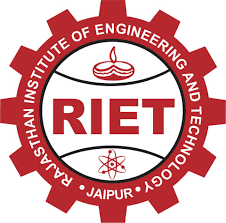
**RAJASTHAN INSTITUTE OF ENGINEERING & TECHNOLOGY,JAIPUR**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**COMPILER DESIGN LAB (5CS4-22)**

**SUBMITTED TO: SUBMITTED BY:**

**MR. MUKESH CHAUDHARY SIR NAME : ANKIT MALPANI**

**ROLL NO. : 18ERECS009(C.S.E)**

**III YEAR, V SEMESTER (B.TECH.)**

**BATCH: A1**

**INDEX**

|  |  |
| --- | --- |
| **SN** | **List of Experiments** |
| **1** | Introduction: Objective, scope and outcome of the course. |
| **2** | To identify whether given string is keyword or not. |
| **3** | Count total no. of keywords in a file. [Taking file from user] |
| **4** | Count total no of operators in a file. [Taking file from user] |
| **5** | Count total occurrence of each character in a given file. [Taking file from user] |
| **6** | Write a C program to insert, delete and display the entries in Symbol Table. |
| **7** | Write a LEX program to identify following:   1. Valid mobile number 2. Valid URL 3. Valid identifier 4. Valid date (dd/mm/yyyy) 5. Valid time (hh: mm: ss) |
| **8** | Write a lex program to count blank spaces, words, lines in a given file. |
| **9** | Write a lex program to count the no. of vowels and consonants in a C file. |
| **10** | Write a YACC program to recognize strings aaab, abbb using a^nb^n, where b>=0. |
| **11** | Write a YACC program to evaluate an arithmetic expression involving operators +, -, \* and /. |
| **12** | Write a YACC program to check validity of a strings abcd, aabbcd using grammar a^nb^nc^md^m, where n, m>0 |
| **13** | Write a C program to find first of any grammar. |

**INTRODUCTION**

**OBJECTIVE:**

This laboratory course is intended to make the students experiment on the basic techniques of compiler construction and tools that can used to perform syntax-directed translation of a high-level programming language into an executable code. Students will design and implement language processors in C by using tools to automate parts of the implementation process. This will provide deeper insights into the more advanced semantics aspects of programming languages, code generation, machine independent optimizations, dynamic memory allocation, and object orientation.

**OUTCOMES:**

Upon the completion of Compiler Design practical course, the student will be able to:

1. Understand the working of lex and YACC compiler for debugging of programs.

2. Understand & define the role of lexical analyzer, use of regular expression & transition diagrams.

3. Understand and use Context free grammar, and parse tree construction.

4. Learn & use the new tools and technologies used for designing a compiler.

5. Develop program for solving parser problems.

6. Learn how to write programs that execute faster

**EXPERIMENT NO. 1**

**AIM**: To identify whether a given string is a keyword or not.

**RESOURCE**: Turbo C++

**PROGRAM LOGIC:**

1. Read the file from user
2. Check whether the file exists or not.
3. Check whether the input is a keyword or string.
4. Print that result.

**PROCEDURE:**

1. Go to debug -> run or press CTRL + F9 to run the program

**PROGRAM:**

#include <stdio.h>

#include <string.h>

int main() {

char keyword[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"

} ;

char str[]="while";

int flag=0,i;

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

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

{

flag=1;

}

}

if(flag==1)

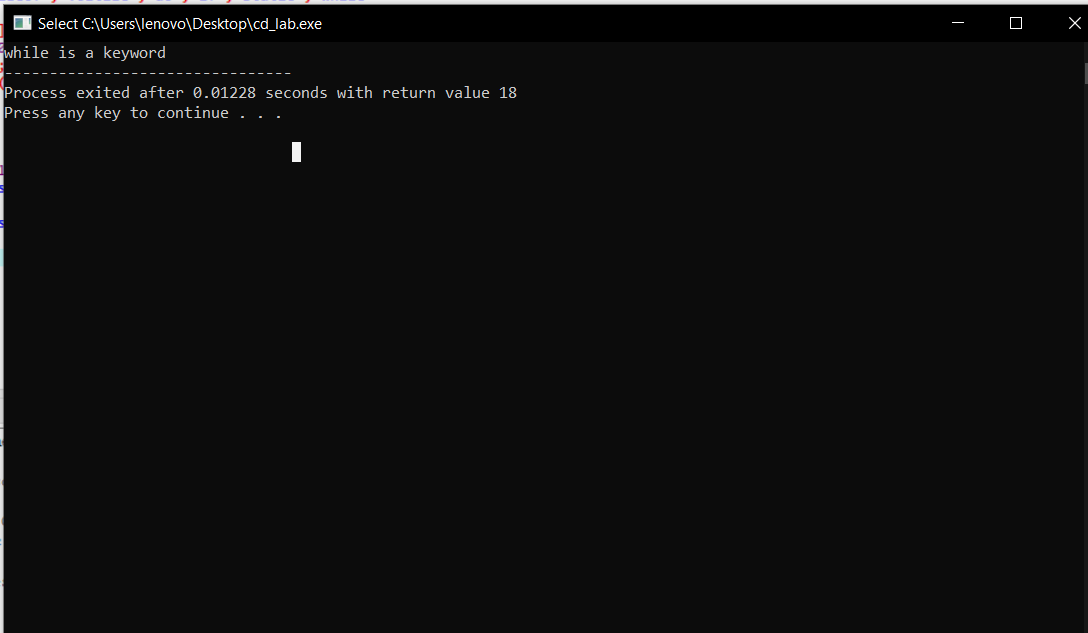
printf("%s is a keyword",str);

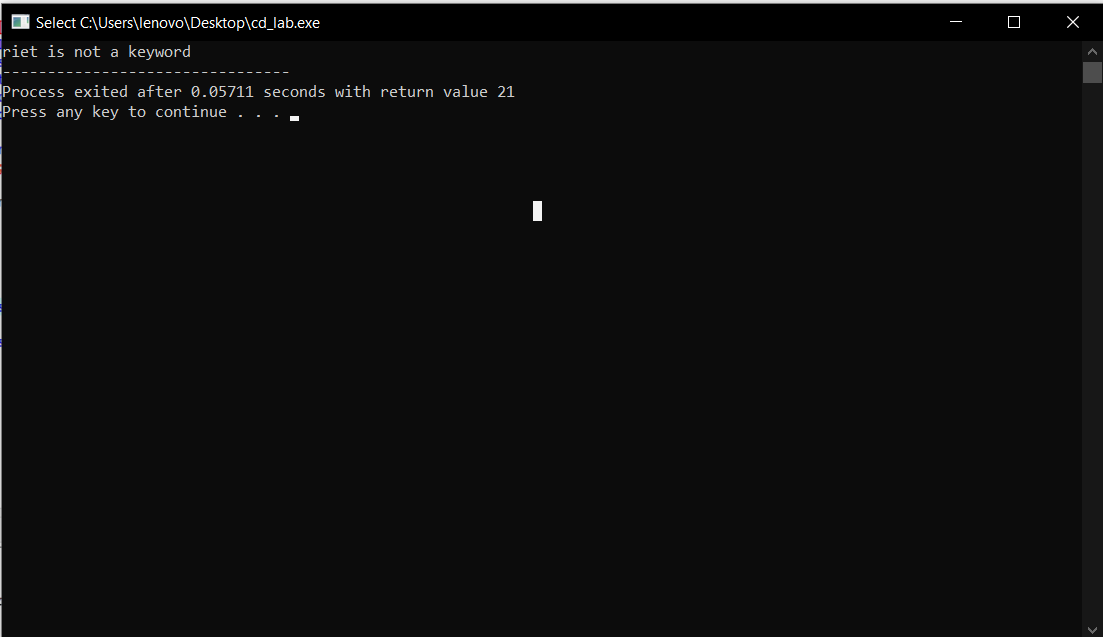
else

printf("%s is not a keyword",str);

}

**OUTPUT:**

****

****

**EXPERIMENT NO. 2**

**AIM**: To count total no. of keywords in a file.

**RESOURCE**: Turbo C++

**PROGRAM LOGIC**:

1. Read the file from user
2. Check whether the file exist or not.
3. Count the total keyword present.
4. Print that result.

**PROCEDURE**:

1. Go to debug -> run or press CTRL + F9 to run the program

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <conio.h>

#include<string.h>

int main ()

{

char file [50] = "access modifier in class.cpp";

printf("Enter File name : ");

gets(file);

printf("\n");

FILE \* fp = fopen(file, "r");

FILE \* kout = fopen("keyword.txt", "w");

FILE \* iout = fopen("identifier.txt", "w");

if (fp == NULL)

{

printf("\n'%s' file not found...\n", file);

getch();

return 1;

}

int k, result, count, kcount;

char c, str[10];

kcount = count = 0;

char keywords[][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" };

while((c = fgetc(fp)) != EOF)

{ if(c == ' ' || c == '\n')

{ ++count;

printf("%d). %s\n", count, str);

for(k=0; k<32; k++)

{

result = strcmp(keywords[k], str);

if (result == 0)

{

++kcount;

printf(str);

printf("\n");

break;

}

}

if (result == 0)

fprintf(kout, "%s\n", str);

else

fprintf(iout, "%s\n", str);

strcpy(str,"");

}

else

{

if (isalpha(c) && c!=' ')

strncat(str, &c, 1);

}

}

fclose(fp);

fclose(kout);

fclose(iout);

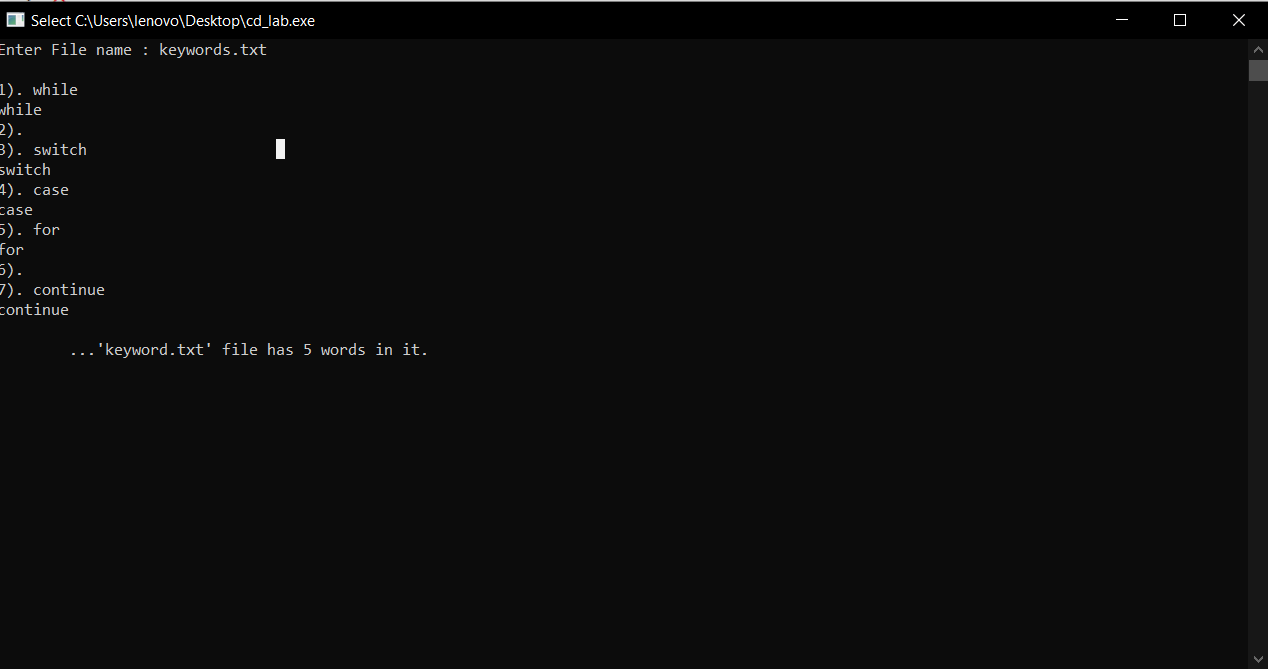
printf("\n\t...'%s' file has %d words in it.\n", "keyword.txt", kcount);

getch();

return 0;

}

**OUTPUT**:



**EXPERIMENT NO. 3**

**AIM:** To count total no. of operators in a file.

**RESOURCE**: Turbo C++

**PROGRAM LOGIC:**

1. Read the file from user
2. Check whether the file exist or not.
3. Count the total operator present.
4. Print that result.

**PROCEDURE:**

1. Go to debug -> run or press CTRL + F9 to run the program

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int main()

{

char file[50];

printf("Enter File name : ");

gets(file);

printf("\n");

FILE \* fp = fopen(file, "r");

if (fp == NULL)

{

printf("\n'%s' file not found...\n", file);

getch();

return 1;

}

int k, result, count, kcount;

char c, str[10];

kcount = count = 0;

while((c = fgetc(fp)) != EOF)

{ if(c == '+' || c == '-' || c == '%' || c == '\*' || c == '/')

++count;

}

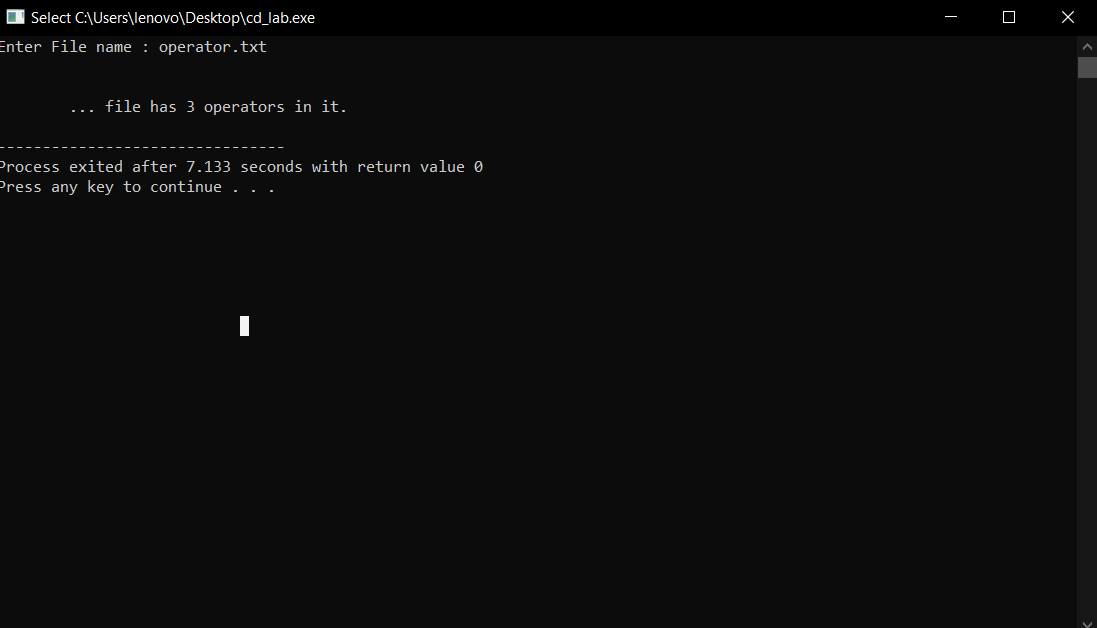
fclose(fp);

printf("\n\t... file has %d operators in it.\n", count);

return 0;

}

**OUTPUT:**

****

**EXPERIMENT NO. 4**

**AIM:** Count total occurrence of each character in a given file.

**RESOURCE** : Dev C++

**PROGRAM LOGIC:**

1. Read the file from user
2. Check whether the file exist or not.
3. Count the total character present.
4. Print that result.

**PROCEDURE:**

1. Go to debug -> run or press CTRL + F9 to run the program

**PROGRAM:**

#include <stdio.h>

#define MAX\_FILE\_NAME 100

int main()

{

FILE\* fp;

int count = 0;

char filename[MAX\_FILE\_NAME];

// To store a character read from file //

char c;

printf("Enter file name: ");

scanf("%s", filename);

fp = fopen(filename, "r");

if (fp == NULL) {

printf("Could not open file %s",

filename);

return 0;

}

// Extract characters from file

// and store in character c

for (c = getc(fp); c != EOF; c = getc(fp))

// Increment count for this character

count = count + 1;

fclose(fp);

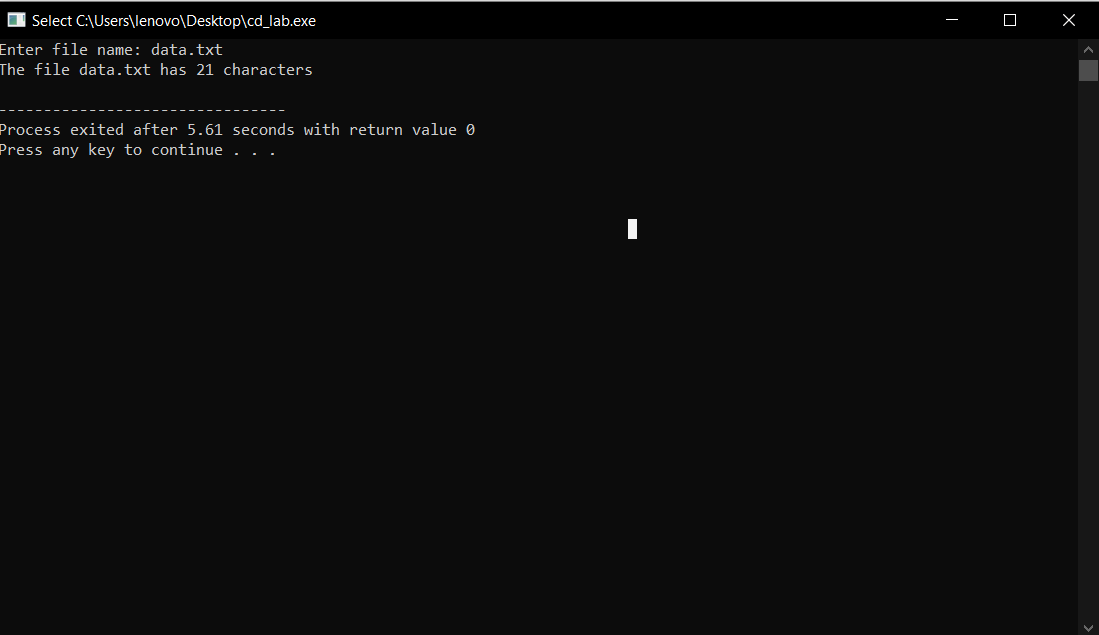
printf("The file %s has %d characters\n ",

filename, count);

return 0;

}

**OUTPUT:**

****

**EXPERIMENT NO. 5**

**AIM:** To write a program for implementing Symbol Table using C.

**ALGORITHM:**

**Step 1:** Start the program for performing insert, display, delete, search and modify option in symbol table

**Step 2:** Define the structure of the Symbol Table

**Step 3:** Enter the choice for performing the operations in the symbol Table

**Step 4:** If the entered choice is 1, search the symbol table for the symbol to be inserted. If the symbol is already present, it displays “Duplicate Symbol”. Else, insert the symbol and the corresponding address in the symbol table.

**Step 5:** If the entered choice is 2, the symbols present in the symbol table are displayed.

**Step 6:** If the entered choice is 3, the symbol to be deleted is searched in the symbol table.

**Step 7:** If it is not found in the symbol table it displays “Label Not found”. Else, the symbol is deleted.

**Step 8:** If the entered choice is 5, the symbol to be modified is searched in the symbol table

**PROGRAM CODE:**

//Implementation of symbol table//

#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

void main ()

{

int i=0,j=0,x=0,n;

void \*p,\*add[5];

char ch, srch,b[15],d[15],c;

printf("Expression terminated by $:");

while((c=getchar())!='$')

{

b[i]=c;

i++;

}

n=i-1;

printf("Given Expression:"); i=0;

while(i<=n)

{

printf("%c",b[i]); i++;

}

printf("\n Symbol Table\n");

printf("Symbol \t addr \t type");

while(j<=n)

{

c=b[j];

if(isalpha(toascii(c)))

{

p=malloc(c);

add[x]=p;

d[x]=c;

printf("\n%c \t %d \t identifier\n",c,p);

x++;

j++;

}

Else

{

ch=c;

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

{

p=malloc(ch);

add[x]=p;

d[x]=ch;

printf("\n %c \t %d \t operator\n",ch,p);

x++;

j++;

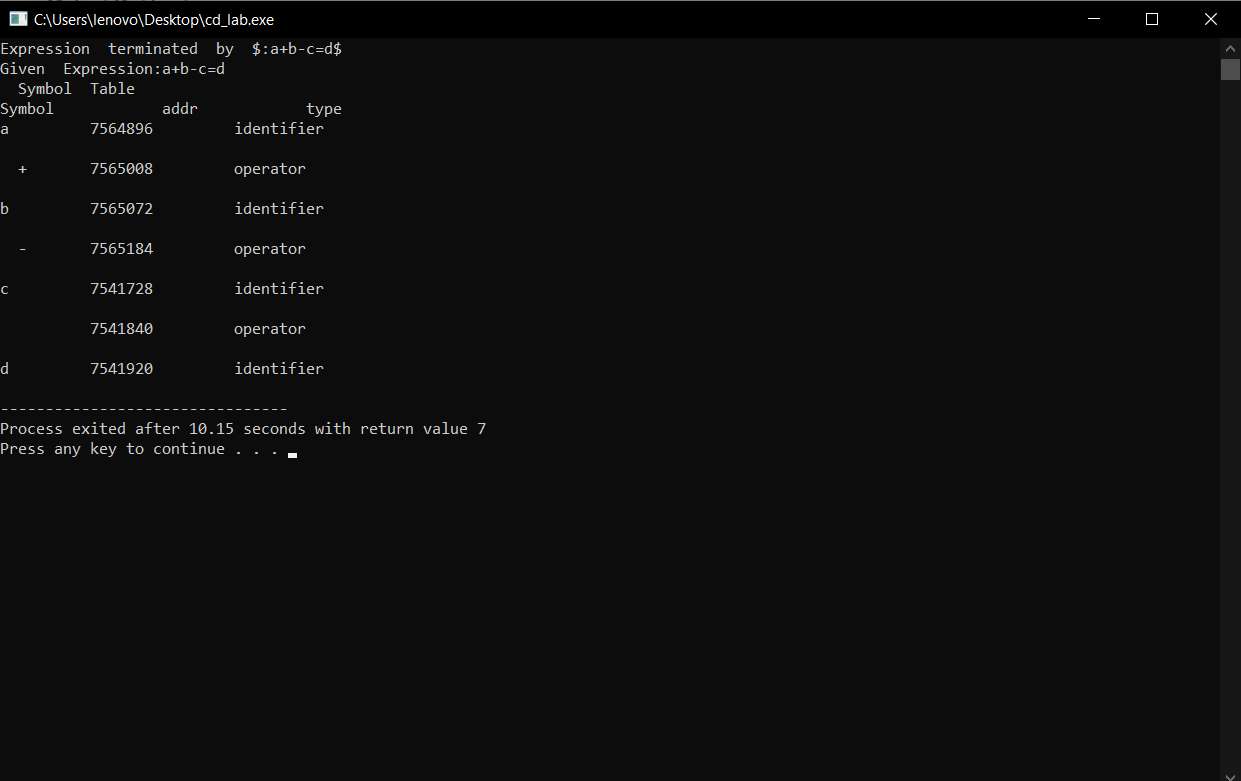
}

}

}

}

**OUTPUT:**

****