1. // Online C compiler to run C program online

#include <stdio.h>

#include<string.h>

int addsub();

int muldiv();

int term();

char input[101];

int pos = 0;

int term(){

int n = 0;

if(input[pos] == '('){

pos++;

n = addsub();

if(input[pos] == ')'){

pos++;

return n;

}

}else{

while('0' <= input[pos] && input[pos] <= '9'){

n = n\*10 + (input[pos] - '0');

pos++;

}

}

return n;

}

int muldiv(){

int first,second;

first = term();

for(;;){

if(input[pos] == '\*'){

pos++;

second = term();

first \*= second;

}else if(input[pos] == '/'){

pos++;

second = term();

first /= second;

}else{

return first;

}

}

}

int addsub(){

int first,second;

first = muldiv();

for(;;){

if(input[pos] == '+'){

pos++;

second = muldiv();

first += second;

}else if(input[pos] == '-'){

pos++;

second = muldiv();

first -= second;

}else{

return first;

}

}

}

int main(){

int n,i,j;

printf("Input an expression using +, -, \*, / operators:\n");

scanf("%s",input);

printf("%d\n",addsub());

return 0; }

2.a /lex program to count number of words/

%{

#include<stdio.h>

#include<string.h>

int i = 0;

%}

/\* Rules Section\*/

%%

([a-zA-Z0-9])\* {i++;} /\* Rule for counting

number of words\*/

"\n" {printf("%d\n", i); i = 0;}

%%

int yywrap(void){}

int main()

{

// The function that starts the analysis

yylex();

return 0;

}

2b. %{

int vow\_count=0;

int const\_count =0;

%}

%%

[aeiouAEIOU] {vow\_count++;}

[a-zA-Z] {const\_count++;}

%%

int yywrap(){}

int main()

{

printf("Enter the string of vowels and consonants:");

yylex();

printf("Number of vowels are: %d\n", vow\_count);

printf("Number of consonants are: %d\n", const\_count);

return 0;

}

10. #include <stdio.h>

#include <stdbool.h>

#include <string.h>

// Function to check the character

// is an alphabet or not

bool isChar(char c)

{

return ((c >= 'a' && c <= 'z')

|| (c >= 'A' && c <= 'Z'));

}

// Function to check the character

// is an digit or not

bool isDigit(const char c)

{

return (c >= '0' && c <= '9');

}

// Function to check email id is

// valid or not

bool is\_valid(char email[])

{

// Check the first character

// is an alphabet or not

if (!isChar(email[0])) {

// If it's not an alphabet

// email id is not valid

return false;

}

// Variable to store position

// of At and Dot

int At = -1, Dot = -1;

// Traverse over the email id

// string to find position of

// Dot and At

for (int i = 0; i < strlen(email); i++) {

// If the character is '@'

if (email[i] == '@') {

At = i;

}

// If character is '.'

else if (email[i] == '.') {

Dot = i;

}

}

// If At or Dot is not present

if (At == -1 || Dot == -1)

return false;

// If Dot is present before At

if (At > Dot)

return false;

// If Dot is present at the end

return !(Dot >= (strlen(email) - 1));

}

// Driver Code

int main()

{

// Given string email

char email[] = "contribute@geeksforgeeks.org";

// Function Call

bool ans = is\_valid(email);

// Print the result

if (ans) {

printf("%s : valid\n", email);

}

else {

printf("%s : invalid\n", email);

}

return 0;

}

9. #include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX\_LENGTH 50

/\* function to check if a string is a valid operator \*/

int is\_operator(char \*str) {

if (strcmp(str, "+") == 0 || strcmp(str, "-") == 0 ||

strcmp(str, "\*") == 0 || strcmp(str, "/") == 0 ||

strcmp(str, "%") == 0 || strcmp(str, "=") == 0 ||

strcmp(str, "==") == 0 || strcmp(str, "!=") == 0 ||

strcmp(str, ">") == 0 || strcmp(str, "<") == 0 ||

strcmp(str, ">=") == 0 || strcmp(str, "<=") == 0) {

return 1;

}

return 0;

}

/\* main function \*/

int main() {

char input[MAX\_LENGTH];

char operator[MAX\_LENGTH];

int i, j = 0;

printf("Enter an expression: ");

fgets(input, MAX\_LENGTH, stdin);

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

if (input[i] == '+' || input[i] == '-' || input[i] == '\*' ||

input[i] == '/' || input[i] == '%' || input[i] == '=' ||

input[i] == '>' || input[i] == '<' || input[i] == '!') {

operator[j++] = input[i];

operator[j] = '\0';

if (!is\_operator(operator)) {

printf("Error: '%s' is not a valid operator.\n", operator);

exit(0);

}

}

else {

j = 0;

}

}

printf("All operators in the expression are valid.\n");

return 0;

}

8.1

/\* LEX code to count the frequency

of the given word in a file \*/

/\* Definition section \*/

/\* variable word indicates the word

whose frequency is to be count \*/

/\* variable count is used to store the

frequency of the given word \*/

%{

#include<stdio.h>

#include<string.h>

char word [] = "geeks";

int count = 0;

%}

/\* Rule Section \*/

/\* Rule 1 compares the matched token with the

word to count and increments the count variable

on successful match \*/

/\* Rule 2 matches everything other than string

(consists of alphabets only ) and do nothing \*/

%%

[a-zA-Z]+ { if(strcmp(yytext, word)==0)

count++; }

. ;

%%

int yywrap()

{

return 1;

}

/\* code section \*/

int main()

{

extern FILE \*yyin, \*yyout;

/\* open the input file

in read mode \*/

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

yylex();

printf("%d", count);

}

8.2

/lex code to find the length of the longest word/

% {

int counter = 0; %

}

%

% [a - zA - Z] + {

if (yyleng > counter) {

counter = yyleng;

}

} %

%

main() {

yylex();

printf("largest: %d", counter);

printf("\n");

}

6. parser

#include <stdio.h>

#include <string.h>

#define SUCCESS 1

#define FAILED 0

int E(), Edash(), T(), Tdash(), F();

const char \*cursor;

char string[64];

int main()

{

puts("Enter the string");

// scanf("%s", string);

sscanf("i+(i+i)\*i", "%s", string);

cursor = string;

puts("");

puts("Input Action");

puts("--------------------------------");

if (E() && \*cursor == '\0') {

puts("--------------------------------");

puts("String is successfully parsed");

return 0;

} else {

puts("--------------------------------");

puts("Error in parsing String");

return 1;

}

}

int E()

{

printf("%-16s E -> T E'\n", cursor);

if (T()) {

if (Edash())

return SUCCESS;

else

return FAILED;

} else

return FAILED;

}

int Edash()

{

if (\*cursor == '+') {

printf("%-16s E' -> + T E'\n", cursor);

cursor++;

if (T()) {

if (Edash())

return SUCCESS;

else

return FAILED;

} else

return FAILED;

} else {

printf("%-16s E' -> $\n", cursor);

return SUCCESS;

}

}

int T()

{

printf("%-16s T -> F T'\n", cursor);

if (F()) {

if (Tdash())

return SUCCESS;

else

return FAILED;

} else

return FAILED;

}

int Tdash()

{

if (cursor == ' ') {

printf("%-16s T' -> \* F T'\n", cursor);

cursor++;

if (F()) {

if (Tdash())

return SUCCESS;

else

return FAILED;

} else

return FAILED;

} else {

printf("%-16s T' -> $\n", cursor);

return SUCCESS;

}

}

int F()

{

if (\*cursor == '(') {

printf("%-16s F -> ( E )\n", cursor);

cursor++;

if (E()) {

if (\*cursor == ')') {

cursor++;

return SUCCESS;

} else

return FAILED;

} else

return FAILED;

} else if (\*cursor == 'i') {

cursor++;

printf("%-16s F -> i\n", cursor);

return SUCCESS;

} else

return FAILED;

}

5.

#include<stdio.h>

#include<conio.h>

char array[10][20], temp[10];

int c, n;

void fun(int, int[]);

int fun2(int i, int j, int p[], int);

int main() {

int p[2], i, j;

printf("Enter the no. of productions :");

scanf("%d", & n);

printf("Enter the productions :\n");

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

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

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

c = -1, p[0] = -1, p[1] = -1;

fun(i, p);

printf("First(%c) : [ ", array[i][0]);

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

printf("%c,", temp[j]);

printf("\b ].\n");

getch();

}

}

int fun2(int i, int j, int p[], int key) {

int k;

if (!key) {

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

if (array[i][j] == array[k][0])

break;

p[0] = i;

p[1] = j + 1;

fun(k, p);

return 0;

} else {

for (k = 0; k <= c; k++) {

if (array[i][j] == temp[k])

break;

}

if (k > c) return 1;

else return 0;

}

}

void fun(int i, int p[]) {

int j, k, key;

for (j = 2; array[i][j] != NULL; j++) {

if (array[i][j - 1] == '/') {

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

key = 0;

fun2(i, j, p, key);

} else {

key = 1;

if (fun2(i, j, p, key))

temp[++c] = array[i][j];

if (array[i][j] == '@' && p[0] != -1) //taking '@' as null symbol

{

if (array[p[0]][p[1]] >= 'A' && array[p[0]][p[1]] <= 'Z') {

key = 0;

fun2(p[0], p[1], p, key);

} else

if (array[p[0]][p[1]] != '/' && array[p[0]][p[1]] != NULL) {

if (fun2(p[0], p[1], p, key))

temp[++c] = array[p[0]][p[1]];

}

}

}

}

}

}

7. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

int main() {

char var\_name[50];

char value[50];

int i, len;

int decimal = 0;

printf("Enter the variable name: ");

scanf("%s", var\_name);

printf("Enter the variable value: ");

scanf("%s", value);

len = strlen(value);

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

if(!isdigit(value[i]) && value[i] != '.') {

printf("Invalid input\n");

return 0;

}

if(value[i] == '.') {

decimal++;

if(decimal > 1) {

printf("Invalid input\n");

return 0;

}

}

}

if(decimal == 1) {

float num = atof(value);

printf("%s is a float with value %.2f\n", var\_name, num);

} else {

int num = atoi(value);

printf("%s is an integer with value %d\n", var\_name, num);

}

return 0;

}

4.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node {

char sym[10];

char desc[20];

struct node \*next;

};

int size=0;

struct node\* head;

int search(char \*s);

void traverse()

{

struct node \*temp = head;

printf("\nSYMBOL TABLE");

printf("\n============");

printf("\nSymbol |\tDescription");

while(temp != NULL) {

printf("\n%s \t|\t%s",temp->sym,temp->desc);

temp = temp->next;

}

}

void insert(char s[], char d[])

{

if(search(s))

printf("\n\nWARNING: Duplicate symbol. \n");

else{

struct node \*newNode;

newNode = (struct node\*)malloc(sizeof(struct node));

strcpy(newNode->sym, s);

strcpy(newNode->desc, d);

newNode->next = NULL;

if(size==0)

{

head = newNode;

}

else{

struct node \*temp = head;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

}

size++;

}

}

void del(char \*s)

{

if(search(s))

{

struct node\* current = head;

struct node\* prev = head;

while (current != NULL) {

if (strcmp(current->sym,s) == 0)

break;

prev=current;

current = current->next;

}

if(prev==head)

head=prev->next;

else

prev->next=current->next;

}

else{

printf("\n\nWARNING: Label not found.\n");

}

}

int search(char \*s)

{

struct node\* current = head;

while (current != NULL) {

if (strcmp(current->sym,s) == 0)

return 1;

current = current->next;

}

return 0;

}

void modify(char \*s)

{

char choice;

if(search(s))

{

struct node\* current = head;

while (current != NULL) {

if (strcmp(current->sym,s) == 0)

break;

current = current->next;

}

printf("\nDo you want to modify the label? [Y/N] ");

scanf("%c", &choice);

fflush(stdin);

if(choice=='Y' || choice=='y')

{

printf("Enter new label: ");

scanf("%[^\n]%\*c", current->sym);

}

printf("\nDo you want to modify the description? [Y/N] ");

scanf("%c", &choice);

fflush(stdin);

if(choice=='Y'|| choice=='y')

{

printf("Enter new description: ");

scanf("%[^\n]%\*c", current->desc);

}

}

else{

printf("\n\nWARNING: Label not found.\n");

}

}

int main(){

char s[10];

char d[20];

int i=1;

while(i>0 && i<6)

{

printf("\n\nCHOICE MENU:");

printf("\n1: To insert into symbol table.");

printf("\n2: To display symbol table.");

printf("\n3: To delete a symbol symbol table.");

printf("\n4: To search for a symbol in symbol table.");

printf("\n5: To modify symbol table.");

printf("\n6: Exit the program.");

printf("\n\nEnter your choice: ");

scanf("%d",&i);

fflush(stdin);

switch(i)

{

case 1:

printf("\nEnter the symbol to be inserted: ");

// fgets(s, 10, stdin);

scanf("%[^\n]%\*c", s);

printf("Enter the description: ");

// fgets(d, 20, stdin);

scanf("%[^\n]%\*c", d);

insert(s,d);

break;

case 2:

traverse();

break;

case 3:

printf("\nEnter the symbol to be deleted: ");

scanf("%[^\n]%\*c", s);

del(s);

break;

case 4:

printf("\nEnter the symbol to be searched: ");

scanf("%[^\n]%\*c", s);

if(search(s))

printf("\nSUCCESS: %s found in symbol table.",s);

else

printf("\nWARNING: %s not found in symbol table.",s);

break;

case 5:

printf("\nEnter the symbol to be modified: ");

scanf("%[^\n]%\*c", s);

modify(s);

break;

default:

printf("\nProgram Terminated. ");

}

}

}

3.

#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

// Returns 'true' if the character is a DELIMITER.

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);

}

// Returns 'true' if the character is an OPERATOR.

bool isOperator(char ch)

{

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

ch == '/' || ch == '>' || ch == '<' ||

ch == '=')

return (true);

return (false);

}

// Returns 'true' if the string is a VALID IDENTIFIER.

bool validIdentifier(char\* str)

{

if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||

str[0] == '3' || str[0] == '4' || str[0] == '5' ||

str[0] == '6' || str[0] == '7' || str[0] == '8' ||

str[0] == '9' || isDelimiter(str[0]) == true)

return (false);

return (true);

}

// Returns 'true' if the string is a KEYWORD.

bool isKeyword(char\* str)

{

if (!strcmp(str, "if") || !strcmp(str, "else") ||

!strcmp(str, "while") || !strcmp(str, "do") ||

!strcmp(str, "break") ||

!strcmp(str, "continue") || !strcmp(str, "int")

|| !strcmp(str, "double") || !strcmp(str, "float")

|| !strcmp(str, "return") || !strcmp(str, "char")

|| !strcmp(str, "case") || !strcmp(str, "char")

|| !strcmp(str, "sizeof") || !strcmp(str, "long")

|| !strcmp(str, "short") || !strcmp(str, "typedef")

|| !strcmp(str, "switch") || !strcmp(str, "unsigned")

|| !strcmp(str, "void") || !strcmp(str, "static")

|| !strcmp(str, "struct") || !strcmp(str, "goto"))

return (true);

return (false);

}

// Returns 'true' if the string is an INTEGER.

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] == '-' && i > 0))

return (false);

}

return (true);

}

// Returns 'true' if the string is a REAL NUMBER.

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);

}

// Extracts the SUBSTRING.

char\* subString(char\* str, int left, int right)

{

int i;

char\* subStr = (char\*)malloc(

sizeof(char) \* (right - left + 2));

for (i = left; i <= right; i++)

subStr[i - left] = str[i];

subStr[right - left + 1] = '\0';

return (subStr);

}

// Parsing the input STRING.

void parse(char\* str)

{

int left = 0, right = 0;

int len = strlen(str);

while (right <= len && left <= right) {

if (isDelimiter(str[right]) == false)

right++;

if (isDelimiter(str[right]) == true && left == right) {

if (isOperator(str[right]) == true)

printf("'%c' IS AN OPERATOR\n", str[right]);

right++;

left = right;

} else if (isDelimiter(str[right]) == true && left != right

|| (right == len && left != right)) {

char\* subStr = subString(str, left, right - 1);

if (isKeyword(subStr) == true)

printf("'%s' IS A KEYWORD\n", subStr);

else if (isInteger(subStr) == true)

printf("'%s' IS AN INTEGER\n", subStr);

else if (isRealNumber(subStr) == true)

printf("'%s' IS A REAL NUMBER\n", subStr);

else if (validIdentifier(subStr) == true

&& isDelimiter(str[right - 1]) == false)

printf("'%s' IS A VALID IDENTIFIER\n", subStr);

else if (validIdentifier(subStr) == false

&& isDelimiter(str[right - 1]) == false)

printf("'%s' IS NOT A VALID IDENTIFIER\n", subStr);

left = right;

}

}

return;

}

// DRIVER FUNCTION

int main()

{

// maximum length of string is 100 here

char str[100] = "int a = b + 1c; ";

parse(str); // calling the parse function

return (0);

}