Symbol Tables

Symbol Table:

- In computer science, a symbol table is a data structure used by a language translator such as a compiler or interpreter, where each identifier in a program's source code is associated with information relating to its declaration or appearance in the source.
- A data structure used by a compiler to keep track of semantics of variables. Variables mainly classified with the representation on data types and scope of things. It is a data structure used by compiler to keep track of semantics of names.
- After syntax tree have been constructed, we must check whether the input program is type-correct (called type checking and part of the semantic analysis). During type checking, a compiler checks whether the use of names (such as variables, functions, type names) is consistent with their definition in the program.
- Consequently, it is necessary to remember declarations so that we can detect inconsistencies and misuses during type checking. This is the task of a symbol table.
- First, any symbol table can define what is the data type and scope where it needs to be stored. These all the statges we need to consider for the representation part. Please refer the below diagram (Fig 1)

www.sakshieducation.com

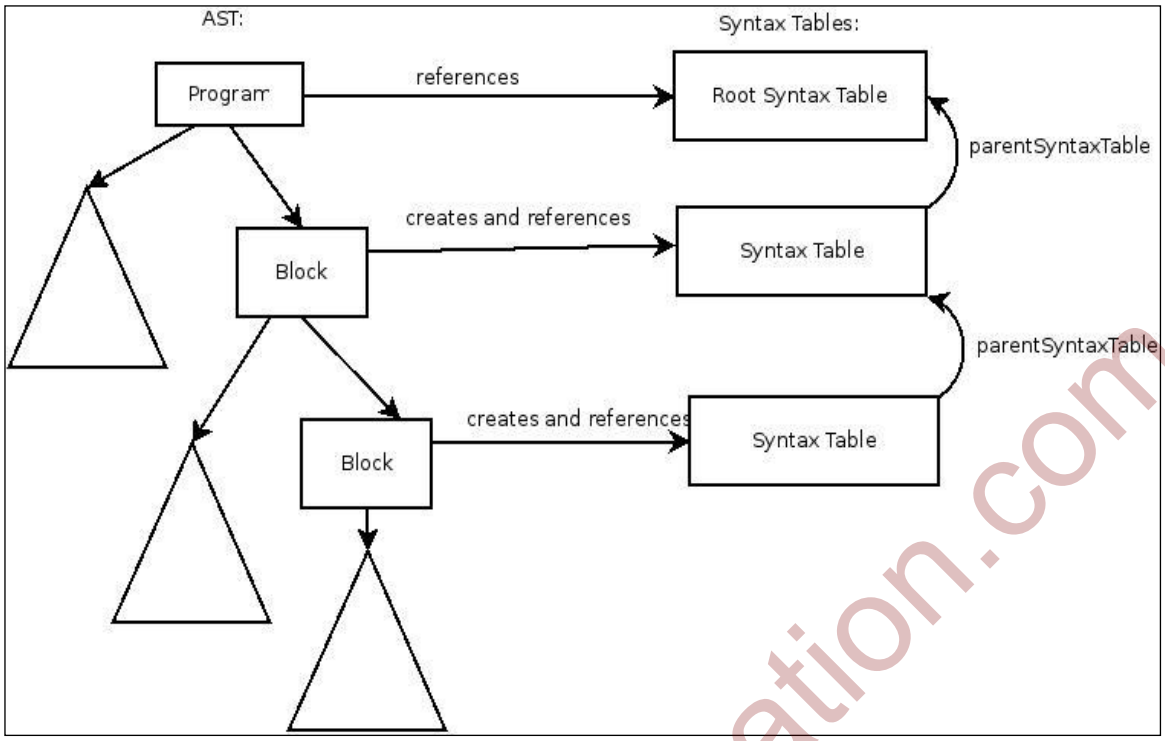

Fig 1

Tree Structures:

- Mostly tree structrure is the best way to implement and using the symbol tables, hash tables.
- This is the most efficient approach to building symbol table in memory database. You can use various in-memory databases as modern computer provide ample amount of memory.

Relational database:

- In database, we have used relational database as to describe and define the all tables relation in MYSQL and SQL.
- In this relational database representation, it provides important advantages due to better visibility and ability to manipulate the database using SQL. That greatly simplifies debugging.

Scope Rules:

www.sakshieducation.com

- Scope rules require a more complicated symbol table organization than simply a list of associations between names and attributes. That's why many experimental languages and also in the past such languages as Perl use flat "global" namespace.
- The simplest approach of to use multiple symbol tables, one for each active block, such as the block that the compiler is currently in. The other is to use block prefix to denote the "real" name of the variable. You can also use nesting depth as part of the name. In this case a triple {*procedure name, nesting depth, variable name*} uniquely identifies all variables. organization for block structures languages:
- A block consists of a sequence of statements and/ or blocks, preceded by declarations of variables. Variables declared at the head of a block are visible throughout the block and any nested blocks, unless a variable of the same name is declared at the head of an inner block.
- Mainly the declaration are happening like inner block, outer block-structured, based on this block we are mainly declare the veriables, block of code and methods, classes and objects which are described in the block of code.
- The concept of block structure was introduced in the Algol family of languages, and block-structured languages are sometimes described as Algol-like. The concept of nested scopes implicit in block structure contrasts with FORTRAN, where variables are either local to a program unit (subroutine) or global to several program units if declared to be common. Both of these contrast with COBOL, where all data items are visible throughout the entire program. Please refer the below fig (2)

```
PROCEDURE get_happy (ename_in IN VARCHAR2)      Header
IS
   hiredate DATE;                               Declaration
BEGIN
   hiredate := SYSDATE - 2;
   INSERT INTO employee                         Execution
      (emp_name, hiredate)
   VALUES (ename_in, hiredate);
EXCEPTION
   WHEN DUP_VAL_IN_INDEX
   THEN
      DBMS_OUTPUT.PUT_LINE                      Exception
         ('Cannot insert.');
END;
```

Fig (2)

www.sakshieducation.com

Example:

```
Public Class
{
Int a =10;
Int b=20;
int c=a+b;
public static void main (“Addition”+c);
}
```

Hashing:

- Most of the programming languages we have suppose to use for writing and developing the projects and products in that also, we are maintaining the concept as programming with security.
- Some of the banking side domains we are frequently used for transaction of money, let us imagine if we dont have security then easily hackers will hack the data so to override this problem we preferred as hashing.
- Producing hash values for accessing data or for security. A hash value (or simply hash), also called a message digest, is a number generated from a string of text.
- The hash is substantially smaller than the text itself, and is generated by a formula in such a way that it is extremely unlikely that some other text will produce the same hash value.

Hash Function:

- A hash function maps a big number or string to a small integer that can be used as index in hash table.
- Properties:
    - Efficiently computable.
    - Should uniformly distribute the keys (Each table position equally likely for each key)

Hash Table:

- An array that stores pointers to records corresponding to a given phone number. An entry in hash table is NIL if no existing phone number has hash function value equal to the index for the entry. (fig 3)

www.sakshieducation.com

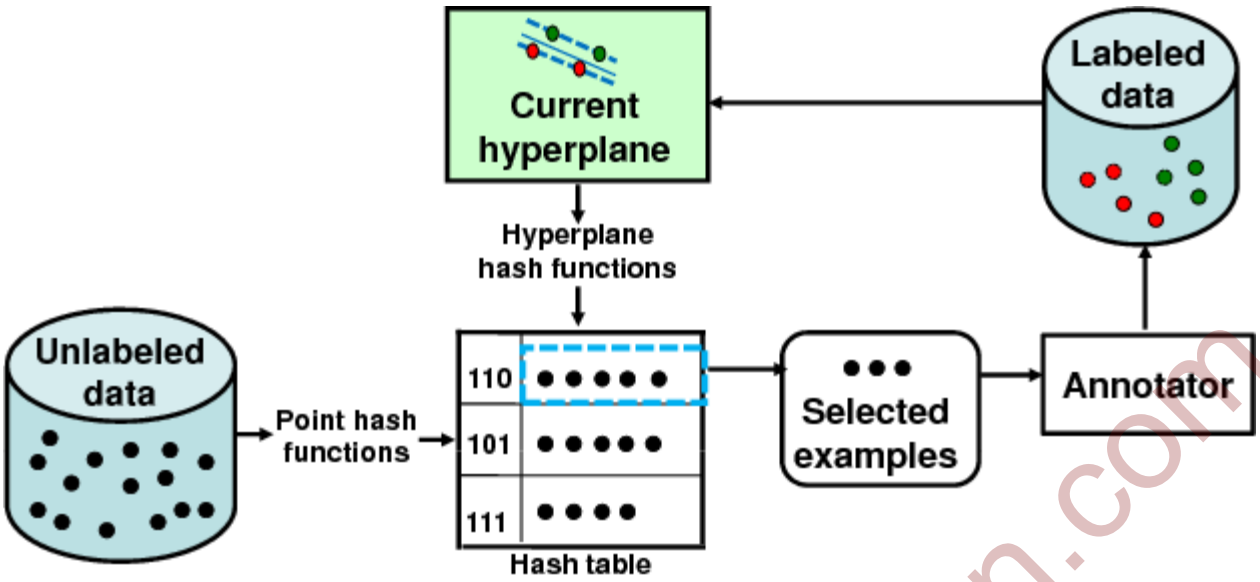

Fig 3

Tree Structures Representation of Scope Information:

- A tree structure or tree diagram is a way of representing the hierarchical nature of a structure in a graphical form. It is named a "tree structure" because the classic representation resembles a tree, even though the chart is generally upside down compared to an actual tree, with the "root" at the top and the "leaves" at the bottom.
- A tree structure is conceptual, and appears in several forms. For a discussion of tree structures in specific fields, see Tree (data structure) for computer science: insofar as it relates to graph theory, see tree (graph theory), or also tree (set theory). Other related pages are listed below.

www.sakshieducation.com

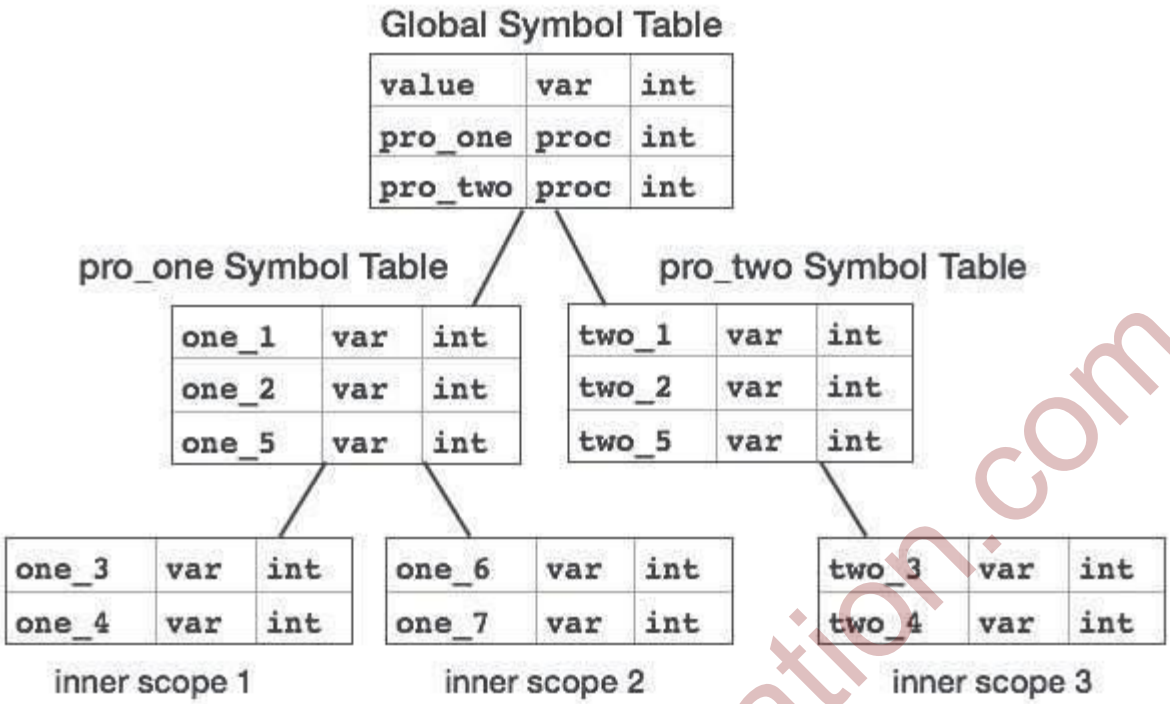

Fig 4

- A compiler maintains two types of symbol tables: a global symbol table, which can be accessed by all the procedures, and scope symbol tables that are created for each scope in the program Fig 4.
- To determine the scope of a name, symbol tables are arranged in hierarchical structure as shown in the example below:

```
. . .
int value=10;
void pro_one()
 {
 int one_1;
int one_2;

{          \
 int one_3;     |_ inner scope 1
 int one_4;     |
}           /
int one_5;
{           \
 int one_6;     |_ inner scope 2
```

```
        int one_7;      |
        }               /
      }
  void pro_two()
  {
  int two_1;
  int two_2;
      {             \
    int two_3;      |_  inner scope 3
    int two_4;      |
    }               /

  int two_5;
  }
. . .
```

Block Structures and Non-Block Structure Storage Allocation:

- Mainly the structures in programming language has defined in two ways that is block-structured and non block-structured. While executing the programs this type of structures will play main role like retrieving the values for input and out put results and using of data types, arrays, looping, conditional and structured statements etc.... In order to represent the looping and conditional statements with block-structured only.

Static Storage Allocation

- In a static storage-allocation strategy, it is necessary to be able to decide at compile time exactly where each data object will reside at run time. In java, you see "static variables", "static methods", "static classes" and "static blocks".
- Static variables, static methods and static classes are known to everyone but what is this "static block". Let’s see what, where and how these static blocks are used.
- However, before going into "static block", lets refresh what other static stuff are. Now "static variables" are class variables i.e., there will be only one copy for each class and not one copy for each object of the class and these variables will be accessed without instantiating the class.
- Then what are static methods. Again they are class methods i.e., they can be accessed without

www.sakshieducation.com

creating an instance of the class and like static variables; static methods will be accessed without instantiating the class.

- Note that static methods cannot access instance variables. They can access only static variables.
- Next, what are static classes? You cannot declare a top-level class as a static class. Java will throw a compilation error. Only inner classes that are member classes can be declared as static.
- If we declare member classes as static, we can use it as a top-level class outside the context of top-level class.
- One catch here is "The static keyword does not do to a class declaration what it does to a variable or a method declaration." - what it means is say for example you have a static variable, then to access that static variable you will use the notation.

Runtime stack and heap storage allocation:

- Stacks in computing architectures are regions of memory where data is added or removed in a last-in-first-out (LIFO) manner.
- In most modern computer systems, each thread has a reserved region of memory referred to as its stack. When a function executes, it may add some of its state data to the top of the stack; when the function exits it is responsible for removing that data from the stack.
- At a minimum, a thread's stack is used to store the location of function calls in order to allow return statements to return to the correct location, but programmers may further choose to explicitly use the stack. If a region of memory lies on the thread's stack, that memory is said to have been allocated on the stack.
- Because the data is added and removed in a last-in-first-out manner, stack-based memory allocation is very simple and typically faster than heap-based memory allocation (also known as dynamic memory allocation).
- Another feature is that memory on the stack is automatically, and very efficiently, reclaimed when the function exits, which can be convenient for the programmer if the data is no longer required. If however, the data needs to be kept in some form, then it must be copied from the stack before the function exits. Therefore, stack based allocation is suitable for temporary data or data which is no longer required after the creating function exits.
- A thread's assigned stack size can be as small as only a few bytes on some small CPU's. Allocating more memory on the stack than is available can result in a crash due to stack overflow.

- Some processor families, such as the x86, have special instructions for manipulating the stack of the currently executing thread. Other processor families, including PowerPC and MIPS, do not have explicit stack support, but instead rely on convention and delegate stack management to the operating system's application binary interface (ABI). Refer below fig 5

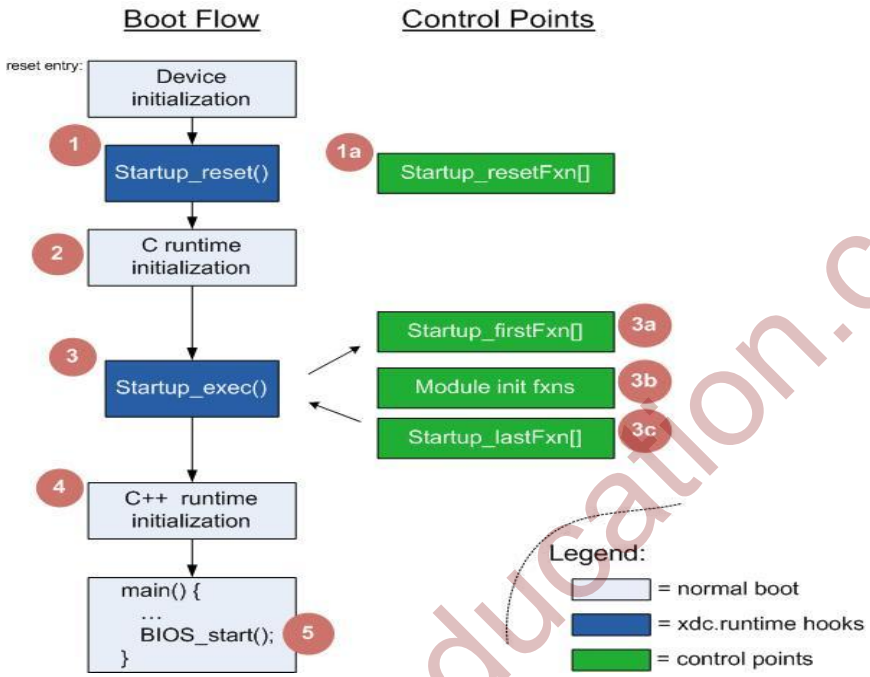

Fig 5

Storage Allocation for Arrays, Strings and Records:

- In computer programming, a string is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation).
- A string is generally understood as a data type and is often implemented as an array of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. A string may also denote more general arrays or other sequence (or list) data types and structures.
- Depending on programming language and precise data type used, a variable declared to be a string may either cause storage in memory to be statically allocated for a predetermined maximum length or employ dynamic allocation to allow it to hold variable number of elements.
- When a string appears literally in source code, it is known as a string literal or an anonymous string.

- In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set called an alphabet. An array is a systematic arrangement of similar objects, usually in rows and columns.

www.sakshieducation.com