Report Of Week- 2

Symbol Table is an important data structure created and maintained by the compiler in order to keep track of semantics of variables.It stores information about the scope and binding information about names, information about instances of various entities such as variable and function names, classes, objects, etc.It is built in lexical and syntax analysis phases.The information is collected by the analysis phases of the compiler and is used by the synthesis phases of the compiler to generate code.It is used by the compiler to achieve compile-time efficiency.

It is used by various phases of the compiler as follows:-

i) **Lexical Analysis:** Creates new table entries in the table, for example like entries about tokens.

ii) **Syntax Analysis:** Adds information regarding attribute type, scope, dimension, line of reference, use, etc in the table.

iii) **Semantic Analysis:** Uses available information in the table to check for semantics i.e. to verify that expressions and assignments are semantically correct(type checking) and update it accordingly.

iv) **Intermediate Code generation:** Refers symbol table for knowing how much and what type of run-time is allocated and table helps in adding temporary variable information.

v) **Code Optimization:** Uses information present in the symbol table for machine-dependent optimization.

vi) **Target Code generation:** Generates code by using address information of identifier present in the table.

Items stored in Symbol table:

- Variable names and constants
- Procedure and function names
- Literal constants and strings
- Compiler generated temporaries
- Labels in source languages

Information used by the compiler from Symbol table:

- Data type and name
- Declaring procedures

- Offset in storage
- If structure or record then, a pointer to structure table.
- For parameters, whether parameter passing by value or by reference
- Number and type of arguments passed to function
- Base Address

Implementation of Symbol table :

We used Linked List and Hash Table to implement the symbol table in Week 2 Assignment.

i) **Linked List** –

This implementation is using a linked list. A link field is added to each record.

Searching of names is done in order pointed by the link of the link field.

A pointer "**head**" is maintained to point to the first record of the symbol table.

Insertion is fast O(1), but lookup is slow for large tables – O(n) on average.

We have maintained a linked list and used the insert, modify, delete and find function to perform certain operations in the symbol table.

ii) **Hash Table** –

In hashing scheme, two tables are maintained – a hash table and symbol table and are the most commonly used method to implement symbol tables.

A hash table is an array with an index range: 0 to table size – 1. These entries are pointers pointing to the names of the symbol table.

To search for a name we use a hash function that will result in an integer between 0 to table size – 1.

Insertion and lookup can be made very fast – O(1).

The advantage is quick to search is possible and the disadvantage is that hashing is complicated to implement.

We have used a "**hashf**" function to find the index of the given string and then we maintain a linked list for a given index and used the insert, modify, delete and find function to perform certain operations in the symbol table.

The code is written in C++ language.

Members

Jayesh Jethy

Achyut Katiyar

Vishal Kumar Singh

Challa Aditya koundinya

Kaushik Aadhithya ch

D chandra sai reddy

Akanksha kailash Patil