Aim: - Write a program to implement symbol table.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define a structure for a symbol

typedef struct

{

    char name[50]; // Variable name

    char type[10]; // Variable type (int, float, etc.)

    int size;      // Size of the variable (based on its type)

    int scope;     // Scope level

    int address;   // Memory address (just for simulation)

} Symbol;

// Define the maximum number of symbols in the table

#define MAX\_SYMBOLS 100

// Symbol table

Symbol symbolTable[MAX\_SYMBOLS];

int symbolCount = 0;

// Function to add a symbol to the symbol table

void addSymbol(char \*name, char \*type, int scope, int address)

{

    if (symbolCount >= MAX\_SYMBOLS)

    {

        printf("Error: Symbol table is full!\n");

        return;

    }

    // Check if the symbol already exists

    for (int i = 0; i < symbolCount; i++)

    {

        if (strcmp(symbolTable[i].name, name) == 0 && symbolTable[i].scope == scope)

        {

            printf("Error: Symbol '%s' already exists in the current scope!\n", name);

            return;

        }

    }

    // Create a new symbol

    strcpy(symbolTable[symbolCount].name, name);

    strcpy(symbolTable[symbolCount].type, type);

    // Set the size based on the type

    if (strcmp(type, "int") == 0)

    {

        symbolTable[symbolCount].size = sizeof(int);

    }

    else if (strcmp(type, "float") == 0)

    {

        symbolTable[symbolCount].size = sizeof(float);

    }

    else if (strcmp(type, "char") == 0)

    {

        symbolTable[symbolCount].size = sizeof(char);

    }

    else

    {

        printf("Error: Unsupported data type '%s'\n", type);

        return;

    }

    symbolTable[symbolCount].scope = scope;

    symbolTable[symbolCount].address = address;

    symbolCount++;

}

// Function to search for a symbol in the table by name and scope

int searchSymbol(char \*name, int scope)

{

    for (int i = 0; i < symbolCount; i++)

    {

        if (strcmp(symbolTable[i].name, name) == 0 && symbolTable[i].scope == scope)

        {

            return i;

        }

    }

    return -1; // Symbol not found

}

// Function to display the symbol table

void displaySymbolTable()

{

    printf("\n%-10s %-10s %-5s %-10s %-10s\n", "Name", "Type", "Size", "Scope", "Address");

    printf("--------------------------------------------------------\n");

    for (int i = 0; i < symbolCount; i++)

    {

        printf("%-10s %-10s %-5d %-10d %-10d\n",

               symbolTable[i].name,

               symbolTable[i].type,

               symbolTable[i].size,

               symbolTable[i].scope,

               symbolTable[i].address);

    }

}

// Main function to demonstrate the symbol table operations

int main()

{

    int option;

    char name[50], type[10];

    int scope, address;

    while (1)

    {

        printf("\nSymbol Table Operations\n");

        printf("1. Add Symbol\n");

        printf("2. Search Symbol\n");

        printf("3. Display Symbol Table\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        if (scanf("%d", &option) != 1) {

            printf("Invalid input! Please enter a valid number.\n");

            while (getchar() != '\n'); // Clear the input buffer

            continue;

        }

        switch (option)

        {

            case 1:

                {

                    // Add a new symbol

                    printf("Enter symbol name: ");

                    scanf("%s", name);

                    printf("Enter symbol type (int, float, char): ");

                    scanf("%s", type);

                    printf("Enter scope level: ");

                    if (scanf("%d", &scope) != 1) {

                        printf("Invalid scope input! Please enter a valid integer.\n");

                        while (getchar() != '\n');

                        continue;

                    }

                    printf("Enter memory address: ");

                    if (scanf("%d", &address) != 1) {

                        printf("Invalid memory address! Please enter a valid integer.\n");

                        while (getchar() != '\n');

                        continue;

                    }

                    addSymbol(name, type, scope, address);

                }

                break;

            case 2:

                {

                    // Search for a symbol

                    printf("Enter symbol name to search: ");

                    scanf("%s", name);

                    printf("Enter scope level: ");

                    if (scanf("%d", &scope) != 1) {

                        printf("Invalid scope input! Please enter a valid integer.\n");

                        while (getchar() != '\n');

                        continue;

                    }

                    int index = searchSymbol(name, scope);

                    if (index != -1)

                    {

                        printf("Symbol found at index %d\n", index);

                        printf("Name: %s, Type: %s, Scope: %d, Address: %d\n",

                            symbolTable[index].name, symbolTable[index].type,

                            symbolTable[index].scope, symbolTable[index].address);

                    }

                    else

                    {

                        printf("Symbol not found!\n");

                    }

                }

                break;

            case 3:

                displaySymbolTable();

                break;

            case 4:

                // Exit the program

                exit(0);

            default:

                printf("Invalid choice! Please try again.\n");

                break;

        }

    }

    return 0;

}

Output: -







