

**Lab Task # 5**

**Submitted By:** Sana Naz (SP21-BCS-031)

**Course Instructor:** Mr. Bilal bukhari

**Course:** Compiler constrcution

**Date:** 25th Octuber, 2024

**Lab Task 1:** Implement symbol table using hash function.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5Task

{

// A class to represent a symbol in the symbol table

class Symbol

{

public string Name { get; set; }

public string Type { get; set; }

public string Scope { get; set; }

public int LineNumber { get; set; }

public Symbol(string name, string type, string scope, int lineNumber)

{

Name = name;

Type = type;

Scope = scope;

LineNumber = lineNumber;

}

public override string ToString()

{

return $"Identifier's Name: {Name}\nType: {Type}\nScope: {Scope}\nLine Number: {LineNumber}";

}

}

// A class to represent the symbol table

class SymbolTable

{

// Use a Dictionary (hash table) to store symbols

private Dictionary<string, Symbol> table;

public SymbolTable()

{

table = new Dictionary<string, Symbol>();

}

// Insert a symbol into the table

public void Insert(string name, string type, string scope, int lineNumber)

{

if (!table.ContainsKey(name))

{

var symbol = new Symbol(name, type, scope, lineNumber);

table.Add(name, symbol);

Console.WriteLine($"{name} inserted -successfully");

}

else

{

Console.WriteLine($"Error: Symbol '{name}' already exists.");

}

}

// Lookup a symbol in the table by name

public Symbol Lookup(string name)

{

if (table.ContainsKey(name))

{

Console.WriteLine("Identifier Is present");

return table[name];

}

else

{

Console.WriteLine($"Error: Identifier '{name}' not found.");

return null;

}

}

// Delete a symbol from the table by name

public void Delete(string name)

{

if (table.ContainsKey(name))

{

table.Remove(name);

Console.WriteLine($"{name} Identifier is deleted");

}

else

{

Console.WriteLine($"Error: Identifier '{name}' not found.");

}

}

// Update a symbol in the table

public void Update(string name, string newType, string newScope, int newLineNumber)

{

if (table.ContainsKey(name))

{

var symbol = table[name];

symbol.Type = newType;

symbol.Scope = newScope;

symbol.LineNumber = newLineNumber;

Console.WriteLine($"{name} Identifier updated");

}

else

{

Console.WriteLine($"Error: Identifier '{name}' not found.");

}

}

// Display all symbols in the table

public void Display()

{

Console.WriteLine("\n\*\*\*\* SYMBOL\_TABLE \*\*\*\*");

foreach (var symbol in table.Values)

{

Console.WriteLine(symbol);

Console.WriteLine();

}

}

}

class Program

{

static void Main(string[] args)

{

// Create a new symbol table

var symbolTable = new SymbolTable();

// Insert symbols

symbolTable.Insert("if", "keyword", "local", 4);

symbolTable.Insert("number", "variable", "global", 3);

// Display the symbol table

symbolTable.Display();

// Lookup 'if' identifier

Console.WriteLine("\nLookup 'if':");

var symbolIf = symbolTable.Lookup("if");

if (symbolIf != null) Console.WriteLine(symbolIf);

// Delete 'if' identifier

symbolTable.Delete("if");

// Update 'number' identifier

symbolTable.Update("number", "variable", "global", 3);

// Lookup 'number' identifier

Console.WriteLine("\nLookup 'number':");

var symbolNumber = symbolTable.Lookup("number");

if (symbolNumber != null) Console.WriteLine(symbolNumber);

// Keep the console window open

Console.ReadLine();

}

}

}

**Output:**

