Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

CSE 4130
Formal Languages & Compilers Lab

Assignment No: 02

Submitted By:

Name: Mayisha Farzana
ID: 16.02.04.028
Lab Group: A1

Date of Submission: 4th March,2020

Question:

Suppose, we have a C source program scanned and filtered as it was done in Session 1. We now take that modified file as input, and separate the lexemes first. We further recognize and mark the lexemes as different types of tokens like keywords, identifiers, operators, separators, parenthesis, numbers, etc.

Input File:

char c; int x1, x_2; float y1, y2; x1=5; x_2= 10; y1=2.5+x1*45; y2=100.o5-x_2/3; if(y1<=y2) c='y'; else c='n';

Source Code:

```
#include<bits/stdc++.h>

const int N = 100005;
char inp[N + 5];
char modify[N + 5];
void seperate_2()
{
   int n = strlen(inp);
   int ind = 0;
   for (int i = 0; i < n; i++)
   {
      if (inp[i] == '>' && inp[i + 1] == '=')
      {
         modify[ind] = ' ';
         ind++;
         modify[ind] = inp[i];
         ind++;
         modify[ind] = inp[i + 1];
         ind++;
         modify[ind] = ' ';
         ind++;
         i = i + 1;
      }
      else if (inp[i] == '<' && inp[i + 1] == '=')
      {
         modify[ind] = ' ';
         ind++;
         modify[ind] = inp[i];
         ind++;
         modify[ind] = inp[i + 1];
         ind++;
```

```
            modify[ind] = ' ';
            ind++;
            i = i + 1;
        }
        else if (inp[i] == '=' && inp[i + 1] == '=')
        {
            modify[ind] = ' ';
            ind++;
            modify[ind] = inp[i];
            ind++;
            modify[ind] = inp[i + 1];
            ind++;
            modify[ind] = ' ';
            ind++;
            i = i + 1;
        }
        else
        {
            modify[ind] = inp[i];
            ind++;
        }
    }
    modify[ind] = '\0';
    strcpy(inp, modify);
    return ;
}


void seperate_1()
{
    // , ; = + - * / ( ) { }
    int n = strlen(inp);
    int ind = 0;
    for (int i = 0; i < n; i++)
    {
        if (check(inp[i]))
        {
            modify[ind] = ' ';
            ind++;
            modify[ind] = inp[i];
```

```
            ind++;
            modify[ind] = ' ';
            ind++;
        }
        else
        {
            modify[ind] = inp[i];
            ind++;
        }
    }
    modify[ind] = '\0';
    strcpy(inp, modify);
}
bool id(char ar[])
{
    int table [100][256];
    for(int i=0; i<100; i++)
    {
        for( int j=0; j<256; j++)
        {
          table[i][j]=0;
        }
    }
    table[1]['_']=2;
    for( char c='a';c<='z';c++)
    {
        table[1][c]=2;
    }
    for( char c='A';c<='Z';c++)
    {
        table[1][c]=2;
    }
    table[2]['_']=2;
     for( char c='a';c<='z';c++)
    {
        table[2][c]=2;
    }
    for( char c='A';c<='Z';c++)
    {
        table[2][c]=2;
```

```
    }
    for( char c='0';c<='9';c++)
    {
        table[2][c]=2;
    }
    int state = 1;
    int n = strlen(ar);
    for (int i = 0; i < n; i++){
        state = table[state][ar[i]];
    }
    if (state == 2){
        return true;
    }
    return false;
}


bool keyword(char ar[])
{
    int table[100][256];
    for (int i = 0; i < 100; i++)
    {
        for (int j = 0; j < 256; j++)
        {
            table[i][j] = 0;
        }
    }
    table[1]['i'] = 2;
    table[2]['n'] = 3;
    table[3]['t'] = 4;
    int n = strlen(ar);
    int state = 1;
    // int
    for (int i = 0; i < n; i++)
    {
        state = table[state][ar[i]];
    }
    if (state == 4)
    {
        return true;
```

```
}
for (int i = 0; i < 100; i++)
{
   for (int j = 0; j < 256; j++)
   {
      table[i][j] = 0;
   }
}
table[1]['c'] = 2;
table[2]['h'] = 3;
table[3]['a'] = 4;
table[4]['r'] = 5;
state = 1;
for (int i = 0; i < n; i++)
{
   state = table[state][ar[i]];
}
if (state == 5)
{
   return  true;
}
for (int i = 0; i < 100; i++)
{
   for (int j = 0; j < 256; j++)
   {
      table[i][j] = 0;
   }
}
table[1]['f'] = 2;
table[2]['l'] = 3;
table[3]['o'] = 4;
table[4]['a'] = 5;
table[5]['t'] = 6;
state = 1;
for (int i = 0; i < n; i++)
{
   state = table[state][ar[i]];
}
if (state == 6)
{
```

```
    return  true;
}
for (int i = 0; i < 100; i++)
{
    for (int j = 0; j < 256; j++)
    {
        table[i][j] = 0;
    }
}
table[1]['i'] = 2;
table[2]['f'] = 3;
state = 1;
for (int i = 0; i < n; i++)
{
    state = table[state][ar[i]];
}
if (state == 3)
{
    return  true;
}
for (int i = 0; i < 100; i++)
{
    for (int j = 0; j < 256; j++)
    {
        table[i][j] = 0;
    }
}
table[1]['e'] = 2;
table[2]['l'] = 3;
table[3]['s'] = 4;
table[4]['e'] = 5;

state = 1;
for (int i = 0; i < n; i++)
{
    state = table[state][ar[i]];
}
if (state == 5)
{
    return  true;
```

```
    }
    return false;
}

bool parenthesis(char ar[]){
    int table[100][256];
    for (int i=0;i<100;i++)
    {
        for(int j=0;j<256;j++)
        {
            table[i][j]=0;
        }
    }
    table[1]['(']=2;
    table[1][')']=2;

    int state = 1;
    int n = strlen(ar);
    for (int i = 0; i < n; i++){
        state = table[state][ar[i]];
    }

    if (state == 2){
        return true;
    }

    return false;
}


bool _operator(char ar[]){
    int table[100][256];
    for (int i = 0; i < 100; i++){
        for (int j = 0; j < 256; j++){
            table[i][j] = 0;
        }
    }
    table[1]['+'] = 2;
    table[1]['-'] = 2;
    table[1]['*'] = 2;
```

```
    table[1]['/'] = 2;
    table[1]['='] = 2;
    table[1]['<'] = 2;
    table[1]['>'] = 2;
    table[2]['='] = 3;
    int state = 1;
    int n = strlen(ar);
    for (int i = 0; i < n; i++){
        state = table[state][ar[i]];
    }
    if (state == 2 || state==3){
        return true;
    }
    return false;
}

bool seperator(char ar[]){
    int table[100][256];
    for (int i = 0; i < 100; i++){
        for (int j = 0; j < 256; j++){
            table[i][j] = 0;
        }
    }
    table[1][','] = 2;
    table[1][';'] = 2;
    int state = 1;
    int n = strlen(ar);
    for (int i = 0; i < n; i++){
        state = table[state][ar[i]];
    }
    if (state == 2){
        return true;
    }
    return false;
}

bool number(char ar[]){
    int table[100][256];
    for (int i = 0; i < 100; i++){
        for (int j = 0; j < 256; j++){
```

```
            table[i][j] = 0;
        }
    }
    table[1]['.'] = 3;
    for (char c = '0'; c <= '9'; c++){
        table[1][c] = 2;
    }
    table[2]['.'] = 3;
    for (char c = '0'; c <= '9'; c++){
        table[2][c] = 2;
    }
    for (char c = '0'; c <= '9'; c++){
        table[3][c] = 4;
    }
    for (char c = '0'; c <= '9'; c++){
        table[4][c] = 4;
    }
    int state = 1;
    for (int i = 0; i < strlen(ar); i++){
        state = table[state][ar[i]];
    }
    if (state == 2 || state == 4){
        return true;
    }
    return false;
}

void categorize()
{
    int n = strlen(inp);
    for (int i = 0; i < n; i++)
    {
        char token[100];
        int ind = 0;
        while (inp[i] == ' '){
            i++;
        }
        while (inp[i] != ' ')
        {
            token[ind] = inp[i];
```

```
            ind++;
            i++;
        }
        token[ind] = '\0';
        //~ puts(token);
        if (keyword(token))
        {
            printf("[kw %s]", token);
        } else if (id(token)){
            printf("[id %s]", token);
        } else if (seperator(token)){
            printf("[sep %s]", token);
        } else if (_operator(token)){
            printf("[op %s]", token);
        } else if (number(token)){
            printf("[num %s]", token);
        } else if (parenthesis(token)){
            printf("[par %s]", token);
        } else {
            printf("[unkwn %s]", token);
        }
    }
}

int main()
{
    FILE *in = fopen("input_ass2.c", "r");
    int i = 0;
    char c;
    while ((c = fgetc(in)) != EOF)
    {
        inp[i] = c;
        i++;
    }
    inp[i] = '\0';
    seperate_2();
    seperate_1();
    categorize();
    return 0;
}
```

Output:

[kw char] [id c] [sep ;] [kw int] [id x1] [sep ,] [id x_2] [sep ;] [kw float] [id y1] [sep ,] [id y2] [sep ;] [id x1] [op =] [num 5] [sep ;] [id x_2] [op =] [num 10] [sep ;] [id y1] [op =] [num 2.5] [op +] [id x1] [op *] [num 45] [sep ;] [id y2] [op =] [unkn 100.o5] [op -] [id x_2] [op /] [num 3] [sep ;] [kw if] [par (] [id y1] [op <=] [id y2] [par)] [id c] [op =] [sep '] [id y] [sep '] [sep ;] [kw else] [id c] [op =] [sep '] [id n] [sep '] [sep ;]