# Practical-7

# Implement Predictive Parser for the above given grammar.

#include <conio.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

void main()

{

clrscr();

int i=0,j=0,k=0,m=0,n=0,o=0,o1=0,var=0,l=0,f=0,c=0,f1=0;

char str[30],str1[40]="E",temp[20],temp1[20],temp2[20],tt[20],t3[20];

strcpy(temp1,'\0');

strcpy(temp2,'\0');

char t[10];

char array[6][5][10] = {"NT", "<id>","+","\*",";","E","Te","Error","Error","Error","e", "Error","+Te","Error","\0","T", "Vt","Error","Error","Error","t", "Error","\0","\*Vt","\0","V", "<id>","Error","Error","Error"};

printf("\n\tLL(1) PARSER TABLE \n");

for(i=0;i<6;i++)

{

for(j=0;j<5;j++)

{

printf(“%d”,array[i][j]);

}

printf(“\n”);

}

printf(“\n”);

printf("\n\tENTER THE STRING :");

gets(str);

if(str[strlen(str)-1] != ';')

{

printf("END OF STRING MARKER SHOULD BE ';'");

getch();

exit(1);

}

printf("\n\tCHECKING VALIDATION OF THE STRING ");

printf("\n\t" << str1);

i=0;

while(i<strlen(str))

{

again:

if(str[i] == ' ' && i<strlen(str))

{

print("\n\tSPACES IS NOT ALLOWED IN SOURSE STRING ");

getch();

exit(1);

}

temp[k]=str[i];

temp[k+1]='\0';

f1=0;

again1:

if(i>=strlen(str))

{

getch();

exit(1);

}

for(int l=1;l<=4;l++)

{

if(strcmp(temp,array[0][l])==0)

{

f1=1;

m=0,o=0,var=0,o1=0;

strcpy(temp1,'\0');

strcpy(temp2,'\0');

int len=strlen(str1);

while(m<strlen(str1) && m<strlen(str))

{

if(str1[m]==str[m])

{

var=m+1;

temp2[o1]=str1[m];

m++;

o1++;

}

else

{

if((m+1)<strlen(str1))

{

m++;

temp1[o]=str1[m];

o++;

}

else

m++;

}

}

temp2[o1] = '\0';

temp1[o] = '\0';

t[0] = str1[var];

t[1] = '\0';

for(n=1;n<=5;n++)

{

if(strcmp(array[n][0],t)==0)

break;

}

strcpy(str1,temp2);

strcat(str1,array[n][l]);

strcat(str1,temp1);

printf("\n\t" <<str1);

getch();

if(strcmp(array[n][l],'\0')==0)

{

if(i==(strlen(str)-1))

{

int len=strlen(str1);

str1[len-1]='\0';

printf("\n\t"<<str1);

printf("\n\n\tENTERED STRING IS VALID");

getch();

exit(1);

}

strcpy(temp1,'\0');

strcpy(temp2,'\0');

strcpy(t,'\0');

goto again1;

}

if(strcmp(array[n][l],"Error")==0)

{

printf("\n\tERROR IN YOUR SOURCE STRING");

getch();

exit(1);

}

strcpy(tt,'\0');

strcpy(tt,array[n][l]);

strcpy(t3,'\0');

f=0;

for(c=0;c<strlen(tt);c++)

{

t3[c]=tt[c];

t3[c+1]='\0';

if(strcmp(t3,temp)==0)

{

f=0;

break;

}

else

f=1;

}

if(f==0)

{

strcpy(temp,'\0');

strcpy(temp1,'\0');

strcpy(temp2,'\0');

strcpy(t,'\0');

i++;

k=0;

goto again;

}

else

{

strcpy(temp1,'\0');

strcpy(temp2,'\0');

strcpy(t,'\0');

goto again1;

}

}

}

i++;

k++;

}

if(f1==0)

printf("\nENTERED STRING IS INVALID");

else

printf("\n\n\tENTERED STRING IS VALID");

getch();

}

OUTPUT:

LL(1) PARSER TABLE

NT <id> + \* ;

E Te Error Error Error

e Error +Te Error

T Vt Error Error Error

t Error \*Vt

V <id> Error Error Error

ENTER THE STRING :<id>+<id>;

CHECKING VALIDATION OF THE STRING

E

Te

Vte

<id>te

<id>e

<id>+Te

ENTERED STRING IS INVALID