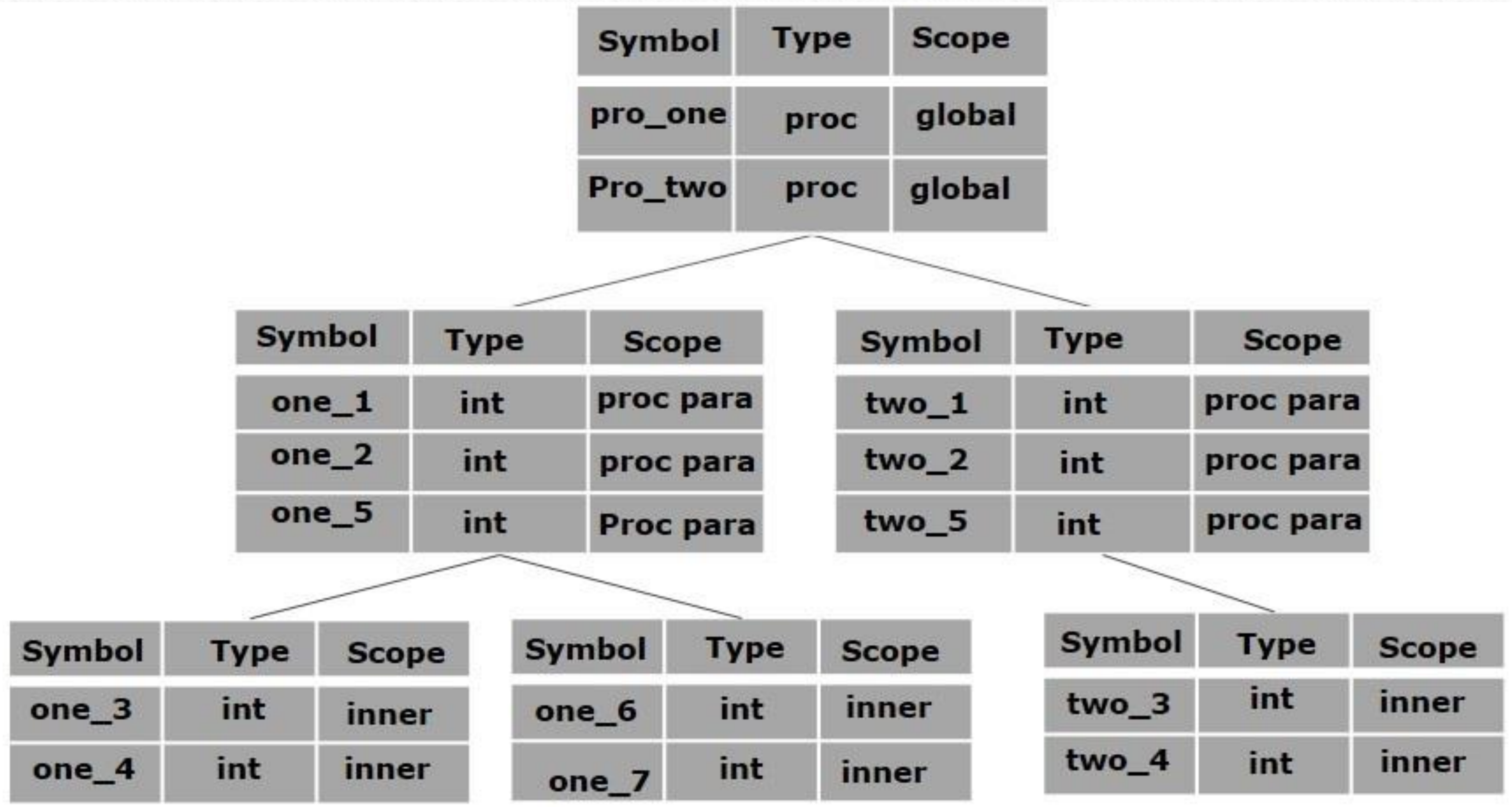
```
int value=10;
void pro_one()
       { int one_1;
          int one_2;
             {  int one_3;
                int one_4;  }
         int one_5;
            {  int one_6;
               int one_7;  }

      }
```

```
;
void pro_two()
       { int two_1;
          int two_2;
             {  int two_3;
                int two_4;  }
         int two_5;

}
```

Hierarchical structure of Symbol Tables for different scope



Symbol	Type	Scope
pro_one	proc	global
Pro_two	proc	global

Symbol	Type	Scope
one_1	int	proc para
one_2	int	proc para
one_5	int	Proc para

Symbol	Type	Scope
two_1	int	proc para
two_2	int	proc para
two_5	int	proc para

Symbol	Type	Scope
one_3	int	inner
one_4	int	inner

Symbol	Type	Scope
one_6	int	inner
one_7	int	inner

Symbol	Type	Scope
two_3	int	inner
two_4	int	inner