Lab 04 – Construction of Symbol Table

Name: Pranamya G Kulal
Class: CSE A1
Reg no: 220905018
Roll no: 8

**Q1) Using getNextToken( ) implemented in Lab No 3, design a Lexical Analyser to implement the
following symbol tables.
a. local symbol table
b. global symbol table**

```
#include "la.h"

struct node {
    char lexeme[20];
    struct node* next;
};

int cnt=1;
struct entry {
    int index;
    char lexeme[20];
    char type[10];
    char dtype[10];
    int size;
};

const char *Datatypes[] = {"int", "char", "float"};
struct entry symbolTable[100];
int entryCount = 0;

unsigned long hash(unsigned char *str) {
    unsigned long hash = 5381;
    int c;
    while ((c = *str++))
        hash = ((hash << 5) + hash) + c;
    return (hash % 100);
}

struct node* HashMap[100] = {NULL};

void insert(char* str) {
    int hashVal = hash(str);
    struct node* temp = (struct node*)malloc(sizeof(struct node));
    strcpy(temp->lexeme, str);
    temp->next = HashMap[hashVal];
    HashMap[hashVal] = temp;
}

int search(char* str) {
    int hashVal = hash(str);
    struct node* temp = HashMap[hashVal];
    while (temp != NULL) {
        if (strcmp(temp->lexeme, str) == 0) {
            return 1;
        }
        temp = temp->next;
    }
    return 0;
}

int retSize(const char* dtype) {
    if (strcmp(dtype, "int") == 0) return 4;
    if (strcmp(dtype, "char") == 0) return 1;
    if (strcmp(dtype, "float") == 0) return 4;
    if (strcmp(dtype, "FUNC") == 0) return -1;
    return 0;
}

int isDtype(const char* str) {
    for (int i = 0; i < sizeof(Datatypes) / sizeof(char*); i++) {
        if (strcmp(str, Datatypes[i]) == 0)
            return 1;
    }
    return 0;
}

int main() {
    FILE* fin = fopen("sampleread.c", "r");
    if (!fin) {
        printf("Error! File cannot be opened!\n");
        return 0;
    }

    struct token tkn;
    char currentDtype[10] = "Void";
    int i = 1;

    while ((tkn = getNextToken(fin)).row != -1) {
        printf("%d. <%s, %d, %d>\n", cnt++, tkn.type, tkn.row, tkn.col);

        struct entry tuple;
        tuple.index = i++;
        strcpy(tuple.lexeme, tkn.lexeme);
        strcpy(tuple.type, tkn.type);

        if (strcmp(tkn.type, "Keyword") == 0 && isDtype(tkn.lexeme)) {
            strcpy(currentDtype, tkn.lexeme);
            strcpy(tuple.dtype, "Void");
            tuple.size = 0;
        } else if (strcmp(tkn.type, "Identifier") == 0) {
            strcpy(tuple.dtype, currentDtype);
            tuple.size = retSize(currentDtype);
        } else {
            strcpy(tuple.dtype, "N/A");
            tuple.size = 0;
        }

        if (search(tuple.lexeme) == 0) {
            symbolTable[entryCount++] = tuple;
            insert(tuple.lexeme);
        }

        if (strcmp(tkn.lexeme, ";") == 0) {
            strcpy(currentDtype, "Void");
        }
    }

    printf("\n\n\t\t\tSYMBOL TABLE\n\n");
    printf("Index, Lexeme, Type, Dtype, Size\n");
    for (int i = 0; i < entryCount; i++) {
        printf("%d, %s, %s, %s, %d\n",
               symbolTable[i].index,
               symbolTable[i].lexeme,
               symbolTable[i].type,
               symbolTable[i].dtype,
               symbolTable[i].size);
    }

    fclose(fin);
    return 0;
}
```

Terminal output:

```
CD_LAB_A1@debianpc-02:~/Desktop/220905018/Lab4$ gcc -o l4q1 l4q1.c
CD_LAB_A1@debianpc-02:~/Desktop/220905018/Lab4$ ./l4q1
1. <Keyword, 1, 20>
2. <Identifier, 1, 24>
3. <(, 1, 28>
4. <), 1, 29>
5. <{, 1, 30>
6. <Keyword, 1, 36>
7. <Identifier, 1, 40>
8. <=, 1, 44>
9. <Number, 1, 46>
10. <;, 1, 48>
11. <Keyword, 1, 54>
12. <(, 1, 56>
13. <Identifier, 1, 57>
14. <==, 1, 61>
15. <Number, 1, 64>
16. <), 1, 66>
17. <Identifier, 1, 68>
18. <(, 1, 74>
19. <StringLiteral, 1, 75>
20. <), 1, 88>
21. <;, 1, 89>
22. <Keyword, 1, 95>
23. <Keyword, 1, 100>
24. <(, 1, 102>
25. <Identifier, 1, 103>
26. <<=, 1, 107>
27. <Number, 1, 110>
28. <), 1, 112>
29. <Identifier, 1, 114>
30. <(, 1, 120>
31. <StringLiteral, 1, 121>
32. <), 1, 137>
33. <;, 1, 138>
34. <Keyword, 1, 144>
35. <Number, 1, 151>
36. <;, 1, 152>
37. <}, 1, 154>
```

SYMBOL TABLE

```
Index, Lexeme, Type, Dtype, Size
1, int, Keyword, Void, 0
2, main, Identifierint, int, 4
3, (, (, N/A, 0
4, ), ), N/A, 0
5, {, {, N/A, 0
7, num, Identifierint, int, 4
8, =, =, N/A, 0
9, 18, Number, N/A, 0
10, ;, ;, N/A, 0
11, if, Keyword, N/A, 0
17, printf, IdentifierVoid, Void, 0
19, "age equals 18", StringLiteN/A, N/A, 0
22, else, Keyword, N/A, 0
26, <, <=, N/A, 0
31, "age less than 18", StringLiteN/A, N/A, 0
34, return, Keyword, N/A, 0
35, 0, Number, N/A, 0
37, }, }, N/A, 0
```