CD LAB : WEEK3 PROGRAMS AND SCREENSHOTS

SAGNIK CHATTERJEE
180905478
SEC-B
ROLLNO 61

1.Program to make a lexical analyzer which contains get Next token .

code:

```
/*
AUTHOR :SAGNIK CHATTERJEE
DATE : DEC 9,2020
Usage: ./getnexttoken input.txt

where input.txt is the input file

*/



#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

struct token{
    char token_name [100];
    int index;
    unsigned int row,col; //Line numbers.
    char type[100];
} token;

void print_token(struct token s){
```

```
   printf("<%s,%d,%d>",s.token_name,s.row,s.col);
   return;
}

void removeComments(){
     FILE *fa,*fb; int ca,cb;

     fa = fopen("input.txt","r");
     if(fa == NULL){
          printf("Cannot open\n");
               return;
     }
     fb = fopen("q1out.txt","w+");
     ca = getc(fa);
     while(ca != EOF){
          if( ca == ' '){
               putc(ca,fb);
               while(ca == ' '|| ca=='\t') ca = getc(fa);
          }
          if(ca == '/'){//checking for single and mutli line
comments
               cb = getc(fa);
               if(cb == '/'){//if another / then it makes a
single line commnet
                    while(ca != '\n') ca = getc(fa);
                    putc(ca,fb);
               }
               else if( cb == '*'){//else start for multine
comment
                    do{
                         while(ca != '*') ca = getc(fa);
```

```
                        ca = getc(fa);//traverse till the
end till we fiind another closing bracket
                    }while(ca != '/');
               }
               else{
                    putc(ca,fb);
                    putc(cb,fb);
               }
          }
          else putc(ca,fb);
          ca = getc(fa);
     }
     fclose(fa);
     fclose(fb);
}

void removeDirectives(){
     FILE *fa,*fb; int ca,cb;
     removeComments();

     fa = fopen("q1out.txt","r");
     if(fa == NULL){
          printf("Cannot open\n");
          return;
     }
     fb = fopen("q2tempout.txt","w+");
     ca = getc(fa);
     while(ca != EOF){
          if( ca == '#'){//removing all directives that are
present
               //directives start with #
               do{
```

```
                ca = getc(fa);
            }while(ca != '\n');
        }
        putc(ca,fb);
        ca = getc(fa);
    }

    fclose(fa);
    fclose(fb);

    fa= fopen("q2tempout.txt","r");
        if(fa == NULL){
        printf("Cannot open temporary file for writing
\n");
        return;
    }
    fb = fopen("q2out.txt","w+");
    ca = fgetc(fa);
    while(ca == '\n'){
            ca = fgetc(fa);
        }
    while(ca != EOF){

        putc(ca,fb);
        ca = fgetc(fa);
    }

    fclose(fa);
    fclose(fb);

    if (remove("q2tempout.txt") != 0) printf("Error\n");
}
```

```
//keywords
char key[32][10] = {
     "auto","double","int","struct","break","else","long",
      "switch","case","enum","register","typedef","char",
      "extern","return","union","const","float","short",
      "unsigned","continue","for","signed","void","default",
      "goto","sizeof","voltile","do","if","static","while"
};

int isKeyword(char* word){//check if keyword or not
     // printf("%s\n",word );
     for(int i = 0; i < 32; i++){
          if(strcmp(key[i], word) == 0) return 1;
     }
     return 0;
}


bool isDelimiter(char ch)
{
    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}')
        return true;
    return false;
}


bool isRealNumber(char* str)
{
```

```
    int i, len = strlen(str);
    bool hasDecimal = false;

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] !=
'5'
            && str[i] != '6' && str[i] != '7' && str[i] !=
'8'
            && str[i] != '9' && str[i] != '.' ||
            (str[i] == '-' && i > 0))
            return (false);
        if (str[i] == '.')
            hasDecimal = true;
    }
    return hasDecimal;
}


bool isInteger(char* str)
{
    int i, len = strlen(str);

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] !=
'5'
```

```
            && str[i] != '6' && str[i] != '7' && str[i] !=
'8'
            && str[i] != '9'|| str[i]=='.' || (str[i] == '-'
&& i > 0))
            return (false);
    }
    return (true);
}


int main(int argc, char const *argv[])
{
     FILE *fa,*fb;
     int ca,cb;
     removeDirectives();

     fa = fopen("q2out.txt","r");
     if(fa == NULL){
          printf("Cannot open\n");
          return 0;
     }
     char word[20], num[20];
     int i = 0;
     num[0]='\0';
     ca = getc(fa);
     int row=1, col=1;
     while(ca != EOF){
          struct token s;
          // line break
          if(ca == '\n'){
               row++;
               col = 1;
```

```
            printf("\n");
        }

        // check string
        else if(ca == '"'){
            strcpy(s.token_name,"string literal");
            s.row=row;
            s.col=col;
            print_token(s);
            ca = getc(fa);
            while(ca != '"'){
                col++;
                ca = getc(fa);
            }
        }

        else if(ca == ' ') {
            ca= getc(fa);
            col++;
            continue;
        }

        // is a word ->  keyword / variable
        else if(isalpha(ca)) {
            word[i++] = ca;
            while(isalpha(ca) || isdigit(ca) || ca ==
'_'){
                word[i++] = ca;
                ca = getc(fa);
                col++;
            }
            word[i]='\0';
```

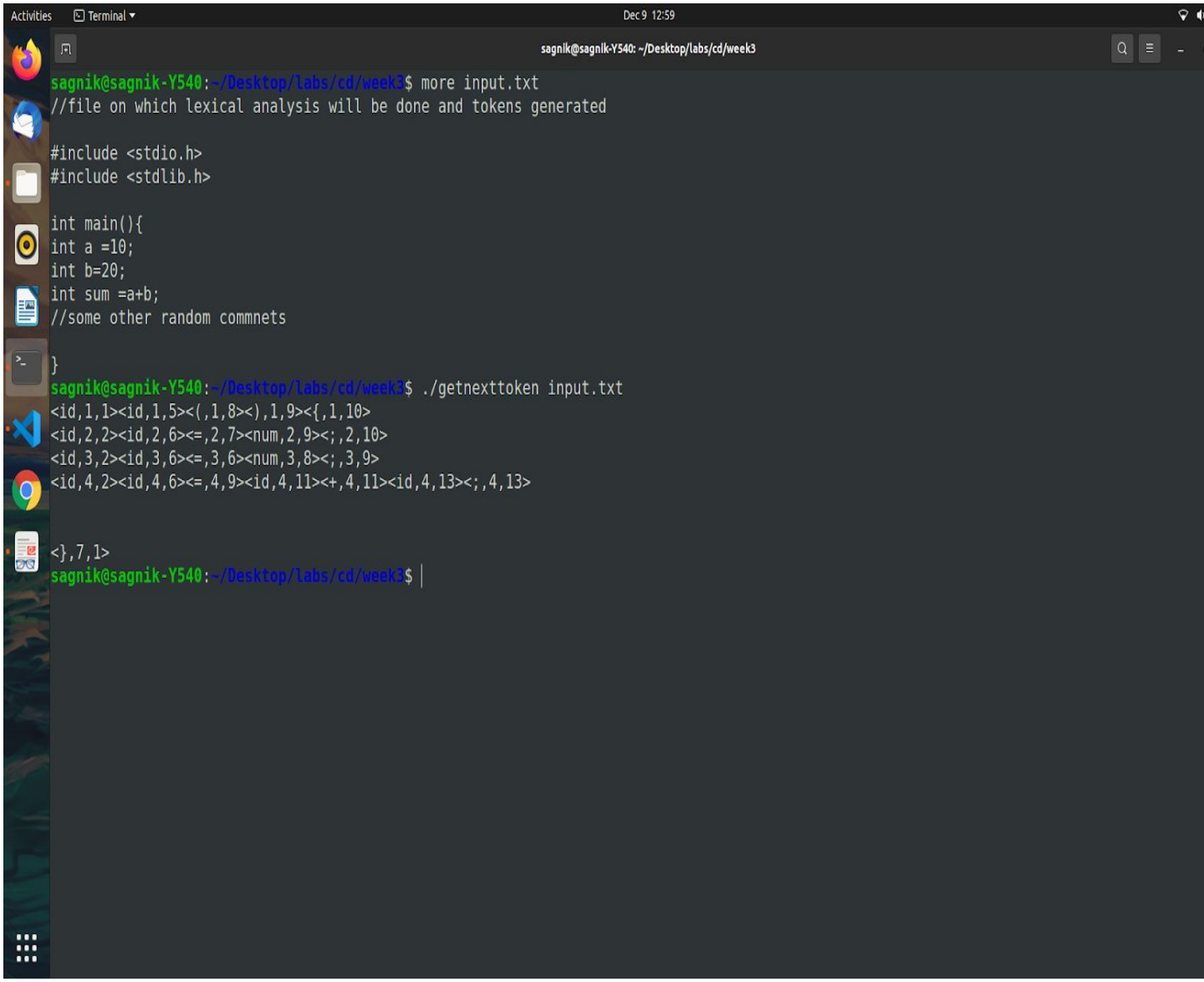
```
     if(isKeyword(word)){
               strcpy(s.token_name,word);
               s.row=row;
               s.col=col- (int)(strlen(word))+1;
               print_token(s);
   }
     else{
              strcpy(s.token_name,"id");
               s.row=row;
               s.col=col- (int)(strlen(word))+1;
               print_token(s);
     }
     i = 0;
     word[0]='\0';
     continue;
}

// is an Delimeter
else if(isDelimiter(ca)){
               s.token_name[0]=ca;
               s.token_name[1]='\0';
               s.row=row;
               s.col=col-1;
               print_token(s);
}

// is a number of any sort
else if(isdigit(ca)){
     num[i++] = ca;
     while(isdigit(ca)|| ca == '.'){
          num[i++] = ca;
          ca = getc(fa);
```

```
                col++;
            }
            num[i]='\0';
            if(isRealNumber(num) || isInteger(num)){
                    strcpy(s.token_name,"num");
                    s.row=row;
                    s.col=col- (int)(strlen(num))+1;
                    print_token(s);
            }
            i = 0;
            num[0]='\0';
            continue;
        }

        col++;
        ca = getc(fa);
    }
    fclose(fa);
    return 0;
}
```


Activities
Terminal
Dec 9 12:59
sagnik@sagnik-Y540: ~/Desktop/labs/cd/week3
sagnik@sagnik-Y540:~/Desktop/labs/cd/week3$ more input.txt
//file on which lexical analysis will be done and tokens generated
#include <stdio.h>
#include <stdlib.h>
int main(){
int a =10;
int b=20;
int sum =a+b;
//some other random commnets
}
sagnik@sagnik-Y540:~/Desktop/labs/cd/week3$ ./getnexttoken input.txt
<id,1,1><id,1,5><(,1,8><),1,9><{,1,10>
<id,2,2><id,2,6><=,2,7><num,2,9><;,2,10>
<id,3,2><id,3,6><=,3,6><num,3,8><;,3,9>
<id,4,2><id,4,6><=,4,9><id,4,11><+,4,11><id,4,13><;,4,13>
<},7,1>
sagnik@sagnik-Y540:~/Desktop/labs/cd/week3$