Compiler Design Lab (CS 306L)

Week 2: Symbol Table Implementation

1. A symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various identifiers such as variable names, function names, objects, classes, interfaces, etc. The symbol table is used by both the analysis and the synthesis parts of a compiler. Symbol table can be implemented in one of the following ways:
   - Linear (sorted or unsorted) list
   - Binary Search Tree
   - Hash table
   - And other ways.

CODE:

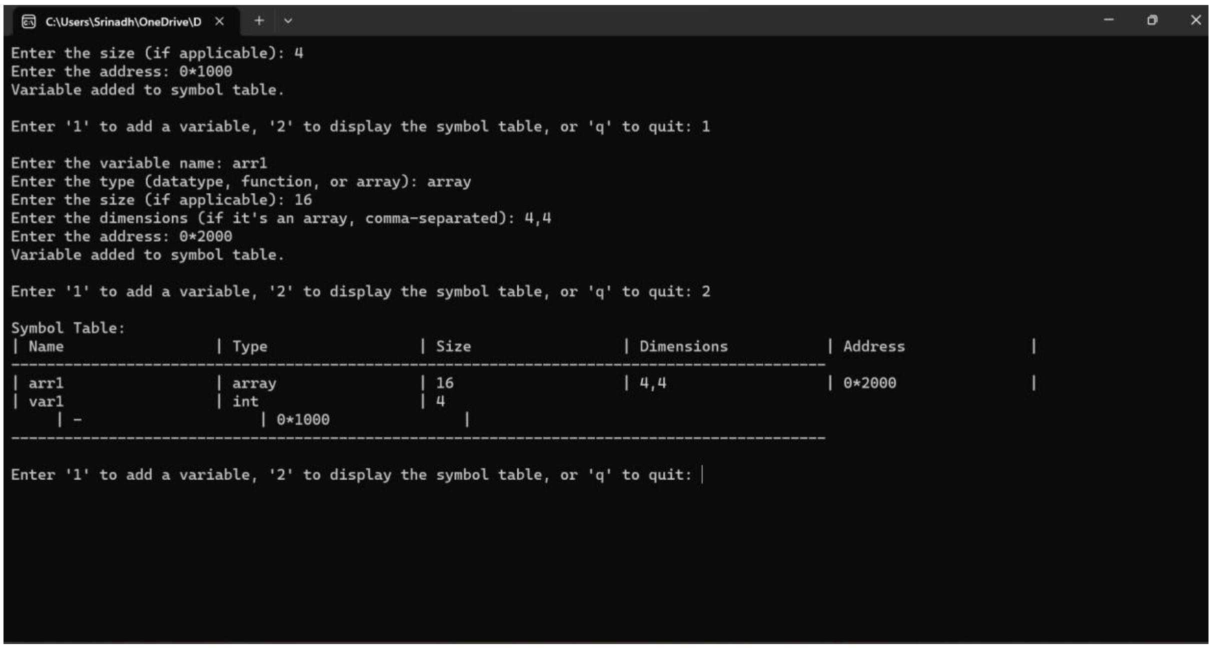
```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define HASH_TABLE_SIZE 100


struct Variable {     char

name[50];     char

data_type[50];     char

size[50];     char

dimensions[50];     char

address[50];     struct

Variable* next;

};
```

```
struct Variable* hash_table[HASH_TABLE_SIZE];

unsigned int hash(const char* name) {
unsigned int hash_value = 0;     while (*name)
{         hash_value = (hash_value << 5) +
*name;         name++;
   }
   return hash_value % HASH_TABLE_SIZE;
}

void add_variable(char name[], char data_type[], char size[], char dimensions[], char
address[]) {     struct Variable* new_variable = (struct Variable*)malloc(sizeof(struct
Variable));     strcpy(new_variable->name, name);     strcpy(new_variable-
>data_type, data_type);     strcpy(new_variable->size, size);     strcpy(new_variable-
>dimensions, dimensions);     strcpy(new_variable->address, address);
new_variable->next = NULL;

   unsigned int index = hash(name);     new_variable-
>next = hash_table[index];     hash_table[index] =
new_variable;
}

void display_symbol_table() {
printf("\nSymbol Table:\n");
   printf("| %-20s | %-20s | %-20s | %-20s | %-20s |\n", "Name", "Type", "Size",
"Dimensions", "Address");
   printf("---------------------------------------------------------------------------------------------\n");
for (int i = 0; i < HASH_TABLE_SIZE; i++) {
        struct Variable* current = hash_table[i];         while
(current != NULL) {             printf("| %-20s | %-20s | %-
20s | %-20s | %-20s |\n",
                   current->name, current->data_type, current->size, current->dimensions, current-
>address);             current =
current->next;
        }     }     printf("--------------------------------------------------------------------------------
--------\n"); }

int main() {     char choice;     char name[50], data_type[50], size[50],
dimensions[50], address[50];

    while (1) {         printf("\nEnter '1' to add a variable, '2' to display the symbol table,
or 'q' to quit: ");         scanf(" %c", &choice);

        switch (choice) {
            case '1':
                printf("\nEnter the variable name: ");                 scanf("%s", name);
printf("Enter the type (datatype, function, or array): ");                 scanf("%s",
data_type);                 printf("Enter the size (if applicable): ");
scanf("%s", size);                 if (strcmp(data_type, "array") == 0) {
printf("Enter the dimensions (if it's an array, comma-separated): ");
scanf("%s", dimensions);
                } else {
                    strcpy(dimensions, "-");
                }
                printf("Enter the address: ");
scanf("%s", address);
```

```
                add_variable(name, data_type, size, dimensions, address);
printf("Variable added to symbol table.\n");
                break;
             case
'2':
                display_symbol_table();
break;
             case
'q':
                for (int i = 0; i < HASH_TABLE_SIZE; i++) {
struct Variable* current = hash_table[i];
while (current != NULL) {                       struct
Variable* temp = current;                       current =
current->next;                        free(temp);
                    }
}                 return
0;

default:
                printf("Invalid choice. Try again.\n");
        }
    }
}
```

<u>OUTPUT:</u>

- Using Hash Table



```
Enter the size (if applicable): 4
Enter the address: 0*1000
Variable added to symbol table.

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit: 1

Enter the variable name: arr1
Enter the type (datatype, function, or array): array
Enter the size (if applicable): 16
Enter the dimensions (if it's an array, comma-separated): 4,4
Enter the address: 0*2000
Variable added to symbol table.

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit: 2

Symbol Table:
| Name                 | Type                 | Size                 | Dimensions           | Address              |
---------------------------------------------------------------------------------------------
| arr1                 | array                | 16                   | 4,4                  | 0*2000               |
| var1                 | int                  | 4
     | -                     | 0*1000                |
---------------------------------------------------------------------------------------------

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit:
```

Using Hash Table we have implemented this code First we need to enter the variable name, datatype, size, and address. If it is the Function we need to mention the dimension of the code.

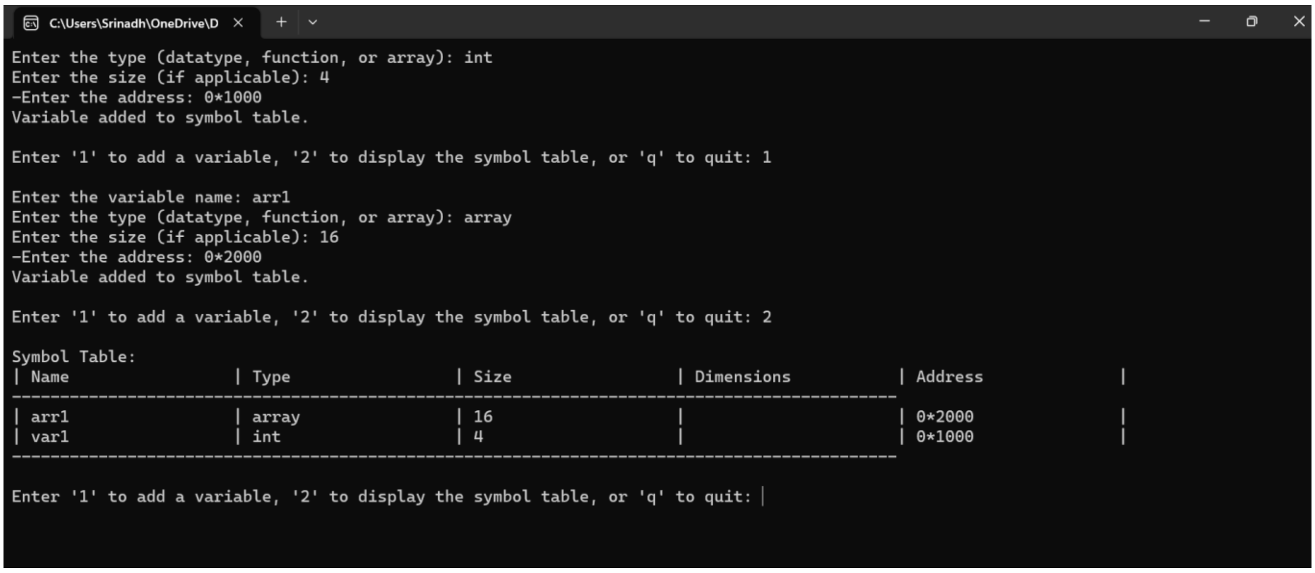
- Using LinkedList

CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



struct Variable {     char

name[50];     char

data_type[50];     char

size[50];     char

dimensions[50];     char

address[50];     struct

Variable* next;

};
```

```
struct Variable* head = NULL;


void add_variable(char name[], char data_type[], char size[], char dimensions[], char
address[]) {     struct Variable* new_variable = (struct Variable*)malloc(sizeof(struct
Variable));     strcpy(new_variable->name, name);     strcpy(new_variable-
>data_type, data_type);     strcpy(new_variable->size, size);     strcpy(new_variable-
>dimensions, dimensions);     strcpy(new_variable->address, address);
new_variable->next = head;     head = new_variable;
}

void display_symbol_table() {
struct Variable* current = head;
printf("\nSymbol Table:\n");
   printf("| %-20s | %-20s | %-20s | %-20s | %-20s |\n", "Name", "Type", "Size",
"Dimensions", "Address");
   printf("------------------------------------------------------------------------------------------\n");
   while (current != NULL) {        printf("| %-20s | %-20s |
%-20s | %-20s | %-20s |\n",
              current->name, current->data_type, current->size, current->dimensions,
current>address);
       current = current->next;
   }
   printf("------------------------------------------------------------------------------------------\n"); }

int main() {     char choice;     char name[50], data_type[50], size[50],
dimensions[50], address[50];
```

```
   while (1) {        printf("\nEnter '1' to add a variable, '2' to display the symbol table, or
'q' to quit: ");        scanf(" %c", &choice);

       switch (choice) {
           case '1':
               printf("\nEnter the variable name: ");
scanf("%s", name);                printf("Enter the type (datatype,
function, or array): ");                scanf("%s", data_type);
printf("Enter the size (if applicable): ");                scanf("%s",
size);                if(data_type == "array"){
                          printf("Enter the dimensions (if it's an array, comma-separated): ");
scanf("%s", dimensions);
                                                  }
                                                  else{
                                                              printf("-");
                                                  }
               printf("Enter the address: ");
               scanf("%s", address);

               add_variable(name, data_type, size, dimensions, address);
printf("Variable added to symbol table.\n");
               break;

           case '2':
               display_symbol_table();
               break;

           case 'q':
            while (head != NULL) {
struct Variable* temp = head;
head = head->next;
free(temp);
            }
            return 0;

         default:
            printf("Invalid choice. Try again.\n");
      }
   }
}
```



```
Enter the type (datatype, function, or array): int
Enter the size (if applicable): 4
-Enter the address: 0*1000
Variable added to symbol table.

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit: 1

Enter the variable name: arr1
Enter the type (datatype, function, or array): array
Enter the size (if applicable): 16
-Enter the address: 0*2000
Variable added to symbol table.

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit: 2

Symbol Table:
| Name                | Type                | Size                | Dimensions          | Address             |
-------------------------------------------------------------------------------------------
| arr1                | array               | 16                  |                     | 0*2000              |
| var1                | int                 | 4                   |                     | 0*1000              |
-------------------------------------------------------------------------------------------

Enter '1' to add a variable, '2' to display the symbol table, or 'q' to quit:
```

<u>SRINADH DOPPALAPUDI</u>

<u>AP21110010951</u>

<u>CSE-O</u>