

*By Shweta Tiwari from IT Department*

**Covered All Topics UNIT-4 of "CD- COMPILER DESIGN (KIT-052)"**

[shwetatiwario8@recabn.ac.in](mailto:shwetatiwario8@recabn.ac.in)  
[shwetatiwario8aug@gmail.com](mailto:shwetatiwario8aug@gmail.com)

---

# **CD: UNIT-4**

# **Symbol Tables**

*FALL SEMESTER, YEAR (V/VI, 3rd)*

*FALL SESSION (2022-23)*

*(CD)*

*MS. SHWETA TIWARI*

*Published: September, 2022*



[shwetatiwario8@recabn.ac.in](mailto:shwetatiwario8@recabn.ac.in)  
[shwetatiwario8aug@gmail.com](mailto:shwetatiwario8aug@gmail.com)

---

## **UNIT-4**

# **CD**

# **COMPILER DESIGN**

---

**COMPILER DESIGN**

**UNIT IV – Symbol Tables**

**PREPARED FOR**  
Engineering Students  
All Engineering College

**PREPARED BY**  
SHWETA TIWARI  
Guest Faculty

---

Symbol Table: Symbol table is a data structure containing a record for each identifier with fields for the attributes of the identifier.

- \* It stores the information about scope and binding information about names.
- \* Symbol table is built in lexical & syntax analysis phases.
- \* The symbol table is used by various phases as follows:- Semantic analysis phase refers symbol table for type conflict issue. Code generation refers symbol table knowing how much run-time space is allocated. What type of runtime space is allocated.

### Symbol Table Entries

- \* Each entry in the symbol table is for the declaration of a name(string).
- \* The format of entries does not have to be uniform;
- \* The items to be stored in symbol table are
  - (i) Variable names
  - (ii) Constants
  - (iii) Procedure names
  - (iv) Function Names.
- \* Each entry can be implemented as a record consisting of a sequence of consecutive words of memory.
- \* Information is entered into the symbol

(2)

table at various times.

- \* The symbol table entry can be setup itself when the role of a name becomes clear, with the attribute values being filled in as the information becomes available.
- \* The entries in symbol table stores information about data types, scope information (where it is used), storage allocation & allocation size.

Various

### Data structures used for symbol table

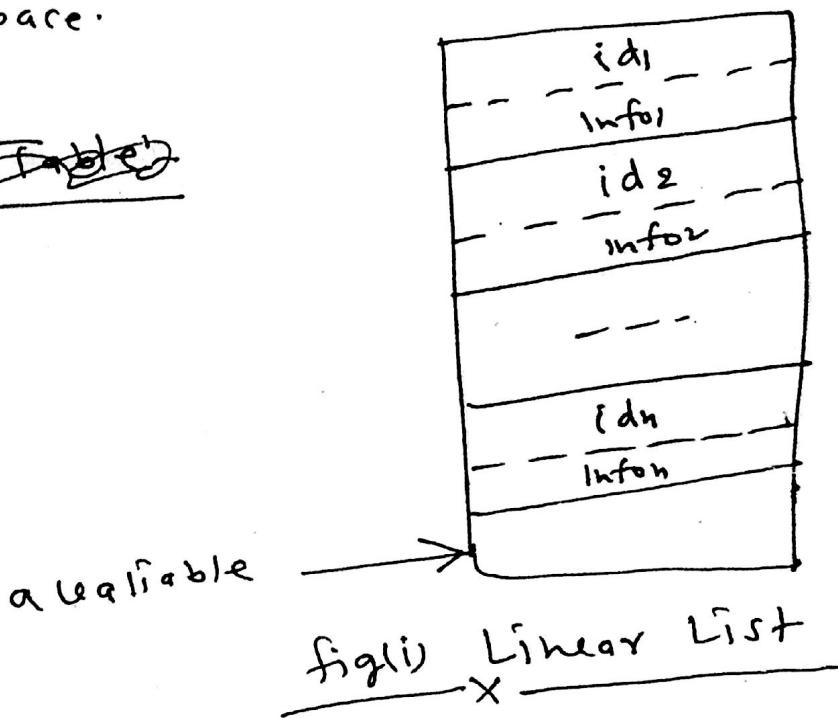
The various data structures used for symbol table construction are as given -

#### ① Linear List:-

- (a) It is the simplest & easiest to implement data structure for a symbol table.
- (b) In this, we use single array or set of arrays to store names & their associated information. New names are added to the list in the order in which they are encountered.
- (c) The position of the end of the array is marked by pointer "Available", pointing to where the next symbol table entry will go.
- (d) The search for a name proceeds backwards from the end of the array to the beginning.

- (e) If we reach the beginning of the array without finding a name, a fault occurs - an expected name is not in the table.
- (f) While inserting a new name, we should ensure that it should not be already there. If it is there "multiple defined Name".
- (g) This data structure takes min. amount of space.

## ② Hash Table



## ② Hash Table

- (a) Hash table method is an important technique used to search the records of symbol table.
- (b) This method is superior to linear list.
- (c) A hash table consisting of a fixed array of m pointers to table entries.

- (c) In hashing scheme, two tables are maintained  
 (i) hash table      (ii) symbol table.
- (d) The hash table consists of  $k$  entries from 0 to  $k-1$ .  
 These entries are basically pointers to symbol table.
- (e) To determine whether the 'Name' is in symbol table, we use a hash function ( $h$ ) such that  
 $\text{position of Name} = h(\text{Name}) = \text{Index } 0 \text{ to } k-1$
- (f) The position provides the exact locations of name in symbol table.
- (g). The adv. of using hashing is that it is very quick in searching but the disadv. is that hashing is complicated to implement.

HashTable

Sum
i
j
:
avg
:
:

Symbol Table

Name      Info      hash link

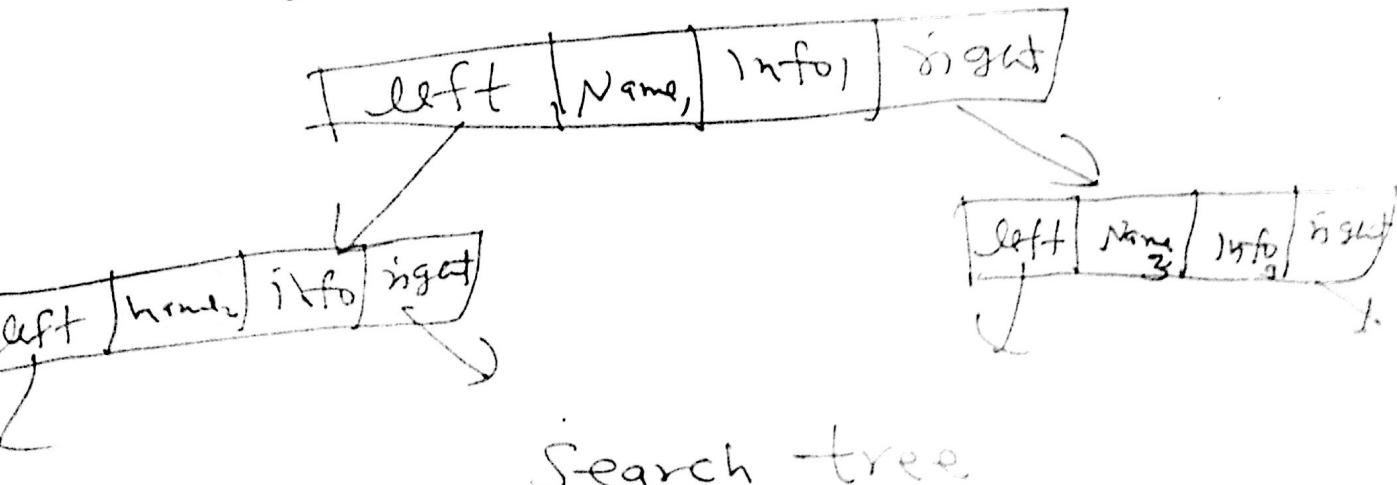
Sum		
i		
j		
avg		

(b) Hash Table Method

Syntax Vmt-y

(1)

Search trees:- A Search tree is a more efficient approach to symbol-table organization. We add two links, left & right in each record, and these links point to the record in the search tree. Whenever a name is to be added, first the name is searched in the tree. If the name is not available, then the record for the new name is created and added to the proper position in the search tree. The organization has the property of alphabetical accessibility.



## Representation of Scope information in symbol

### Table

- ⇒ Each Name possesses a region of validity within the source program, called the 'Scope' of that Name.
  - ⇒ The scope rules of a language determine which declaration of a name applies when the name appears in the program.
  - ⇒ The rules governing the scope of names in a block structured language are as follows-
    - (a) A name declared within a block B is valid only within B.
    - (b) If block  $B_1$  is nested within block  $B_2$ , then any name that is valid for  $B_2$  is also valid for  $B_1$ , unless the identifier for that name is redeclared in  $B_1$ .
  - ⇒ Such types of scope rules requires a more complicated symbol table organization.
  - ⇒ One technique is to use multiple symbol tables, one for each active block, such as the block that the compiler is currently in. Each table is a list of names along with their associated attribute. The tables are organized into stacks.
- The representation of scope information can be understand using the following example.



example

(6) Let us consider the following example--

Program main

Var x, y: Integer

Procedure P:

Var n, a: boolean

Procedure Q:

Var n, y, z: real;

Begin

!

end

Begin

!

end

Begin

!

end

The ~~above~~ symbol table organization is as follows:-

Z	Real
Y	Real
X	Real

Symbol Table  
for block Q

Q	Proc.
a	boolean
x	boolean

Symbol Table  
for block P

P	Proc.
y	integer
x	integer

Symbol Table  
for main

↓ null

## Run Time

### Storage Allocation - Organization

⇒ The Compiler obtains a block of storage from the Operating System & use it for running the Compiled program. This block of memory is called runtime storage.

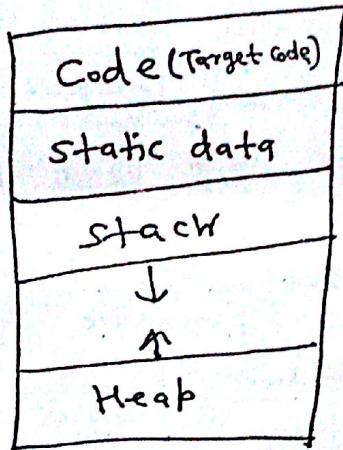
⇒ The runtime storage is divided to hold

(i) Generated Target Code

(ii) Data object

(iii) Control stack to keep track of procedure

activations.



### Runtime Storage organization

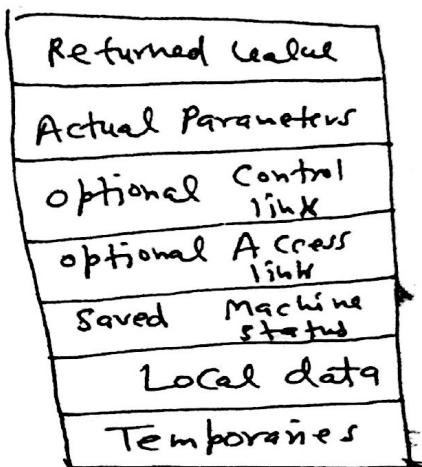
⇒ The size of the generated target code is fixed at compile time, so compiler places them in statically determined area. Similarly the size of some data objects are also known at compile time, so they are also placed in statically determined area.

⇒ Counter part of stacks is used to manage activation of procedures. When a procedure call occurs, execution of an active procedure is interrupted & status (Program Counter & value of machine register) is saved on stack.

⇒ A separate area called heap holds all other information.

## (8) Activation Record:-

- Each execution of a procedure is called its activation.
- The activation record is a block of memory used for managing information needed by a single execution of procedure.
  - The various fields of activation records are as follows:-



General Activation Record

- (i) Returned Value: This field is used by the called procedure to return a value to the calling procedure.
- (ii) Actual Parameter: This field is used by the ~~called procedure~~ calling procedure to supply parameters to the called procedure.
- (iii) optional Control Link: It points to the activation record of the calling procedure.
- (iv) optional Access Link: used to refer to non-local data held in other activation records.
- (v) Saved Machine Status: It holds information about the state of the machine just before the procedure is called.
- (vi) Local Data: It holds data that is local to an execution of a procedure.
- (vii) Temporaries: Temporaries variables are needed during the evaluation of expressions.

## Storage Allocation strategies:

There are three different storage allocation strategies:-

### (1) Static Allocation: It allocates storage

to all the data objects at compile time.

- \* The binding of name with the amount of storage allocated do not change at run time. Due to this, the name of this allocation is static allocation.

ab In static allocation, the compiler can determine the amount of storage required by each data object.

ab The limitations of static allocation are

- (a) The data structures can not be created dynamically.

(b) Recursive procedures are not supported by this type of allocation.

### (2) Stack Allocation: In stack allocation,

stack is used to manage runtime storage.

ab Data structures and data objects can be

created dynamically.

ab Using Last in first out (LIFO) activation

records and data objects are pushed

onto the stack.

ab It supports dynamic memory allocation

but it is slower than static allocation strategy.

ab It supports recursive procedures.

### ③ Heap Allocation

- ⇒ In heap allocation, heap is used to manage dynamic memory allocation.
- ⇒ Data structures & data objects can be created dynamically.
- ⇒ A contiguous block of memory from heap is allocated for activation record or data object.
- ⇒ The efficient heap management can be done by
  - (a) Creating a linked list for the free blocks
  - (b) Allocate the most suitable block of memory from the linked list.

Error Detection & Recovery - Error detection & recovery is one of the imp. function performed by compiler.

Error Handling Process - The process of locating errors and reporting to user is called "error handling process".

### Function of Error Handler

- ① Detection - The error handler should identify all possible errors from source code.
- ② Reporting - It should report these errors with appropriate messages to the user.
- ③ Recovery - The error handler should recover from these errors by repairing them in order to continue processing of the program.

(2) Syntactic Errors

- ab The errors that occurs during the syntax analysis phase of the compiler is known as syntactic error.
- ab Syntactic errors are arises due to the missing operators, misspelled keywords and unbalanced parenthesis.
- ab The syntax analyzer receives tokens from lexical analyzer and if the token do not follow the grammatical rules of programming language, then the syntactical errors occurs.
- ab strategies for recovery from syntactic errors?  
(a) Panic Model This is widely used technique. It is also simpler to implement. In this method, the parser discards input symbol one at a time & continue until one of designated set of synchronizing tokens is found. This method is mainly successful when there are less no. of errors in the same statement.  
(b) Phrase Level Recovery In this method, on discovering error, parser perform local correction on remaining input. The local correction can be replacing comma by semicolon, deletion of extra semicolons or inserting missing semicolon. The only disadvantage of this method is that, it find difficult to handle the situation, where the actual error has occurred before the point of detection.

### (3) Semantic Errors

- ❖ Semantic errors are those errors which get detected during semantic analysis phase.
- ❖ The various errors in this phase are:-
  - Incompatible types of operands.
  - Undeclared variables.
  - Not matching of actual arguments with formal arguments.
- ❖ If the data types of two operands in an expression are not compatible with each other, then automatic type conversion is done by compiler.

### Storage allocation for block structured language

In block structured language, the storage allocation can be done for two types of data variables.

- ① Local data  
(within the same procedure)
- ② Non local data.  
(outside a given procedure)

① Local Data— The local data can be accessed with the help of activation record. The offset relative to base pointer of an activation record points to local data variables within an activation record.

② Access to Non Local data— A procedure sometimes refers to variables which are non local to it. Such variables are called Non local variables.

The way that the nonlocals are accessed depends upon the scope rules of the language.

The scope rules determine (14) the treatment of references to non-local names.

There are two different types of Scope rules:-  
non local names.

(1) Static Scope Rules (2) Dynamic Scope Rules

(1) Static Scope Rules - also called Lexical Scopes

• determined at Compile time.

• ~~context~~ Determined the declaration that applies to a name by examining the program text alone.

• Pascal, C & Ada languages uses Static Scopes.

• This is used by block structured languages.

(2) Dynamic Scope Rules:-

• determines the declaration applicable to a name at run time, by considering the current activation record.

\* Lisp, APL & Shobol are among the languages that use dynamic scope.

• They are used by non block structured languages.

Access Links & Displays ~~use~~ to access Non-local names:-

A procedure may access non local names using either access link or displays

Access links - \* A direct implementation of lexical (static) scope for nested procedures is

obtained by adding a pointer called "Access link" to each activation record.

\* If procedure p is nested immediately within q in the source text, then the access link in an activation record for p points to an access link in the record for the most recent activation of q.

- (15)
- The code to set up access link can be determined at ~~initial~~ compile time, using the idea of nesting depth.

- (a) The outermost scope has nesting depth = 1
- (b) The nesting depth increases by 1 each time we enter a new scope & decreases by 1, when we leave a scope.

- If procedure A at depth  $n_A$  calls procedure B at depth  $n_B$  then
  - (i) If  $n_A < n_B$ , then B is enclosed in A and  $n_A = n_B + 1$
  - (ii) If  $n_A > n_B$ , then it is either a recursive call or calling a previously declared

Displays → The purpose of using displays is to avoid the runtime overhead of multiple access -  
is an array of pointers to activation records used to access activation record.  
in display:-

- (a) An array of pointer to activation record is maintained.
- (b) Array is indexed by nesting level.
- (c) The pointers point to only accessible activation record.
- (d) The display changes when a new activation occurs & it must be reset when control return from the new activation.

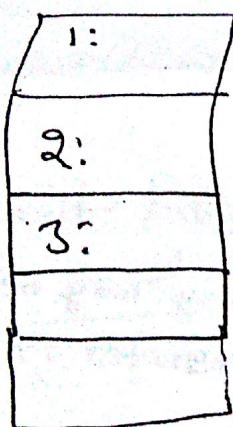
- (e) It is an efficient way to access nonlocal names.  $\rightarrow \rightarrow$  block

(16)

## Comparison b/w Access links & Display

- ① Access links can require more time (at runtime) to access non-locales.
- ② Display require more space at runtime.
- ③ Display is used to generate code effectively.
- ④ It can also require more time to setup a new access link when a procedure is called.

→ In display, for each level, there will be an entry current activation record at Level 1.



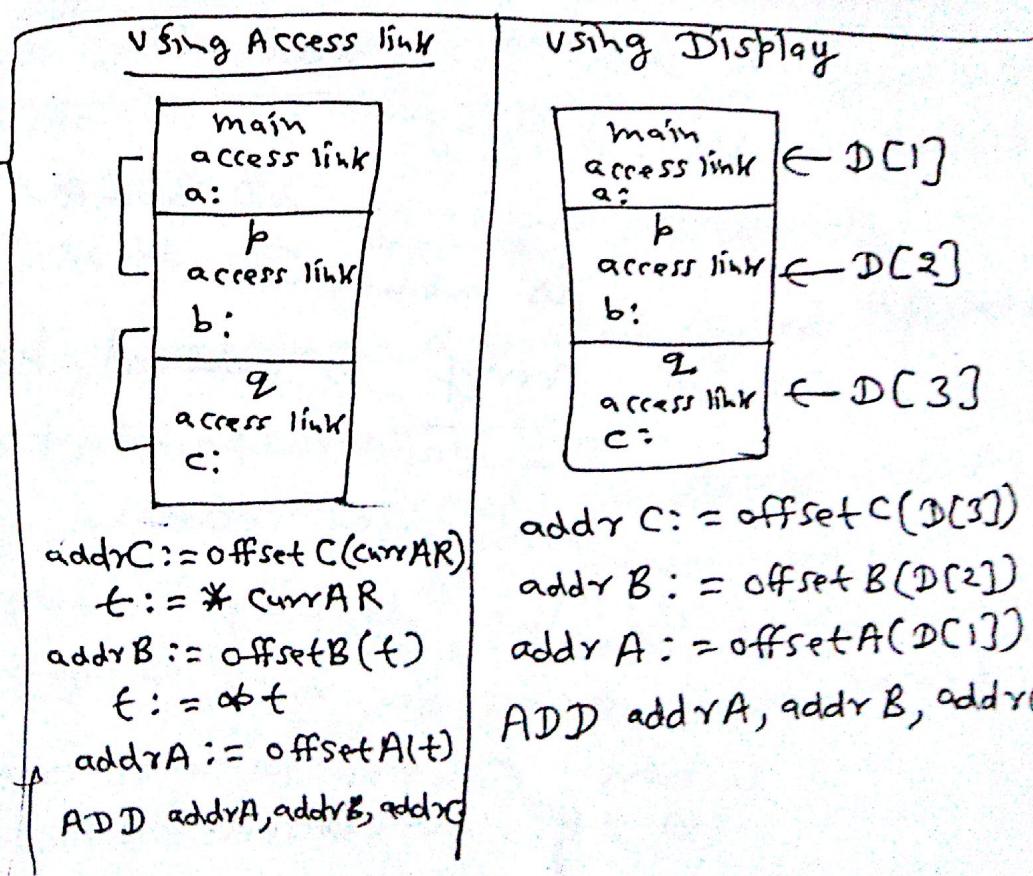
Current activation record at Level 2

Current activation record at Level 3

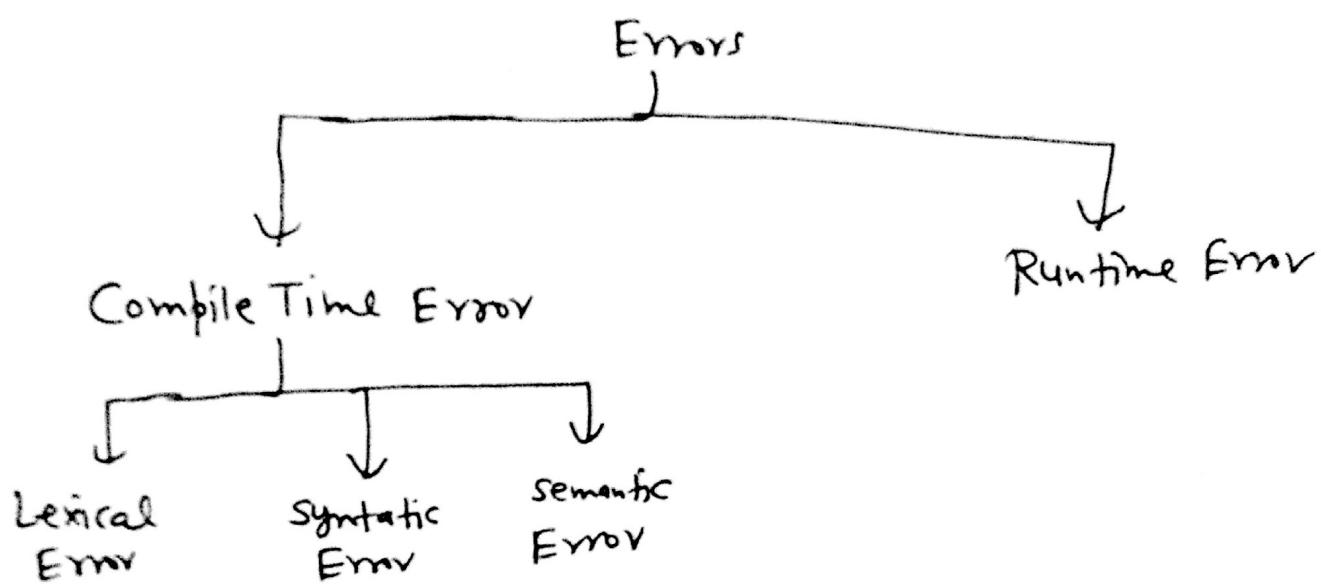
example for Accessing Non Local data using Access links & displays:

```

Program main();
Var a : int;
Procedure p();
Var b : int;
begin z; end.
Procedure q();
Var c : int;
begin
  c := a + b;
end;
begin p(); end.
    
```



## Classification of Errors..



① Lexical Error:- The errors occurring in lexical analysis phase is known as Lexical error.

- ❖ Lexical Errors can be arises due to
  - An Extra Character
  - In correct Character
  - Missing character
  - Two adjacent transposed characters
  - Exceeding length of identifier
- ❖ The lexical analyzer detects an error when it discovers that an input's prefix does not fit the specification of any token class. It makes use of panic mode recovery method in which successive characters from the remaining input are deleted until the well formed token is found.
- ❖ If any unmatched string occurs, then Insert appropriate String/Character to match string.