Name: Mohammad Awais

Class: BSCS-8-A

CMS: 242554

Compiler Construction Lab 3

Tasks

Code:

```
#include <iostream>
#include <list>
#include <fstream>
#include <algorithm>
#include <bits/stdc++.h>
#include <ctime>

using namespace std;

// ================= SYMBOL-TABLE ============================

//---- some pre-requirements for Symbol Table
class Dictionary{ // CLass for Dictionary Objedct
public:
string value;
int hash;
};

list<Dictionary> hash_list;

int get_hash(string value, int rank){ // Used to get Hash to enter in Symbol-Table
for (Dictionary d: hash_list){
if(d.value == value){
return d.hash;
}
}

// Calculated Random Hash -> Append List -> Return random hash
Dictionary d;
d.value = value;
d.hash = rank;
hash_list.push_back(d);

return rank;
}

// ---------- For Symbol Table, Primary Requirement
```

```
class Token{
public:
string name;
string value;
int hash;
};

list<Token> symbol_table;

void symbol_table_show(){
cout<<"\n\t |\t S Y M B O L \t T A B L E \t|\n";
cout<<"\n<TOKEN_NAME>\t\t<TOKEN_VALUE>\t\t<HASH_VALUE>\n\n";
for (Token token: symbol_table){
cout<<" "<<token.name<<"\t\t\t "<<token.value<<"\t\t\t "<<token.hash<<'\n';
}
}

// ------------------------------------------------

bool inRange(int low,int high, int n){ // USER defined self purpose
if(n>=low && n<=high){
return true;
}
else
return false;
}

string commentLess(ifstream *file); // Function to remove comments

list<string> lexer_sep(ifstream *file); // Function to Separate words from File

void lexer_seg(list<string> words); // Function to Segment separated words

// Source Language Specificaitons
string keywords[] = {"break","case","char","const","continue","default", "double", "else",
"enum", "extern", "float", "for", "goto", "if", "int", "long", "return", "short", "static", "struct",
"switch", "void", "while", "#include"};

string arithmetics[] = {"+","-","*","/","%","++","--"};

string relationals[] = {"==","!=",">","<",">=","<="};

string punctuators[] = {"{","}","(",")","[","]","=",",",".",";",":"};

int identifier_number(string word); // DFA implementation to identify
Identifier/Number(int/float)


// Reading File and lexer_seg Analysis
int main(){
string filename = "leapyear.c";
```

```
ifstream file(filename);


// Separating Words
list<string> words = lexer_sep(&file);

// Segmenting Words
lexer_seg(words);


// Hash-value column
cout<<"\n | HASH - VALUE | ~ ( Column Only )\n";
for (Dictionary d: hash_list){
cout<<"\n "<<d.hash<<" - "<<d.value;
}
cout<<"\n";

// Showing Symbol table
symbol_table_show();


file.close();
}


int identifier_number(string word){
set<char> alphabets =
{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
set <char> ALPHABETS;
for(char alph : alphabets){
ALPHABETS.insert(toupper(alph));
}
set<char> numbers = {'0','1','2','3','4','5','6','7','8','9'};

// DFA CODE - IDENTIFIER
int state = 0;
for (char c : word){
switch(state){
case 0:{
if(!(alphabets.count(c)||ALPHABETS.count(c)||c=='_')){
state=2;
}
else{
state=1;
}
break;
}
case 1:{
if(alphabets.count(c)||ALPHABETS.count(c)||numbers.count(c)||c=='_'){
}
else{
```

```
state=2;
}
break;
}
case 2:{
//Failed State
}
}
if(state==2){
break;
}
}
if(state==1){
return 1;
}

// DFA CODE - NUMBER
state = 0;
for (char c : word){
switch(state){
case 0:{
if(!(numbers.count(c))){
state=4;
}
else{
state=1;
}
break;
}
case 1:{
if(numbers.count(c)){
}
else if(c == '.'){
state = 2;
}
else{
state=4;
}
break;
}
case 2:{
if(!(numbers.count(c))){
state=4;
}
else{
state=3;
}
break;
}
case 3:{
if(numbers.count(c)){
```

```
}
else{
state=4;
}
break;
}
case 4:{
//Failed State
}
}
if(state==4){
break;
}
}
if(state==1 || state ==3){
return 2;
}

return 0;

}

// FOR SEGMENTATION
void lexer_seg(list <string> words){
string detect_key = "KEY";
string detect_punc = "PUNC";
string detect_ariths = "OP-Ar";
string detect_relate = "OP-Re";
string detect_str = "STR";
string detect_identif = "ID";
string detect_num = "NUM";
string detect_others = "OTHER";

int segmented;
Token miso;
int counter=1;
for (string word: words){
segmented = 0;
for(string target:keywords){
if(word.compare(target)==0){
miso.name = detect_key;
miso.value = word;
miso.hash = get_hash(word,1000+counter);
counter++;
symbol_table.push_back(miso);


segmented = 1;
break;
}
}
```

```
if(segmented==0){
for(string target:arithmetics){
if(word.compare(target)==0){
miso.name = detect_ariths;
miso.value = word;
miso.hash = get_hash(word,2000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
break;
}
}
}

if(segmented==0){
for(string target:relationals){
if(word.compare(target)==0 ){
miso.name = detect_relate;
miso.value = word;
miso.hash = get_hash(word,3000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
break;
}
}
}

if(segmented==0){
for(string target:punctuators){
if(word.compare(target)==0){
miso.name = detect_punc;
miso.value = word;
miso.hash = get_hash(word,4000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
break;
}
}
}

// FOR STRINGS?
if(segmented==0){
if(word[0]==""){
```

```
miso.name = detect_str;
miso.value = word;
miso.hash = get_hash(word,5000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
}
}


// IDENTIFIER & NUMBER CHECK
if(segmented==0){
int choice = identifier_number(word);
if(choice==1){
miso.name = detect_identif;
miso.value = word;
miso.hash = get_hash(word,6000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
}
else if(choice == 2){
miso.name = detect_num;
miso.value = word;
miso.hash = get_hash(word,7000+counter);
counter++;

symbol_table.push_back(miso);

segmented = 1;
}
}

if(segmented==0 && word!="\r"){
miso.name = detect_others;
miso.value = word;
miso.hash = get_hash(word,8000+counter);
counter++;

symbol_table.push_back(miso);

}
}

}

list<string> lexer_sep(ifstream *file){
```

```
string iLine = commentLess(file);
list <string> sepline;

char c_shad='\0';
int cState=0;
string tempW="";
for (char c: iLine){
if( (c == ' ' || c == '\n' || c == '\t') && !(inRange(31,32,cState))){
if(!tempW.empty()){
sepline.push_back(tempW);
tempW.clear();
}
cState = 0;
}
else if( (inRange(48,57,int(c)) || inRange(97,122,int(c)) || inRange(65,90,int(c)) ||
int(c)==95 || c == '.' || c=='#') && !(inRange(31,32,cState))){
if(!inRange(4,5,cState)){
if(!tempW.empty()){
sepline.push_back(tempW);
tempW.clear();
}

if((inRange(48,57,int(c)))){
cState = 4;
}
else{
cState = 5;
}
}
else if(((inRange(48,57,int(c)) || c == '.' ) && cState ==4)){
}
else{
if(!tempW.empty() && cState ==4){
sepline.push_back(tempW);
tempW.clear();
}

cState =5;
}


tempW+=c;
}
else if (c == '\'' || c == '"'){
if (inRange(31,32,cState)){
int check = 0;
if(c == '\'' && (cState==31)){
check = 1;
}
else if( c == '"' && (cState==32)){
check = 1;
```

```
}
tempW+=c;
if(check==1)
{
sepline.push_back(tempW);
tempW.clear();
cState =-2;
}
}
else{
if(!tempW.empty()){
sepline.push_back(tempW);
tempW.clear();
}
tempW+=c;
cState = (c == '\'')?31:32;

}
}
else if (inRange(31,32,cState)){
tempW+=c;
}
else if (!inRange(31,32,cState))
{
cState = -1;
if(!tempW.empty() && !(c_shad == c && (c == '+' || c == '-' || c == '='))){
sepline.push_back(tempW);
tempW.clear();
}
tempW+=c;

if(c_shad == c){
c_shad = '\0';
continue;
}
}
c_shad = c;
}


return sepline;
}


string commentLess(ifstream *file){
string commentless;
string iLine;
string tempLine="";
char tempW='\0';
int commentMulti=0;
char c_shadow='\0';
```

```
while(getline(*file,iLine)){
for (char c: iLine){
if(commentMulti==1){
if(c=='/' && c_shadow == '*'){
if(tempW=='/'){
tempW='\0';
}
commentMulti = 0;
}
}
else if(c=='/' && c_shadow!='/' ){
tempW='/';
}
else if(c=='/' && c_shadow=='/'){
if(tempW == '/'){
tempW='\0';
}
tempLine+='\n';
break;
}
else if(c=='*' && c_shadow == '/'){
if(tempW=='/'){
tempW='\0';
}
commentMulti = 1;
}
else{
if(tempW == '/'){
tempLine+=tempW;
tempW='\0';
}
tempLine+=c;
}
c_shadow = c;
}
if(commentMulti==1){
tempLine+='\n';
}
commentless+=tempLine;
tempLine.clear();
}
return commentless;
}
```

Output:

Hash-Value Column

```
scolopendra@scolopendra-bytes:~/Local Disk Egg/University/Debian Semester 7/Compiler Construction/Labs/Lab 3/Code$ gcc task.cc -lstdc++ -o task
scolopendra@scolopendra-bytes:~/Local Disk Egg/University/Debian Semester 7/Compiler Construction/Labs/Lab 3/Code$ ./task

 | HASH - VALUE | ~ ( Column Only )

  1001 - #include
  3002 - <
  8003 - stdio.h
  3004 - >
  1005 - int
  6006 - main
  4007 - (
  4008 - )
  4009 - {
  6011 - year
  4012 - ;
  6013 - printf
  5015 - "Enter a year: "
  6018 - scanf
  5020 - "%d"
  4021 - ,
  8022 - &
  1026 - if
  2029 - %
  7030 - 4
  3031 - ==
  7032 - 0
  7039 - 100
  7048 - 400
  5054 - "%d is a leap year."
  1059 - else
  5062 - "%d is not a leap year."
  4067 - }
  1085 - return
```

Symbol Table (Continued)

```
        |       S Y M B O L     T A B L E       |

<TOKEN_NAME>            <TOKEN_VALUE>           <HASH_VALUE>

  KEY                     #include                      1001
  OP-Re                   <                       3002
  OTHER                   stdio.h                       8003
  OP-Re                   >                       3004
  KEY                     int                     1005
  ID                      main                    6006
  PUNC                    (                       4007
  PUNC                    )                       4008
  PUNC                    {                       4009
  KEY                     int                     1005
  ID                      year                    6011
  PUNC                    ;                       4012
  ID                      printf                        6013
  PUNC                    (                       4007
  STR                     "Enter a year: "                      5015
  PUNC                    )                       4008
  PUNC                    ;                       4012
  ID                      scanf                   6018
  PUNC                    (                       4007
  STR                     "%d"                    5020
  PUNC                    ,                       4021
  OTHER                   &                       8022
  ID                      year                    6011
  PUNC                    )                       4008
  PUNC                    ;                       4012
  KEY                     if                      1026
  PUNC                    (                       4007
  ID                      year                    6011
  OP-Ar                   %                       2029
  NUM                     4                       7030
  OP-Re                   ==                      3031
  NUM                     0                       7032
  PUNC                    )                       4008
  PUNC                    {                       4009
  KEY                     if                      1026
  PUNC                    (                       4007
  ID                      year                    6011
  OP-Ar                   %                       2029
```

```
OP-Ar                   %                       2029
NUM                     400                     7048
OP-Re                   ==                      3031
NUM                     0                       7032
PUNC                    )                       4008
ID                      printf                          6013
PUNC                    (                       4007
STR                     "%d is a leap year."                    5054
PUNC                    ,                       4021
ID                      year                    6011
PUNC                    )                       4008
PUNC                    ;                       4012
KEY                     else                    1059
ID                      printf                          6013
PUNC                    (                       4007
STR                     "%d is not a leap year."                        5062
PUNC                    ,                       4021
ID                      year                    6011
PUNC                    )                       4008
PUNC                    ;                       4012
PUNC                    }                       4067
KEY                     else                    1059
ID                      printf                          6013
PUNC                    (                       4007
STR                     "%d is a leap year."                    5054
PUNC                    ,                       4021
ID                      year                    6011
PUNC                    )                       4008
PUNC                    ;                       4012
PUNC                    }                       4067
KEY                     else                    1059
ID                      printf                          6013
PUNC                    (                       4007
STR                     "%d is not a leap year."                        5062
PUNC                    ,                       4021
ID                      year                    6011
PUNC                    )                       4008
PUNC                    ;                       4012
KEY                     return                          1085
NUM                     0                       7032
PUNC                    ;                       4012
scolopendra@scolopendra-bytes:~/Local Disk Egg/University/Debian Semester 7/Compiler Construction/Labs/Lab 3/Code$
```

Referenced File

leapyear.c

Code > leapyear.c > main()

```c
#include <stdio.h>

/* Program for finding a leap year */

int main()
{
    int year;

    printf("Enter a year: ");
    scanf("%d",&year);

    if(year%4 == 0)
    {
        if( year%100 == 0)
        {
            // year is divisible by 400, hence the year is a leap year
            if ( year%400 == 0)
                printf("%d is a leap year.", year);
            else
                printf("%d is not a leap year.", year);
        }
        else
            printf("%d is a leap year.", year );
    }
    else
        printf("%d is not a leap year.", year);

    return 0;
}
```