***INDEX***

|  |  |  |  |
| --- | --- | --- | --- |
| ***S.No.*** | ***Program Name*** | ***Page No.*** | ***Date*** |
| **1.** | Program to find number of characters, spaces, lines, tabs in a given file |  |  |
| **2.** | Program to implement stand alone scanner in C |  |  |
| **3.** | Program to implement stand alone scanner using LEX |  |  |
| **4.** | Program to identify octal or hexadecimal number using LEX |  |  |
| **5.** | Program to capitalize the given comments using LEX |  |  |
| **6.** | Program to find complete real precision using LEX |  |  |
| **7.** | Program to implement recursive descent parser |  |  |
| **8.** | Program to implement brute force method |  |  |
| **9.** | Program to implement parser using YACC |  |  |
| **10.** | Program to implement calculator using YACC |  |  |
| **11.** | Progarm to find FIRST for the given grammar |  |  |
| **12.** | Program to find FOLLOW for the given grammar |  |  |
| **13.** | Progarm to generate LR(0) items for the given grammar |  |  |
| **14.** | Program to generate SLR parsing table |  |  |
| **15.** | Program to generate predictive LL1 parsing table |  |  |
| **16.** | Program to generate code |  |  |
| **17.** | Program to generate 3-address code |  |  |
| **18.** | Program to optimize the code |  |  |

***1.Program to find number of characters, spaces, lines, tabs in a given file***

#include<stdio.h>

int main(int argc,char \*argv[])

{

FILE \*f1;

char ch;

int lines=0,chars=0,tabs=0,spaces=0;

if(argc!=2)

{

printf("please enter file name");

return -1;

}

f1=fopen(argv[1],"r");

if(f1==NULL)

{

printf("can not open file");

return -1;

}

ch=getc(f1);

while(ch!=EOF)

{

switch(ch)

{

case '\n':

lines++;

break;

case '\t':

tabs++;

break;

case ' ':

spaces++;

break;

default:

chars++;

}

ch=getc(f1);

}

printf("number of lines:%d\n",lines);

printf("number of tabs:%d\n",tabs);

printf("number of spaces:%d\n",spaces);

printf("number of characters:%d\n",chars);

return 0;

}

***Output:***

./a.out prog.c

number of lines:43

number of tabs:0

number of spaces:27

number of characters:507

***2. Program to implement stand alone scanner in C***

#include<stdio.h>

#include<ctype.h>

#include<string.h>

int main()

{

FILE \*input, \*output;

int l=1;

int t=0;

int j=0;

int i,flag;

char ch,str[20];

input = fopen("input.txt","r");

output = fopen("output.txt","w");

char keyword[30][30] = {"int","main","if","else","do","while"};

fprintf(output,"Line no. \t Token no. \t Token \t Lexeme\n\n");

while(!feof(input))

{

i=0;

flag=0;

ch=fgetc(input);

if( ch=='+' || ch== '-' || ch=='\*' || ch=='/' )

{

fprintf(output,"%7d\t\t %7d\t\t Operator\t %7c\n",l,t,ch);

t++;

}

else if( ch==';' || ch=='{' || ch=='}' || ch=='(' || ch==')' || ch=='?' || ch=='@' || ch=='!' ||

ch=='%')

{

fprintf(output,"%7d\t\t %7d\t\t Special symbol\t %7c\n",l,t,ch);

t++;

}

else if(isdigit(ch))

{

fprintf(output,"%7d\t\t %7d\t\t Digit\t\t %7c\n",l,t,ch);

t++;

}

else if(isalpha(ch))

{

str[i]=ch;

i++;

ch=fgetc(input);

while(isalnum(ch) && ch!=' ')

{

str[i]=ch;

i++;

ch=fgetc(input);

}

str[i]='\0';

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

{

if(strcmp(str,keyword[j])==0)

{

flag=1;

break;

}

}

if(flag==1)

{

fprintf(output,"%7d\t\t %7d\t\t Keyword\t %7s\n",l,t,str);

t++;

}

else

{

fprintf(output,"%7d\t\t %7d\t\t Identifier\t %7s\n",l,t,str);

t++;

}

}

else if(ch=='\n')

{

l++;

}

}

fclose(input);

fclose(output);

return 0;

}

***Output:***

Line no. Token no. Token Lexeme

1 0 Identifier include

1 1 Identifier stdio

1 2 Identifier h

2 3 Keyword int

2 4 Keyword main

2 5 Keyword int

2 6 Identifier argc

2 7 Identifier char

2 8 Operator \*

2 9 Identifier argv

2 10 Special symbol )

3 11 Special symbol {

4 12 Identifier FILE

4 13 Operator \*

4 14 Identifier f1

5 15 Identifier char

5 16 Identifier ch

6 17 Keyword int

6 18 Identifier lines

***3. Program to implement stand alone scanner using LEX***

%{

int COMMENT=0;

%}

id [a-z][a-z0-9]\*

%%

#.\* {printf("\n%s is a PREPROCESSOR DIRECTIVE",yytext);}

int|double|char {printf("\n\t%s is a KEYWORD",yytext);}

if|then|endif {printf("\n\t%s is a KEYWORD",yytext);}

else {printf("\n\t%s is a KEYWORD",yytext);}

"/\*" {COMMENT=1;}

"\*/" {COMMENT=0;}

{id}\( {if(!COMMENT)printf("\n\nFUNCTION\n\t%s",yytext);}

{id}(\[[0-9]\*\])? {if(!COMMENT) printf("\n\tidentifier\t%s",yytext);}

\{ {if(!COMMENT) printf("\n BLOCK BEGINS");ECHO; }

\} {if(!COMMENT)printf("\n BLOCK ends");ECHO; }

\".\*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}

[+\-]?[0-9]+ {if(!COMMENT)printf("\n\t%s is a NUMBER",yytext);}

\( {if(!COMMENT)printf("\n\t");ECHO;printf("\t delim

openparanthesis\n");}

\) {if(!COMMENT)printf("\n\t");ECHO;printf("\t delim closed

paranthesis");}

\; {if(!COMMENT)printf("\n\t");ECHO;printf("\t delim semicolon");}

\= {if(!COMMENT)printf("\n\t%s is an ASSIGNMENT

OPERATOR",yytext);}

\<|\> {printf("\n\t %s is relational operator",yytext);}

"+"|"-"|"\*"|"/" {printf("\n %s is an operator\n",yytext);}

"\n" ;

%%

main(int argc ,char \*\*argv)

{

if (argc > 1)

yyin = fopen(argv[1],"r");

else

yyin = stdin;

yylex ();

printf ("\n");

}

int yywrap()

{

return 0;

}

**Output:**

lex lexscanner.l

cc lex.yy.c

./a.out input.txt

#include<stdio.h> is a PREPROCESSOR DIRECTIVE

int is a KEYWORD

***4. Program to identify octal or hexadecimal number using LEX***

%{

/\*program to identify octal and hexadecimal numbers\*/

%}

Oct [o][0-9]+

Hex [o][x|X][0-9A-F]+

%%

{Hex} printf("this is a hexadecimal number");

{Oct} printf("this is an octal number");

%%

main()

{

yylex();

}

int yywrap()

{

return 1;

}

**Output:**

./a.out

o5

this is an octal number

ox23

this is a hexadecimal number

***5. Program to capitalize the given comments using LEX***

%{

#include<stdio.h>

#include<ctype.h>

int k;

void display(char \*);

%}

letter [a-z]

com [//]

%%

{com} {k=1;}

{letter} {if(k==1) display(yytext);}

%%

main()

{

yylex();

}

void display(char \*s)

{

int i;

for(i=0;s[i]!='\0';i++)

printf("%c", toupper(s[i]));

}

int yywrap()

{

return 1;

}

***Output:***

lex caplex.l

cc lex.yy.c

./a.out

//hello world

HELLO WORLD

***6.Program to find complete real precision using LEX***

%{

/\*Program to find complete real precision using LEX\*/

%}

integer ([0-9]+)

float ([0-9]+\.[0-9]+)|([+|-]?[0-9]+\.[0-9]\*[e|E][+|-][0-9]\*)

%%

{integer} printf(\n %s is an integer\n,yytext);

{float} printf(\n %s is a floating number\n”,yytext);

%%

main()

{

yylex();

}

int yywrap()

{

return 1;

}

***Output:***

lex real.l

gcc lex.yy.c

./a.out 1234

1234 is an integer

***7.Program to implement recursive descent parser***

#include<stdio.h>

#include<string.h>

void E(),E1(),T(),T1(),F();

int ip=0;

static char s[10];

int main()

{

char k;

int l;

ip=0;

printf("enter the input");

scanf("%s",s);

printf("the string is :%s",s);

E();

if(s[ip]=='$')

printf("\n string is accepted the length of string is %d",strlen(s)-1);

else

printf("\n string not accepted\n");

return 0;

}

void E()

{

T();

E1();

return;

}

void E1()

{

if(s[ip]=='+')

{

ip++;

T();

E1();

}

return;

}

void T()

{

F();

T1();

return;

}

void T1()

{

if(s[ip]=='\*')

{

ip++;

F();

T1();

}

return;

}

void F()

{

if(s[ip]=='(')

{

ip++;

E();

if(s[ip]==')')

ip++;

}

else

if(s[ip]=='i')

ip++;

else

printf("\n id expected");

return;

}

***Output:***

cc recurparser.c

./a.out

enter the input

(i+i)\*(i\*i)$

the string is :(i+i)\*(i\*i)$

string is accepted the length of string is 11

***8. Program to implement brute force method***

#include<stdio.h>

#include<string.h>

char l[10],r[10][10],ch[15];

int flag=1;

char test[10];

int n=4,x,c=0;

int brute(int,char ch[]);

int main()

{

char st;

int i,y;

printf("Enter the no. of non-terminals\n");

scanf("%d",&x);

printf("Enter the non-terminals\n");

for(y=0;y<x;y++)

{

scanf("%s",&l[y]);

}

printf("Enter the corresponding grammar\n");

for( y=0;y<x;y++)

{

scanf("%s",&r[y]);

}

printf("Enter string\n");

scanf("%s",test);

printf("\nEnter starting symbol\n");

scanf("%c",&st);

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

{

if(l[i]==st)

{

strcpy(ch,r[i]);

flag=brute(i,ch);

if(flag==0) break;

}

}

if(flag==-1)

printf("Invalid String\n");

else

printf("Valid String\n");

return 0;

}

int brute(int i,char ch[10])

{

char chc[10];

int len,j,k,lm,tmp,m,ret;

c++;

strcpy(chc,ch);

for(j=0;chc[j]!='\0';)

{

if(chc[j]==test[j])

{

j++;

i=j;

}

else if((chc[j]>=65)&&(chc[j]<=90))

for(k=0;k<x;k++)

{

strcpy(chc,ch);

if(l[k]==chc[j])

{

len=strlen(r[k]);

tmp=strlen(chc);

for(m=tmp;m>=i;m--)

chc[m+len-1]=chc[m];

for(m=0;m<len;m++)

chc[m+i]=r[k][m];

ret=brute(j,chc);

if(ret==0)

return 0;

else

return -1;

}

}

else

return -1;

}

return 0;

}

***Output:***

cc bruteforce.c

./a.out

Enter the no. of non-terminals

3

Enter the non-terminals

S

A

B

Enter the corresponding grammar

S->AB

A->a

B->b

Enter string

ab$

Enter starting symbol

Valid String

***9. Program to implement parser using YACC***

***parser.l***

%{

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext); return NUM;}

[\t]

\n return 0;

return yytext[0];

%%

int yywrap()

{

return 0;

}

***Parser.y***

%token NUM

%%

cmd :E {printf("%d\n",$1);}

;

E :E '+' T {$$=$1+$3;}

|T {$$=$1;}

;

T :T'\*'F {$$=$1\*$3;}

|F {$$=$1;}

;

F :'(' E ')' {$$=$2;}

|NUM {$$=$1;}

;

%%

int main()

{

yyparse();

}

yyerror(char \*s)

{

printf("%s",s);

}

***Output:***

lex parser.l

yacc -d parser.y

gcc lex.yy.c y.tab.c -ll -ly

./a.out

2+3

5

***10. Program to implement calculator using YACC***

***cal.l***

%{

#include<stdio.h>

#include "y.tab.h"

%}

%%

[0-9]+ {yylval.dval=atoi(yytext); return DIGIT;}

\n|. return yytext[0];

%%

Cal.y

%{

/\* \*/

%}

%union

{

int dval;

}

%token <dval> DIGIT

%type <dval> expr

%type <dval> expr1

%%

line : expr '\n' {printf("%d\n",$1);}

;

expr : expr'+'expr1 {$$=$1+$3;}

| expr'-'expr1 {$$=$1-$3;}

| expr'\*'expr1 {$$=$1\*$3;}

| expr'/'expr1 {$$=$1/$3;}

| expr1

;

expr1: '('expr')' {$$=$2;}

| DIGIT

;

%%

int main()

{

yyparse();

}

yyerror(char \*s)

{

printf("%s",s);

}

***Output:***

$ lex cal.l

$ yacc -d cal.y

$ gcc lex.yy.c y.tab.c -ll

$ ./a.out

1+2

3

***11. Progarm to find FIRST for the given grammar***

#include<stdio.h>

#include<ctype.h>

int main()

{

int i,n,j,k;

char str[10][10],f;

printf("Enter the number of productions\n");

scanf("%d",&n);

printf("Enter grammar\n");

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

scanf("%s",&str[i]);

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

{

f= str[i][0];

int temp=i;

if(isupper(str[i][3]))

{

repeat:

for(k=0;k<n;k++)

{

if(str[k][0]==str[i][3])

{

if(isupper(str[k][3]))

{

i=k;

goto repeat;

}

else

{

printf("First(%c)=%c\n",f,str[k][3]);

}

}

}

}

else

{

printf("First(%c)=%c\n",f,str[i][3]);

}

i=temp;

}

}

***Output:***

cc first.c

./a.out

Enter the number of productions

3

Enter grammar

S->AB

A->a

B->b

First(S)=a

First(A)=a

First(B)=b

***12. Program to find FOLLOW for the given grammar***

#include<stdio.h>

main()

{

int np,i,j,k;

char prods[10][10],follow[10][10],Imad[10][10];

printf("enter no. of productions\n");

scanf("%d",&np);

printf("enter grammar\n");

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

{

scanf("%s",&prods[i]);

}

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

{

if(i==0)

{

printf("Follow(%c) = $\n",prods[0][0]);

}

for(j=3;prods[i][j]!='\0';j++)

{

int temp2=j;

if(prods[i][j] >= 'A' && prods[i][j] <= 'Z')

{

if((strlen(prods[i])-1)==j)

{

printf("Follow(%c)=Follow(%c)\n",prods[i][j],prods[i][0]);

}

int temp=i;

char f=prods[i][j];

if(!isupper(prods[i][j+1])&&(prods[i][j+1]!='\0'))

printf("Follow(%c)=%c\n",f,prods[i][j+1]);

if(isupper(prods[i][j+1]))

{

repeat:

for(k=0;k<np;k++)

{

if(prods[k][0]==prods[i][j+1])

{

if(!isupper(prods[k][3]))

{

printf("Follow(%c)=%c\n",f,prods[k][3]);

}

else

{

i=k;

j=2;

goto repeat;

}

}

}

}

i=temp;

}

j=temp2;

}

}

}

***Output:***

./a.out

enter no. of productions

3

enter grammar

S->AB

A->a

B->b

Follow(S) = $

Follow(A)=b

Follow(B)=Follow(S)

***13. Progarm to generate LR(0) items for the given grammar***

#include<stdio.h>

#include<string.h>

char a[8][5],b[7][5];

int c[12][5];

int w=0,e=0,x=0,y=0;

int st2[12][2],st3[12];

char sta[12],ch;

void v1(char,int);

void v2(char,int,int,int);

int main()

{

int i,j,k,l=0,m=0,p=1,f=0,g,v=0,jj[12];

printf("\n\n\t\*\*\*\*\*\*\*Enter the Grammar Rules (max=3)\*\*\*\*\*\*\*\n\t");

for(i=0;i<3;i++)

{

gets(a[i]);

printf("\t");

}

for(i=0;i<3;i++)

{

for(j=0;j<strlen(a[i]);j++)

{

for(k=0;k<strlen(a[i]);k++)

{

if(p==k)

{

b[l][m]='.';

m+=1;

b[l][m]=a[i][k];

m+=1;

}

else

{

b[l][m]=a[i][k];

m++;

}

}

p++;

l++;

m=0;

}

p=1;

}

i=0; p=0;

while(l!=i)

{

for(j=0;j<strlen(b[i]);j++)

{

if(b[i][j]=='.')

{

p++;

}

}

if(p==0)

{

b[i][strlen(b[i])]='.';

}

i++;

p=0;

}

i=0;

printf("\n\t\*\*\*\*\*\*\*Your States will be\*\*\*\*\*\*\*\n\t");

while(l!=i)

{

printf("%d--> ",i);

puts(b[i]);

i++;

printf("\t");

}

printf("\n");

v1('A',l);

p=c[0][0];

m=0;

while(m!=6)

{

for(i=0;i<st3[m];i++)

{

for(j=0;j<strlen(b[p]);j++)

{

if(b[p][j]=='.' && ((b[p][j+1]>=65 && b[p][j+1]<=90)||

(b[p][j+1]>=97&&b[p][j+1]<=122)))

{

st2[x][0]=m;

sta[x]=b[p][j+1];

v2(b[p][j+1],j,l,f);

x++;

}

else

{

if(b[p][j]=='.')

{

st2[x][0]=m;

sta[x]='S';

st2[x][1]=m;

x++;

}

}

}

p=c[m][i+1];

}

m++;

p=c[m][0];

}

g=0;

p=0;

m=0;

x=0;

getchar();

return 0;

}

void v1(char ai,int kk)

{

int i,j;

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

{

if(b[i][2]==ai&&b[i][1]=='.')

{

c[w][e]=i;

e++;

if(b[i][2]>=65 && b[i][2]<=90)

{

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

{

if(b[j][0]==ai && b[j][1]=='.')

{

c[w][e]=j;

e++;

}

}

}

}

}

st3[w]=e;

w++;

e=0;

}

void v2(char ai,int ii,int kk,int tt)

{

int i,j,k;

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

{

if(b[i][ii]=='.'&& b[i][ii+1]==ai)

{

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

{

if(b[j][ii+1]=='.' && b[j][ii]==ai)

{

c[w][e]=j;

e++;

st2[tt][1]=j;

if(b[j][ii+2]>=65 && b[j][ii+1]<=90)

{

for(k=0;k<kk;k++)

{

if(b[k][0]==b[j][ii+2] && b[k][1]=='.')

{

c[w][e]=k;

e++;

}

}

}

}

}

if((b[i][ii+1]>=65 && b[i][ii+1]<=90) && tt==1)

{

for(k=0;k<kk;k++)

{

if(b[k][0]==ai && b[k][1]=='.')

{

c[w][e]=k;

e++;

}

}

}

}

}

st3[w]=e;

w++;

e=0;

}

***Output:***

./a.out

\*\*\*\*\*\*\*Enter the Grammar Rules (max=3)\*\*\*\*\*\*\*

SAB

Aa

Bb

\*\*\*\*\*\*\*Your States will be\*\*\*\*\*\*\*

0--> S.AB

1--> SA.B

2--> SAB.

3--> A.a

4--> Aa.

5--> B.b

6--> Bb.

***14. Program to generate SLR parsing table***

#include<stdio.h>

#include<string.h>

char a[8][5],b[7][5];

int c[12][5];

int w=0,e=0,x=0,y=0;

int st2[12][2],st3[12];

char sta[12],ch;

void v1(char,int);

void v2(char,int,int,int);

main()

{

int i,j,k,l=0,m=0,p=1,f=0,g,v=0,jj[12];

printf("\n\n\t\*\*\*\*\*\*\*Enter the Grammar Rules (max=3)\*\*\*\*\*\*\*\n\t");

for(i=0;i<3;i++)

{

gets(a[i]);

printf("\t");

}

for(i=0;i<3;i++)

{

for(j=0;j<strlen(a[i]);j++)

{

for(k=0;k<strlen(a[i]);k++)

{

if(p==k)

{

b[l][m]='.';

m+=1;

b[l][m]=a[i][k];

m+=1;

}

else

{

b[l][m]=a[i][k];

m++;

}

}

p++;

l++;

m=0;

}

p=1;

}

i=0; p=0;

while(l!=i)

{

for(j=0;j<strlen(b[i]);j++)

{

if(b[i][j]=='.')

{

p++;

}

}

if(p==0)

{

b[i][strlen(b[i])]='.';

}

i++;

p=0;

}

i=0;

printf("\n\t\*\*\*\*\*\*\*Your States will be\*\*\*\*\*\*\*\n\t");

while(l!=i)

{

printf("%d--> ",i);

puts(b[i]);

i++;

printf("\t");

}

printf("\n");

v1('A',l);

p=c[0][0];

m=0;

while(m!=6)

{

for(i=0;i<st3[m];i++)

{

for(j=0;j<strlen(b[p]);j++)

{

if(b[p][j]=='.' && ((b[p][j+1]>=65 && b[p][j+1]<=90)

||(b[p][j+1]>=97&&b[p][j+1]<=122)))

{

st2[x][0]=m;

sta[x]=b[p][j+1];

v2(b[p][j+1],j,l,f);

x++;

}

else

{

if(b[p][j]=='.')

{

st2[x][0]=m;

sta[x]='S';

st2[x][1]=m;

x++;

}

}

}

p=c[m][i+1];

}

m++;

p=c[m][0];

}

g=0; p=0; m=0;x=0;

while(p!=11)

{

for(i=0;i<st3[p];i++)

{

for(k=0;k<p;k++)

{

for(j=0;j<3;j++)

{

if(c[k][j]==c[p][j])

{

m++;

}

}

if(m==3)

{

m=0;

goto ac;

}

m=0;

}

if(m!=3)

{

if(v==0)

{

printf("\tI%d=",g);

v++;

jj[g]=p;

}

printf("%d",c[p][i]);

}

}

printf("\n");

g++;

ac:

p++;

v=0;

}

printf("\t\*\*\*\*\*\*\*Your DFA will be \*\*\*\*\*\*\*");

for(i=0;i<9;i++)

{

printf("\n\t%d",st2[i][0]);

printf("-->%c",sta[i]);

}

getchar();

}

void v1(char ai,int kk)

{

int i,j;

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

{

if(b[i][2]==ai&&b[i][1]=='.')

{

c[w][e]=i;

e++;

if(b[i][2]>=65 && b[i][2]<=90)

{

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

{

if(b[j][0]==ai && b[j][1]=='.')

{

c[w][e]=j;

e++;

}

}

}

}

}

st3[w]=e;

w++;

e=0;

}

void v2(char ai,int ii,int kk,int tt)

{

int i,j,k;

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

{

if(b[i][ii]=='.'&& b[i][ii+1]==ai)

{

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

{

if(b[j][ii+1]=='.' && b[j][ii]==ai)

{

c[w][e]=j;

e++;

st2[tt][1]=j;

if(b[j][ii+2]>=65 && b[j][ii+1]<=90)

{

for(k=0;k<kk;k++)

{

if(b[k][0]==b[j][ii+2] && b[k][1]=='.')

{

c[w][e]=k;

e++;

}

}

}

}

}

if((b[i][ii+1]>=65 && b[i][ii+1]<=90) && tt==1)

{

for(k=0;k<kk;k++)

{

if(b[k][0]==ai && b[k][1]=='.')

{

c[w][e]=k;

e++;

}

}

}

}

}

st3[w]=e;

w++;

e=0;

}

***Output:***

./a.out

\*\*\*\*\*\*\*Enter the Grammar Rules (max=3)\*\*\*\*\*\*\*

SAB

Aa

Bb

\*\*\*\*\*\*\*Your States will be\*\*\*\*\*\*\*

0--> S.AB

1--> SA.B

2--> SAB.

3--> A.a

4--> Aa.

5--> B.b

6--> Bb.

I0=03

I1=15

I2=4

I3=2

I4=6

\*\*\*\*\*\*\*Your DFA will be \*\*\*\*\*\*\*

0-->A

0-->a

1-->B

1-->b

2-->S

3-->S

4-->S

0-->

0-->

***15.Program to generate predictive LL1 parsing table***

#include<stdio.h>

#include<string.h>

char prod[10][20],start[2];

char nonterm[10],term[10];

char input[10],stack[50];

int table[10][10];

int te,nte;

int n;

void main()

{

init();

parse();

}

init()

{

int i,j;

printf("\nNOTE:\n");

printf("The terminals should be entered in single lower case letters,special symbol and\n");

printf("non-terminals should be entered in single upper case letters.\n");

printf("extends to symbol is '->' and epsilon symbol is '@' \n");

printf("\nEnter the no. of terminals:");

scanf("%d",&te);

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

{

fflush(stdin);

printf("Enter the terminal %d:",i+1);

scanf("%c",&term[i]);

}

term[i]='$';

printf("\nEnter the no. of non terminals:");

scanf("%d",&nte);

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

{

fflush(stdin);

printf("Enter the non-terminal %d:",i+1);

scanf("%c",&nonterm[i]);

}

printf("\nEnter the no. of productions:");

scanf("%d",&n);

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

{

printf("Enter the production %d:",i+1);

scanf("%s",prod[i]);

}

fflush(stdin);

printf("\nEnter the start symbol:");

scanf("%c",&start[0]);

printf("\nEnter the input string:");

scanf("%s",input);

input[strlen(input)]='$';

printf("\n\nThe productions are:");

printf("\nProductionNo. Production");

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

printf("\n %d %s",i+1,prod[i]);

printf("\n\nEnter the parsing table:");

printf("\n Enter the production number in the required entry as mentioned above.");

printf("\n Enter the undefined entry or error of table as '0'\n\n");

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

{

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

{

fflush(stdin);

printf("Entry of table[%c,%c]:",nonterm[i],term[j]);

scanf("%d",&table[i][j]);

}

}

}

parse()

{

int i,j,prodno;

int top=-1,current=0;

stack[++top]='$';

stack[++top]=start[0];

do

{

if((stack[top]==input[current])&&(input[current]=='$'))

{

printf("\nThe given input string is parsed");

exit(0);

}

else if(stack[top]==input[current])

{

top--;

current++;

}

else if(stack[top]>='A'&&stack[top]<='Z')

{

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

if(nonterm[i]==stack[top]) break;

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

if(term[j]==input[current]) break;

prodno=table[i][j];

if(prodno==0)

{

printf("\nThe given input string is not parsed");

exit(0);

}

else

{

for(i=strlen(prod[prodno-1])-1;i>=3;i--)

{

if(prod[prodno-1][i]!='@')

stack[top++]=prod[prodno-1][i];

}

top--;

}

}

else

{

printf("\nThe given input string is not parsed");

exit(0);

}

}while(1);

}

***Output:***

***16. Program to generate code***

#include<stdio.h>

char stk[100],stktop=-1,cnt=0;

void push(char pchar)

{

stk[++stktop]=pchar;

}

char pop()

{

return stk[stktop--];

}

char checkoperation(char char1)

{

char oper;

if(char1=='+')

oper='A';

else if(char1=='-')

oper='S';

else if(char1=='\*')

oper='M';

else if(char1=='/')

oper='D';

else if(char1=='@')

oper='N';

return oper;

}

int checknstore(char check)

{

int ret;

if(check!='+' && check!='-' && check!='\*' && check!='/' && check!='@')

{

push(++cnt);

if(stktop>0)

printf("ST $%d\n",cnt);

ret=1;

}

else

ret=0;

return ret;

}

int main()

{

char msg[100],op1,op2,operation;

int i,val;

while(scanf("%s",msg)!=EOF)

{

cnt=0;

stktop=-1;

for(i=0;msg[i]!='\0';i++)

{

if((msg[i] >='A' && msg[i]<='Z') ||(msg[i]>='a' && msg[i]<='z'))

push(msg[i]);

else

{

op1=pop();

op2=pop();

printf("L %c\n",op2);

operation=checkoperation(msg[i]);

printf("%c %c\n",operation,op1);

val=checknstore(msg[i+1]);

while(val==0)

{

op1=pop();

cnt--;

operation=checkoperation(msg[++i]);

if(operation=='S'&&stktop>=-1)

{

printf("N\n");

operation='A';

}

printf("%c %c\n",operation,op1);

val=checknstore(msg[i+1]);

}

}

}

}

}

***Output:***

cc codegen.c

[be09-100@cs1 ~]$ ./a.out

ab+

L a

A b

ab\*

L a

M b

ab/

L a

D b

ab-

L a

S b

***17.Program to generate 3-address code***

#include<stdio.h>

#include<string.h>

char ip[20],op[20],arg1[20],arg2[20],res[20];

char r[5]={'1','2','3','4','5'};

int l,p,j=0;

void rep(int,int,int);

void repl(int,int,int);

void check(int,char);

int oppr(char);

void main()

{

int i;

printf("Enter the input Expression\n");

scanf("%s",ip);

l=strlen(ip);

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

{

p=oppr(ip[i]);

if(p==5)

{

check(i,ip[i]);

}

}

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

{

p=oppr(ip[i]);

if(p==6)

{

rep(i-1,i,i+1);

i=0;

}

}

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

{

p=oppr(ip[i]);

if(p==5)

{

rep(i-1,i,i+1);

i=0;

}

}

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

{

p=oppr(ip[i]);

if(p==4)

{

rep(i-1,i,i+1);

i=0;

}

}

printf("The triple notation is \n");

printf("op\targ1\targ2\n");

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

printf("%c\t%c\t%c\n",op[i],arg1[i],arg2[i]);

}

int oppr(char c)

{

if(c=='\*'||c=='/')

return(6);

else if(c=='+'||c=='-')

return(5);

else if(c=='=')

return(4);

}

void rep(int a,int b,int c)

{

int i,k;

op[j]=ip[b];

arg1[j]=ip[a];

arg2[j]=ip[c];

ip[a]=r[j];

k=b;

k++;

for(i=b;i<l;i++)

{

ip[i]=ip[k+1];

k++;

}

l=l-2;

j++;

}

void check(int i,char c)

{

int a,b;

if(c=='-')

{

if(!isalpha(ip[i-1]))

{

op[j]='m';

arg1[j]=ip[i+1];

ip[j]=r[j];

b=i;

b++;

for(a=i+1;a<l;a++)

{

ip[a]=ip[b+1];

b++;

}

l--;

j++;

}

}

}

***Output:***

cc 3addcode.c

./a.out

Enter the input Expression

a\*b+c/d

The triple notation is

op arg1 arg2

\* a b

/ c d

+ 1 2

***18. Code Optimization***

#include<stdio.h>

int tc[10],fb=0,i=0,j=0,k=0,p=0,fstar=0,c=-1,c1=0,c2=0,t1,t2,t3,t4,f0=0;

char m[30],temp[30],opt[10][4];

int operator1(char,char);

void main()

{

int a,d;

for(i=0;i<10;i++)

tc[i]=-1;

printf("\n code stmt evaluation following precedence:");

printf("\n 1.() with in the () stmet should be of the form: x op z");

printf("\n 2.\*,/ equal precedence");

printf("\n 3.+,- equal precedence");

printf("\n enter ur code stmt-");

scanf("%s",m);

i=0;

while(m[i]!='\0')

{

if(m[i++]=='(')

{

fb++;

break;

}

}

i=0;

printf("\n the intermediate code may be generated as-");

if(fb==1)

{

while(m[i]!='\0')

if(m[i]=='(')

{

temp[j++]='T';

i++;

t3=i;

while(m[i]!=')')

opt[c1][c2++]=m[i++];

for(t4=c1-1;t4>=0;t4--)

if(strcmp(opt[c1],opt[t4])==0)

{

tc[p++]=t4;

f0=1;

}

if(f0==0)

{

tc[p++]=k++;

printf("\n T%d=",k-1);

while(m[t3]!=')')

printf("%c",m[t3++]);

}

i++;

c1++;

c2=f0=0;

}

else if (m[i]!='(')

temp[j++]=m[i++];

if(fb==1)

{

temp[j]='\0';

for(i=0;temp[i]!='\0';i++)

m[i]=temp[i];

m[i]='\0';

}

}

/\*end of evaluating sub emxp\*/

a=operator1('\*','/');

d=operator1('+','-');

if(a==0 && d==0 && m[1]=='=')

printf("\n%s%d",m,k-1);

}

/\*function to evaluate operatores\*/

int operator1(char haj,char haj1)

{

m1:for(i=0;m[i]!='\0';i++)

if(m[i]==haj|| m[i]==haj1)

{

fstar++;

break;

}

if(fstar==1)

{

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

if(m[j]=='T')

c++;

printf("\n T%d=",k);

if(m[i-1]=='T'&&m[i+1]=='T')

{

printf("\n%c%d%c%c%d",m[i-1],tc[c],m[i],m[i+1],tc[c+1]);

tc[c]=k++;

for(t2=c+1;t2<9;t2++)

tc[t2]=tc[t2+1];

}

else if(m[i-1]!='T' &&m[i+1]!='T')

{

printf("%c%c%c",m[i-1],m[i],m[i+1]);

if(c==-1)

{

for(t1=9;t1>0;t1--)

tc[t1]=tc[t1-1];

tc[0]=k++;

}

else if(c>=0)

{

for(t1=9;t1>c+1;t1--)

tc[t1]=tc[t1-1];

tc[t1]=k++;

}

}

else if(m[i-1]=='T' && m[i+1]!='T')

{

printf("%c%d%c%c",m[i-1],tc[c],m[i],m[i+1]);

tc[c]=k++;

}

else if(m[i-1]!='T'&& m[i+1]=='T')

{

printf("%c%c%c%d",m[i-1],m[i],m[i+1],tc[c+1]);

tc[c+1]=k++;

}

for(t1=0;t1<i-1;t1++)

temp[t1]=m[t1];

temp[t1++]='T';

for(t2=i+2;m[t2]!='\0';t2++)

temp[t1++]=m[t2];

temp[t1++]='\0';

fstar=0;

for(i=0;temp[i]!='\0';i++)

m[i]=temp[i];

m[i]='\0';

c=-1;

goto m1;

}

else

return 0;

}

***Output:***

code stmt evaluation following precedence:

1.() with in the () stmet should be of the form: x op z

2.\*,/ equal precedence

3.+,- equal precedence

enter ur code stmt-a:=(b+c)\*(b+c)-2

the intermediate code may be generated as-

T0=b+c

T1=

T0\*T0

T2=T1-2