**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

int isKeyword(char buffer[]){

char keywords[32][10] = {"auto","break","case","char","const","continue","default",

"do","double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef","union",

"unsigned","void","volatile","while"};

int i, flag = 0;

for(i = 0; i < 32; ++i){

if(strcmp(keywords[i], buffer) == 0){

flag = 1;

break;

}

}

return flag;

}

int main(){

char ch, buffer[15], operators[] = "+-\*/%=";

FILE \*fp;

int i,j=0;

fp = fopen("program.txt","r");

if(fp == NULL){

printf("error while opening the file\n");

exit(0);

}

while((ch = fgetc(fp)) != EOF){

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

if(ch == operators[i])

printf("%c is operator\n", ch);

}

if(isalnum(ch)){

buffer[j++] = ch;

}

else if((ch == ' ' || ch == '\n') && (j != 0)){

buffer[j] = '\0';

j = 0;

if(isKeyword(buffer) == 1)

printf("%s is keyword\n", buffer);

else

printf("%s is indentifier\n", buffer);

}

}

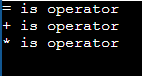
fclose(fp);

return 0;

}

**OUTPUT:**

**Input give is d=a+b\*c**

****

**PROGRAM:**

%{

int COMMENT=0;

%}

identifier [a-zA-Z][a-zA-Z0-9]\*

%%

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

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

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

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

\} {if(!COMMENT)printf("BLOCK ENDS ");}

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

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

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

= {if(!COMMENT)printf("\n %s\t is an ASSIGNMENT OPERATOR",yytext);}

\<= | \>= | \< | == | \> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}

\+ | \- | \\* | \% {if(!COMMENT) printf("\n\t%s is an ARITHMETIC OPERATOR",yytext);}

%%

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

{

FILE \*file;

file=fopen("jo2.c","r");

if(!file)

{

printf("could not open the file");

exit(0);

}

yyin=file;

yylex();

printf("\n");

return(0);}

int yywrap()

{

return(1);

}

**JO2.c program**

#include<stdio.h>

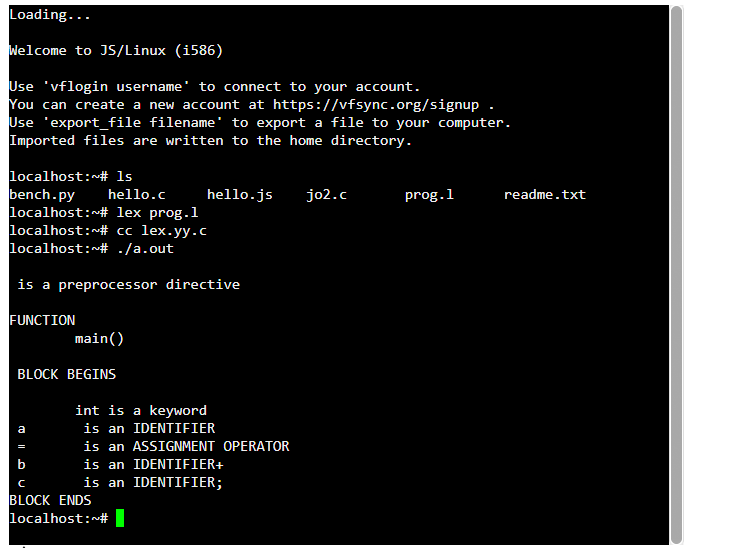
main()

{

int a=b+c;

}

**OUTPUT:**



**PROGRAM:**

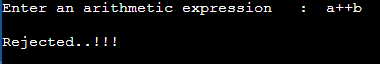
#include<stdio.h>  
#include<string.h>  
#include<ctype.h>  
   
char input[10];  
int i,error;  
void E();  
void T();  
void Eprime();  
void Tprime();  
void F();

main()  
{  
i=0;  
error=0;  
printf("Enter an arithmetic expression   :  "); // Eg: a+a\*a  
gets(input);  
E();  
if(strlen(input)==i&&error==0)  
printf("\nAccepted..!!!\n");  
else printf("\nRejected..!!!\n");  
}

void E()  
{  
     T();  
     Eprime();  
}  
void Eprime()  
{  
     if(input[i]=='+')  
     {  
     i++;  
     T();  
     Eprime();  
     }  
     }  
void T()  
{  
     F();  
     Tprime();  
}  
void Tprime()  
{  
     if(input[i]=='\*')  
     {  
                      i++;  
                      F();  
                      Tprime();  
                      }  
                      }  
     void F()  
     {  
          if(isalnum(input[i]))i++;  
          else if(input[i]=='(')  
          {  
          i++;  
          E();  
          if(input[i]==')')  
          i++;  
  
          else error=1;  
            }  
          
          else error=1;  
          }

**OUTPUT:**

****

****

**PROGRAM:**

**For .l file**

%{

#include <stdlib.h>

void yyerror(char \*);

#include "y.tab.h"

%}

%%

/\* variables \*/

[a-z] {

yylval = \*yytext - 'a';

return VARIABLE;

}

/\* integers \*/

[0-9]+ {

yylval = atoi(yytext);

return INTEGER;

}

/\* operators \*/

[-+()=/\*\n] { return \*yytext; }

/\* skip whitespace \*/

[ \t] ;

/\* anything else is an error \*/

. yyerror("invalid character");

%%

int yywrap(void) {

return 1;

}

**For .y file**

%token INTEGER VARIABLE

%left '+' '-'

%left '\*' '/'

%{

void yyerror(char \*);

int yylex(void);

int sym[26];

%}

%%

program:

program statement '\n'

|

;

statement:

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

| VARIABLE '=' expr { sym[$1] = $3; }

;

expr:

INTEGER

| VARIABLE { $$ = sym[$1]; }

| expr '+' expr { $$ = $1 + $3; }

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

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

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

| '(' expr ')' { $$ = $2; }

;

%%

void yyerror(char \*s) {

printf("Error encoutered");

return 0;

}

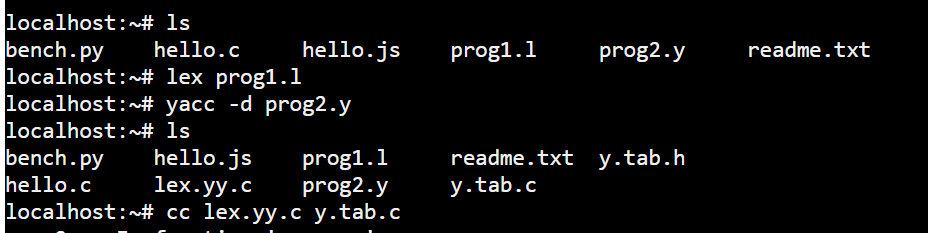
int main(void) {

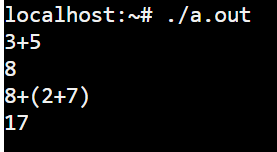
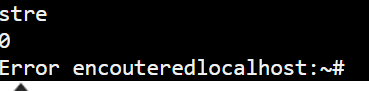
yyparse();

return 0;

}

**OUTPUT:**

****

** **

**Lexical Analyzer Source Code:**

|  |
| --- |
| %{     /\* Definition section \*/    #include<stdio.h>    #include "y.tab.h"  **extern** **int** yylval;  %}    /\* Rule Section \*/  %%  [0-9]+ {            yylval=**atoi**(yytext);  **return** NUMBER;           }  [\t] ;    [\n] **return** 0;    . **return** yytext[0];    %%    **int** yywrap()  {  **return** 1;  } |

**Parser Source Code :**

|  |
| --- |
| %{     /\* Definition section \*/    #include<stdio.h>  **int** flag=0;  %}    %token NUMBER    %left '+' '-'    %left '\*' '/' '%'    %left '(' ')'    /\* Rule Section \*/  %%    ArithmeticExpression: E{    **printf**("\nResult=%d\n", $$);    **return** 0;            };   E:E'+'E {$$=$1+$3;}     |E'-'E {$$=$1-$3;}     |E'\*'E {$$=$1\*$3;}     |E'/'E {$$=$1/$3;}     |E'%'E {$$=$1%$3;}     |'('E')' {$$=$2;}     | NUMBER {$$=$1;}     ;    %%    //driver code  **void** main()  {  **printf**("\nEnter Any Arithmetic Expression which     can have operations Addition,Subtraction, Multiplication, Division,Modulus and Round brackets:\n");     yyparse();  **if**(flag==0)  **printf**("\nEntered arithmetic expression is Valid\n\n");  }    **void** yyerror()  {  **printf**("\nEntered arithmetic expression is Invalid\n\n");     flag=1;  } |

**OUTPUT:**

**Input: 4+5**

**Output: Result=9**

**Entered arithmetic expression is Valid**

**Input: 10-5**

**Output: Result=5**

**Entered arithmetic expression is Valid**

**Input: 10+5-**

**Output:**

**Entered arithmetic expression is Invalid**

**PROGRAM:**

#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

void small();

void dove(int i);

int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;

char sw[5]={'=','-','+','/','\*'},j[20],a[5],b[5],ch[2];

void main()

{

printf("Enter the expression:");

scanf("%s",j);

printf("\tThe Intermediate code is:\n");

small();

}

void dove(int i)

{

a[0]=b[0]='\0';

if(!isdigit(j[i+2])&&!isdigit(j[i-2]))

{

a[0]=j[i-1];

b[0]=j[i+1];

}

if(isdigit(j[i+2]))

{

a[0]=j[i-1];

b[0]='t';

b[1]=j[i+2];

}

if(isdigit(j[i-2]))

{

b[0]=j[i+1];

a[0]='t';

a[1]=j[i-2];

b[1]='\0';

}

if(isdigit(j[i+2]) &&isdigit(j[i-2]))

{

a[0]='t';

b[0]='t';

a[1]=j[i-2];

b[1]=j[i+2];

sprintf(ch,"%d",c);

j[i+2]=j[i-2]=ch[0];

}

if(j[i]=='\*')

printf("\tt%d=%s\*%s\n",c,a,b);

if(j[i]=='/')

printf("\tt%d=%s/%s\n",c,a,b);

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

printf("\tt%d=%s+%s\n",c,a,b);

if(j[i]=='-')

printf("\tt%d=%s-%s\n",c,a,b);

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

printf("\t%c=t%d",j[i-1],--c);

sprintf(ch,"%d",c);

j[i]=ch[0];

c++;

small();

}

void small()

{

pi=0;l=0;

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

{

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

if(j[i]==sw[m])

if(pi<=p[m])

{

pi=p[m];

l=1;

k=i;

}

}

if(l==1)

dove(k);

else

exit(0);

}

**OUTPUT:**

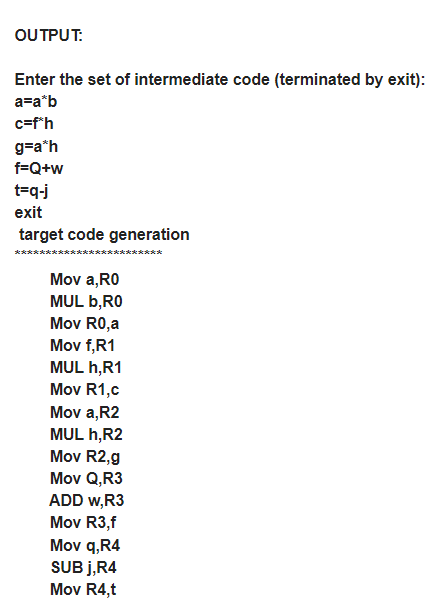
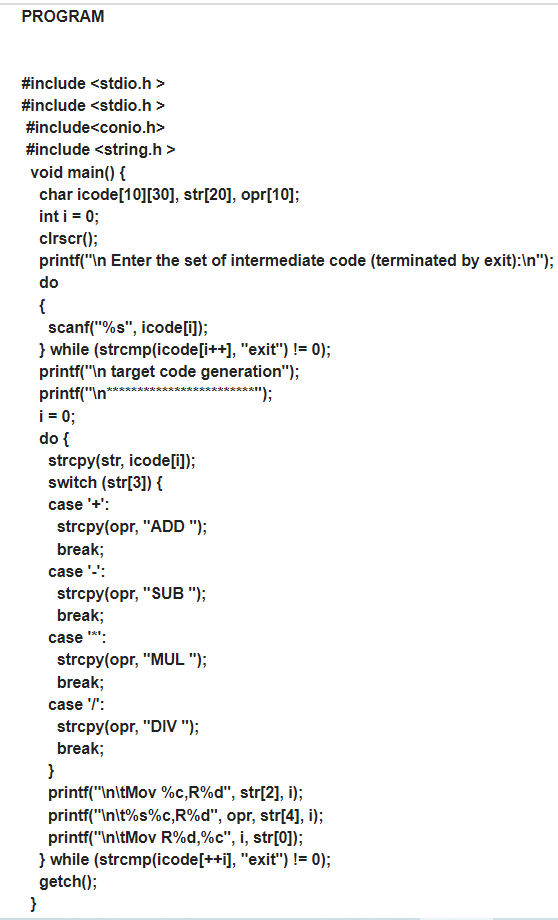
Enter the expression:a=b+c-d

The Intermediate code is:

t1=b+c

t2=t1-d

a=t2

****

**PROGRAM:**

%{

%}

%%

"<"[^>]\*> {**printf**("%s\n", yytext); }

%%

**int** yywrap(){}

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

{

    yyin = **fopen**("tags.txt","r");

    yylex();

**return** 0;

}

**Tags.txt file**

<html>

<head>

</head>

<body>

<h1> My First Heading </h1>

<p> My paragraph </p>

</body>

</html>

**OUTPUT:**

<html>

<head>

</head>

<body>

<h1>

</h1>

<p>

</p>

</body>

</html>