Principals of Compiler Design

(COCSC14 -III)

Submitted By:

Name – Kartikeya Agarwal

Roll No. – 2019UCO1692

Class – COE Section 3

INDEX

PROGRAM 1: IMPLEMENT SYMBOL TABLE IN C/C++

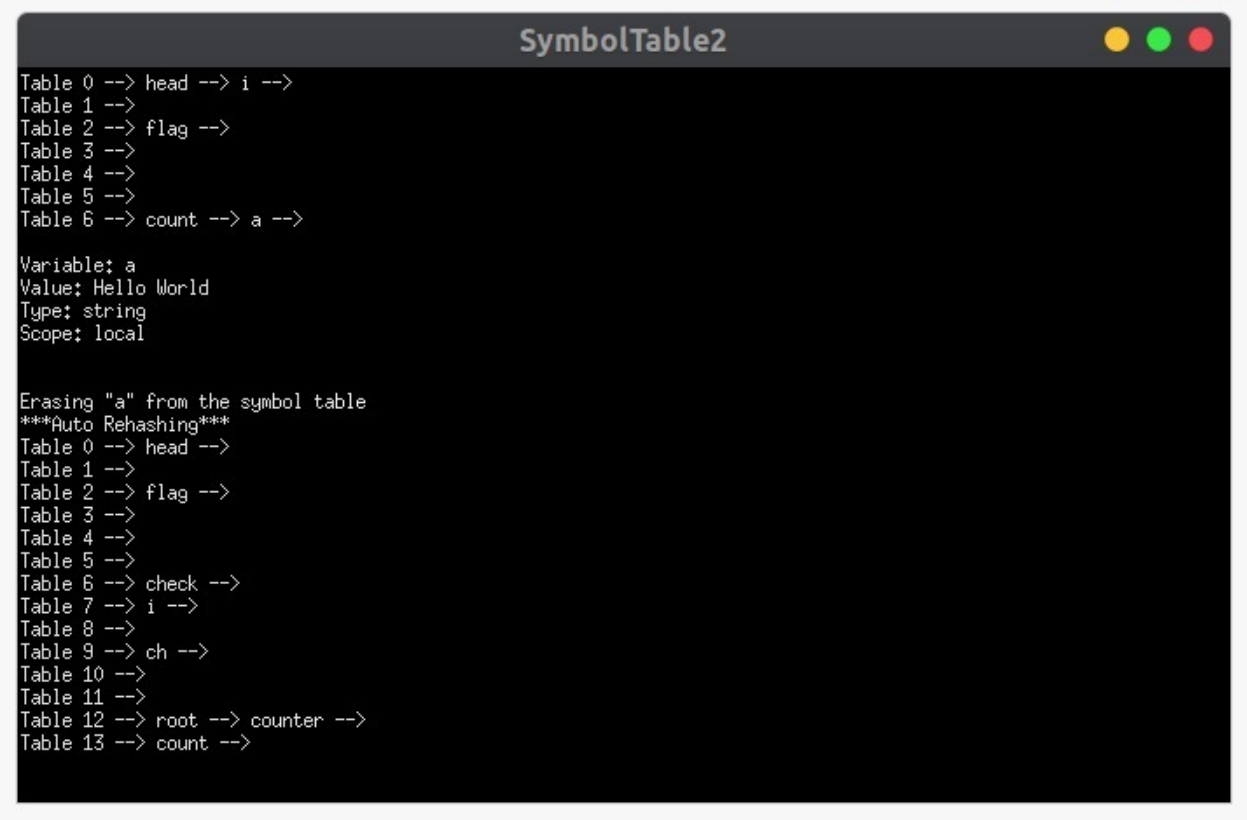
SYMBOL TABLE

```
#include <iostream>
#include<string>
using namespace std;
class Node{
public:
    string name;
    string scope;
    string type;
    string value;
    Node* next;
    Node(string name, string scope, string type, string value){
        this->name = name;
        this->scope = scope;
        this->type = type;
        this->value = value;
        next = NULL;
    }
    ~Node(){
        if(next)
            delete next;
    }
};

class SymbolTable{
private:
    Node** table;
    int cs;
    int ts;

    int hashFn(string key){
        int idx = 0;
        int p = 1;
        for(int j=0;j<key.length();j++){
            idx = idx+(key[j]*p)%ts;
            idx = idx%ts;
            p = (p*27)%ts;
        }
        return idx;
    }
    void rehash(){
```

```
        Node **old = table;
        ts = 2*ts;
        table = new Node*[ts];

        for(int i=0;i<ts;i++){
            table[i] = NULL;
        }
        cs = 0;

        for(int i=0;i<(ts/2);i++){
            Node*temp = old[i];
            while(temp!=NULL){
                insert(temp->name,temp->scope,temp->type,temp->value);
                temp=temp->next;
            }
            if(old[i]!=NULL){
                delete old[i];
            }
        }
        delete [] old;
    }
public:
    SymbolTable(int ts = 7){
        this->ts = ts;
        table = new Node*[ts];
        cs = 0;
        for(int i=0;i<ts;i++)
            table[i] = NULL;
    }
    void insert(string name, string scope, string type, string value){
        int idx = hashFn(name);
        Node *n = new Node(name, scope, type, value);
        n->next = table[idx];
        table[idx] = n;
        cs++;
        float lf = cs/(1.0*ts);
        if(lf>0.8){
            cout<<"***Auto Rehashing***\n";
            rehash();
        }
    }
    void print(){
        for(int i=0;i<ts;i++){
            cout<<"Table "<<i<<" --> ";
            Node *temp = table[i];
```

```
            while(temp!=NULL){
                cout<<temp->name<<" --> ";
                temp = temp->next;
            }
            cout<<endl;
        }
        cout<<endl;
    }
    Node* Search(string name, bool print = true){
        int idx = hashFn(name);
        Node *temp = table[idx];
        while(temp!=NULL){
            if(temp->name==name){
                if(print){
                    cout<<"Variable: "<<temp->name<<endl;
                    cout<<"Value: "<<temp->value<<endl;
                    cout<<"Type: "<<temp->type<<endl;
                    cout<<"Scope: "<<temp->scope<<endl;
                }
                return temp;
            }
            temp=temp->next;
        }
        return NULL;
    }
    void erase(string key){
        if(Search(key, false)!=NULL){
            int idx = hashFn(key);
            Node* temp = table[idx];
            if(temp->name == key){
                table[idx] = temp->next;
                delete temp;
                cs--;
                return;
            }
            while(temp->next!=NULL){
                if(temp->next->name==key)
                    break;
                temp=temp->next;
            }
            if(temp->next == NULL){
                Node* prev = temp;
                prev->next = NULL;
                temp = temp->next;
                delete temp;
```

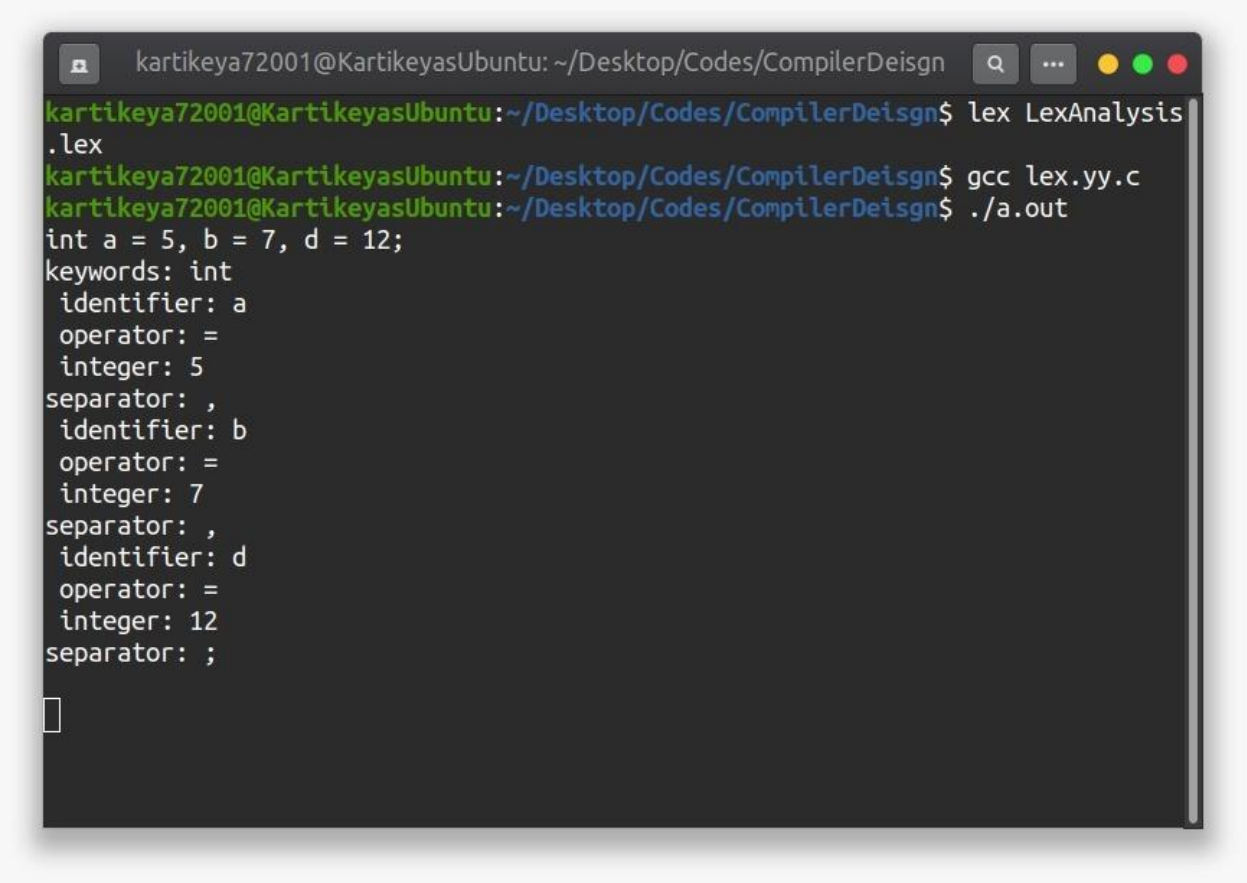
```
                cs--;
                return;
            }
            Node* prev = temp;
            temp=temp->next;
            prev->next = temp->next;
            delete temp;
            cs--;
            return;
        }
    }
};

int main(int argc, char const *argv[]) {
    SymbolTable symb;

    symb.insert("a","local","string","Hello World");
    symb.insert("i","global","int","10");
    symb.insert("head","local","Node*","NULL");
    symb.insert("count","global","int","0");
    symb.insert("flag","local","bool","false");
    symb.print();
    Node* symbObj = symb.Search("a");
    if(symbObj==NULL){
        cout<<"Not Found";
    }
    cout<<"\n\n";
    cout<<"Erasing \"a\" from the symbol table\n";
    symb.erase("a");
    symb.insert("root","local","Node*","0x7ffe67fcdb24");
    symb.insert("counter","local","int","-1");
    symb.insert("check","local","bool","true");
    symb.insert("ch","global","char","g");
    symb.print();
    return 0;
}
```


SymbolTable2
Table 0 --> head --> i -->
Table 1 -->
Table 2 --> flag -->
Table 3 -->
Table 4 -->
Table 5 -->
Table 6 --> count --> a -->
Variable: a
Value: Hello World
Type: string
Scope: local
Erasing "a" from the symbol table
Auto Rehashing
Table 0 --> head -->
Table 1 -->
Table 2 --> flag -->
Table 3 -->
Table 4 -->
Table 5 -->
Table 6 --> check -->
Table 7 --> i -->
Table 8 -->
Table 9 --> ch -->
Table 10 -->
Table 11 -->
Table 12 --> root --> counter -->
Table 13 --> count -->

PROGRAM 2: WRITE A LEX PROGRAM TO RECOGNIZE THE KEYWORDS AND IDENTIFIERS IN THE INPUT C PROGRAM.

LEXICAL ANALYSIS

```
%{
#include<stdio.h>
#include<string.h>
int n = 0;
%}
%%
"while"|"if"|"else" {n++;printf("keywords: %s\n", yytext);}
"int"|"float" {n++;printf("keywords: %s\n", yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {n++;printf("identifier: %s\n", yytext);}
"<="|"=="|"="|"++"|"-"|"*"|"+" {n++;printf("operator: %s\n", yytext);}
[(){}|,;]    {n++;printf("separator: %s\n", yytext);}
[0-9]*"."[0-9]+ {n++;printf("float: %s\n", yytext);}
[0-9]+ {n++;printf("integer: %s\n", yytext);}
.;
%%

int yywrap(void){}

int main(){
    yylex();
    {printf("Number of tokens: %d", n);}
    return 0;
}
```


kartikeya72001@KartikeyasUbuntu: ~/Desktop/Codes/CompilerDeisgn
kartikeya72001@KartikeyasUbuntu:~/Desktop/Codes/CompilerDeisgn$ lex LexAnalysis
.lex
kartikeya72001@KartikeyasUbuntu:~/Desktop/Codes/CompilerDeisgn$ gcc lex.yy.c
kartikeya72001@KartikeyasUbuntu:~/Desktop/Codes/CompilerDeisgn$ ./a.out
int a = 5, b = 7, d = 12;
keywords: int
identifier: a
operator: =
integer: 5
separator: ,
identifier: b
operator: =
integer: 7
separator: ,
identifier: d
operator: =
integer: 12
separator: ;