EX.NO.4

Date:

Implementation of symbol table

AIM

To implement a symbol table using a C++ program and demonstrate various operations like creating, searching, deleting, and displaying symbols.

ALGORITHM

Step 1: Initialize the program and create a structure SymbolInfo to hold symbol details like label, data type, variable name, value, and memory address.
Step 2: Implement the SymbolTable class which includes a hash table (unordered_map) to store the symbols.
Step 3: In the SymbolTable constructor, initialize the base memory address and set an increment value for addresses.
Step 4: Write a method createSymbol that adds a new symbol to the table if it doesn't already exist. Compute the address for the symbol and increment the base address for the next symbol.
Step 5: Write a method searchSymbol to search for a variable in the table. Display its details if found.
Step 6: Write a method deleteSymbol to remove a variable from the symbol table.
Step 7: Implement a method displayTable to display the entire symbol table. Use proper formatting to print the details.
Step 8: In the main function, prompt the user for the starting memory address and provide a menu to perform the operations (create, search, delete, display).
Step 9: Loop until the user chooses to exit.

PROGRAM

```
#include <iostream>

#include <unordered_map>

#include <string>

#include <iomanip>

using namespace std;


struct SymbolInfo {

   string label;

   string datatype;

   string varName;

   string value;
```

```
    string address;

    SymbolInfo(string lbl = "", string dt = "", string vn = "", string val = "", string addr = "")
        : label(lbl), datatype(dt), varName(vn), value(val), address(addr) {}
};

class SymbolTable {
private:
    unordered_map<string, SymbolInfo> table;
    int baseAddress;
    int addressIncrement;

public:
    SymbolTable(int startAddress) {
        baseAddress = startAddress;
        addressIncrement = 4;
    }

    void createSymbol(const string& label, const string& datatype, const string& varName, const string& value) {
        if (table.find(varName) != table.end()) {
            cout << "Error: Variable '" << varName << "' already exists." << endl;
        } else {
            string address = "0x" + to_string(baseAddress);
            table[varName] = SymbolInfo(label, datatype, varName, value, address);
            baseAddress += addressIncrement;
            cout << "Symbol '" << varName << "' created.\n";
        }
    }

    void searchSymbol(const string& varName) {
```

```
    if (table.find(varName) != table.end()) {
        SymbolInfo& sym = table[varName];
        cout << "\nSymbol Found: \n";
        cout << "Label: " << sym.label << ", Datatype: " << sym.datatype << ", Variable: " << sym.varName
             << ", Value: " << sym.value << ", Address: " << sym.address << endl;
    } else {
        cout << "Error: Variable '" << varName << "' not found.\n";
    }
}

void deleteSymbol(const string& varName) {
    if (table.erase(varName)) {
        cout << "Symbol '" << varName << "' deleted.\n";
    } else {
        cout << "Error: Variable '" << varName << "' not found.\n";
    }
}

void displayTable() {
    cout << "\nSymbol Table Contents:\n";
    cout << setw(10) << "Label" << setw(15) << "Datatype" << setw(15) << "Variable"
         << setw(10) << "Value" << setw(10) << "Address" << endl;
    cout << "------------------------------------------------------------\n";
    for (const auto& entry : table) {
        const SymbolInfo& sym = entry.second;
        cout << setw(10) << sym.label << setw(15) << sym.datatype << setw(15) << sym.varName
             << setw(10) << sym.value << setw(10) << sym.address << endl;
    }
    cout << "------------------------------------------------------------\n";
}
```

```
};

int main() {
    int startAddress;
    cout << "Enter the starting memory address (in decimal): ";
    cin >> startAddress;

    SymbolTable symTable(startAddress);
    string label, datatype, varName, value;
    int choice;

    do {
        cout << "\nMenu:\n";
        cout << "1. Create a new symbol\n";
        cout << "2. Search for a symbol\n";
        cout << "3. Delete a symbol\n";
        cout << "4. Display symbol table\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter label: ";
                cin >> label;
                cout << "Enter data type (int, float, etc.): ";
                cin >> datatype;
                cout << "Enter variable name: ";
                cin >> varName;
                cout << "Enter value: ";
```

```
            cin >> value;
            symTable.createSymbol(label, datatype, varName, value);
            break;

        case 2:
            cout << "Enter variable name to search: ";
            cin >> varName;
            symTable.searchSymbol(varName);
            break;
        case 3:
            cout << "Enter variable name to delete: ";
            cin >> varName;
            symTable.deleteSymbol(varName);
            break;
        case 4:
            symTable.displayTable();
            break;
        case 5:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice! Please try again.\n";
            break;
      }
   } while (choice != 5);

   return 0;
}
```

OUTPUT

```
Menu:
1. Create a new symbol
2. Search for a symbol
3. Delete a symbol
4. Display symbol table
5. Exit
Enter your choice: 1
Enter label: y
Enter data type (int, float, etc.): int
Enter variable name: y
Enter value: 10
Symbol 'y' created.

Menu:
1. Create a new symbol
2. Search for a symbol
3. Delete a symbol
4. Display symbol table
5. Exit
Enter your choice: 4

Symbol Table Contents:
     Label       Datatype       Variable     Value   Address
-------------------------------------------------------------
         y            int              y        10    0x1004
         x            int              x         4    0x1000
-------------------------------------------------------------

Menu:
1. Create a new symbol
2. Search for a symbol
3. Delete a symbol
4. Display symbol table
5. Exit
Enter your choice:
```

RESULT

Symbol table implemented successfully, allowing for the creation, searching, deletion, and display of symbols using various operations in a C++ program.