EX NO 1

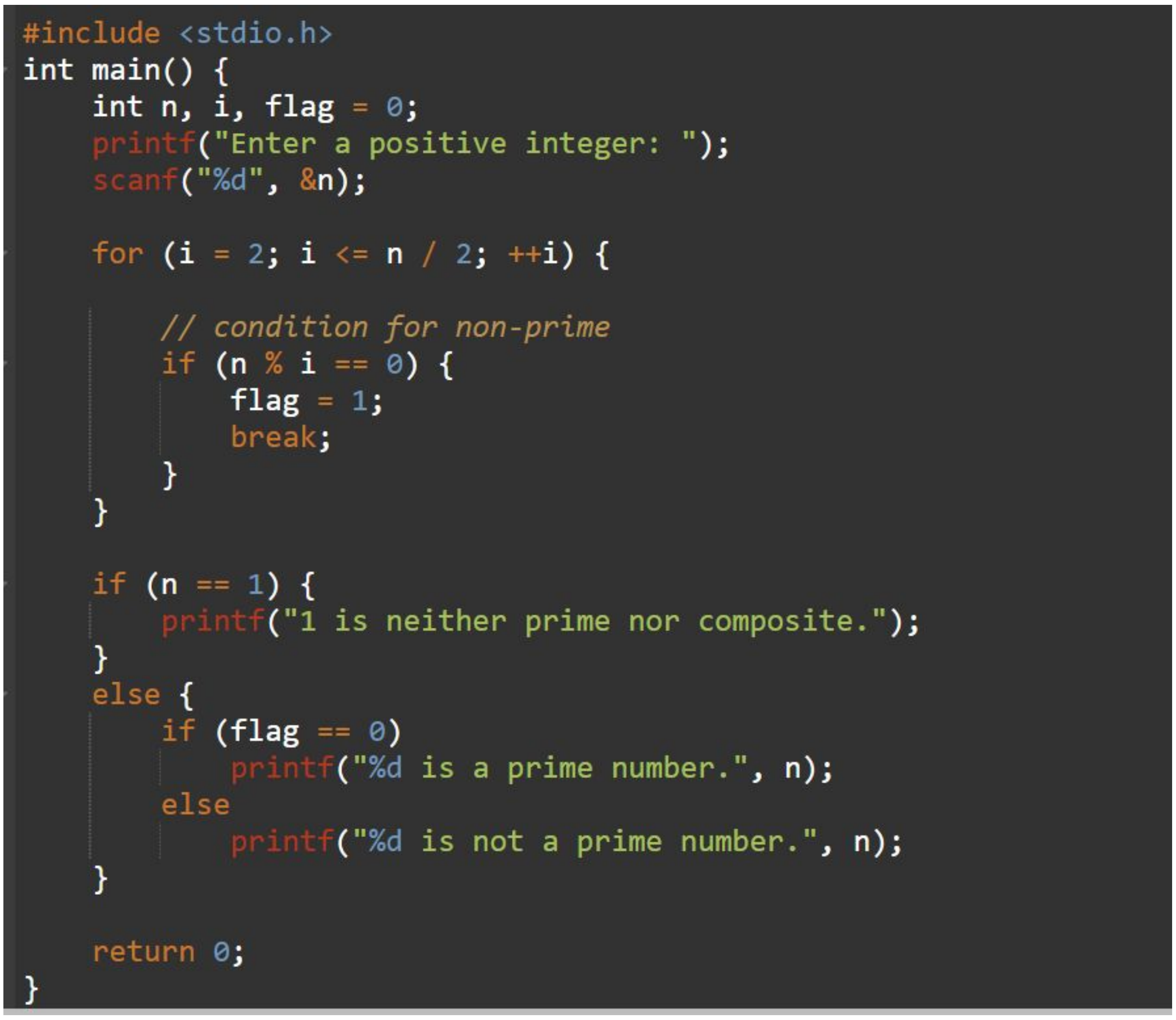
IMPLEMENTATION OF LEXICAL ANALYSER

AIM: To write a program to implement lexical analyser

PROGRAM:
```
#include<bits/stdc++.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

using namespace std;

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],b[30], logical_op[] =
"><",math_op[]="+-*/=",numer[]=".0123456789",other[]=",;\(){}[]":";
   ifstream fin("lexicalinput.txt");
   int mark[1000]={0};
   int i,j=0,kc=0,ic=0,lc=0,mc=0,nc=0,oc=0,aaa=0;
   vector < string > k;
   vector<char >id;
   vector<char>lo;
   vector<char>ma;
   vector<string>nu;
    vector<char>ot;
   if(!fin.is_open()){
      cout<<"error while opening the file\n";
      exit(0);
   }
```

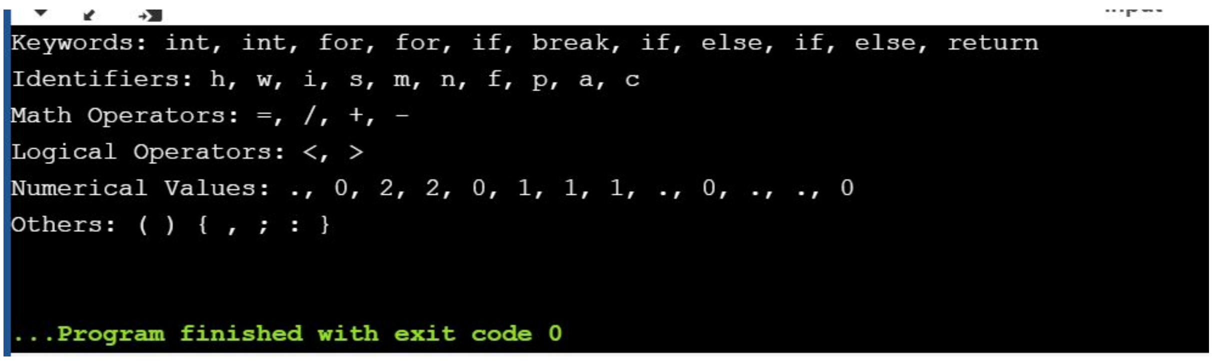
```
    while(!fin.eof()){
           ch = fin.get();
          for(i = 0; i < 12; ++i){
               if(ch == other[i]){
                    int aa=ch;
                if(mark[aa]!=1){
                    ot.push_back(ch);
                    mark[aa]=1;
                    ++oc;
                }
               }
        }

        for(i = 0; i < 5; ++i){
               if(ch == math_op[i]){
                    int aa=ch;
                if(mark[aa]!=1){
                    ma.push_back(ch);
                    mark[aa]=1;
                    ++mc;
                }
               }
        }
        for(i = 0; i < 2; ++i){
               if(ch == logical_op[i]){
                    int aa=ch;
                if(mark[aa]!=1){
                    lo.push_back(ch);
                    mark[aa]=1;
                    ++lc;
                }
               }

        }
        if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' || ch=='6' || ch=='7' || ch=='8' || ch=='9' ||
ch=='.' ||ch == ' ' || ch == '\n' || ch == ';'){

            if(ch=='0' || ch=='1' || ch=='2' || ch=='3' || ch=='4' || ch=='5' || ch=='6' || ch=='7' || ch=='8' ||
ch=='9' || ch=='.')b[aaa++]=ch;
            if((ch == ' ' || ch == '\n' || ch == ';') && (aaa != 0)){
                   b[aaa] = '\0';
                   aaa = 0;
                   char arr[30];
                    strcpy(arr,b);
                        nu.push_back(arr);
                    ++nc;
                }
        }


           if(isalnum(ch)){
               buffer[j++] = ch;
           }
           else if((ch == ' ' || ch == '\n') && (j != 0)){
                   buffer[j] = '\0';
                   j = 0;

                   if(isKeyword(buffer) == 1){

                       k.push_back(buffer);
                       ++kc;
                   }
                   else{



                    if(buffer[0]>=97 && buffer[0]<=122) {
                        if(mark[buffer[0]-'a']!=1){
                        id.push_back(buffer[0]);
                        ++ic;
                        mark[buffer[0]-'a']=1;
                       }

                    }

                   }

           }

    }

    fin.close();
    printf("Keywords: ");
     for(int f=0;f<kc;++f){
            if(f==kc-1){
                cout<<k[f]<<"\n";
            }
            else {
                cout<<k[f]<<", ";
            }
    }
    printf("Identifiers: ");
     for(int f=0;f<ic;++f){
        if(f==ic-1){
            cout<<id[f]<<"\n";
        }
        else {
            cout<<id[f]<<", ";
        }
    }
    printf("Math Operators: ");
    for(int f=0;f<mc;++f){
        if(f==mc-1){
            cout<<ma[f]<<"\n";
        }
        else {
            cout<<ma[f]<<", ";
        }
    }
    printf("Logical Operators: ");
    for(int f=0;f<lc;++f){
        if(f==lc-1){
            cout<<lo[f]<<"\n";
        }
        else {
            cout<<lo[f]<<", ";
        }

    }
    printf("Numerical Values: ");
    for(int f=0;f<nc;++f){
        if(f==nc-1){
            cout<<nu[f]<<"\n";
        }
        else {
            cout<<nu[f]<<", ";
        }

    }
    printf("Others: ");
    for(int f=0;f<oc;++f){
        if(f==oc-1){
            cout<<ot[f]<<"\n";
        }
        else {
            cout<<ot[f]<<" ";
        }

    }

    return 0;
}
```

INPUT:



```
#include <stdio.h>
int main() {
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 2; i <= n / 2; ++i) {

        // condition for non-prime
        if (n % i == 0) {
            flag = 1;
            break;
        }
    }

    if (n == 1) {
        printf("1 is neither prime nor composite.");
    }
    else {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }

    return 0;
}
```

OUTPUT:



```
Keywords: int, int, for, for, if, break, if, else, if, else, return
Identifiers: h, w, i, s, m, n, f, p, a, c
Math Operators: =, /, +, -
Logical Operators: <, >
Numerical Values: ., 0, 2, 2, 0, 1, 1, 1, ., 0, ., ., 0
Others: ( ) { , ; : }


...Program finished with exit code 0
```

RESULT : The lexical analyser is implemented successfully.