



FACULTY OF ENGINEERING & TECHNOLOGY  
Subject Name: Compiler Design  
Laboratory  
Subject Code: 303105350  
B.Tech. 3<sup>rd</sup> Year 6<sup>th</sup> Semester  
COMPILER DESIGN LAB (303105350)

**FACULTY OF ENGINEERING AND TECHNOLOGY**  
**PARUL INSTITUTE OF ENGINEERING & TECHNOLOGY**  
**COMPUTER SCIENCE & ENGINEERING DEPARTMENT**

**COMPILER DESIGN LABORATORY**  
**(303105350)**

**6th SEMESTER**

**6CSE 5 – Batch 1**

**NAME: SHAH SAURABH KAMESHWAR**

**ENROLLMENT NO: 2203031050606**



FACULTY OF ENGINEERING & TECHNOLOGY  
Subject Name: Compiler Design  
Laboratory  
Subject Code: 303105350  
B.Tech. 3<sup>rd</sup> Year 6<sup>th</sup> Semester  
COMPILER DESIGN LAB (303105350)

### TABLE OF CONTENT

Sr. No	Experiment Title	Page No		Date of Performance	Date of Assessment	Marks (out of 10)	Sign
		From	To				
1	Program to implement Lexical Analyzer.						
2	Program to count digits, vowels and symbols in C.						
3	Program to check validation of User Name and Password in C.						
4	Program to implement Predictive Parsing LL(1) in C.						
5	Program to implement Recursive Descent Parsing in C.						
6	Program to implement Operator Precedence Parsing in C.						
7	Program to implement LALR Parsing in C.						
8	To Study about Lexical Analyzer Generator(LEX) and Flex(Fast Lexical						

	Analyzer)						
9	<p>Implement following programs using Lex.</p> <p>a. Create a Lexer to take input from text file and count no of characters, no. of lines &amp; no. of words.</p> <p>b. Write a Lex program to count number of vowels and consonants in a given input string.</p>						
10	<p>Implement following programs using Lex.</p> <p>a. Write a Lex program to print out all numbers from the given file.</p> <p>b. Write a Lex program to printout all HTML tags in file.</p> <p>c. Write a Lex program which adds line numbers to the given file and display the same onto the standard output.</p>						

## **PRACTICAL NO: 1**

### **AIM: Program to implement Lexical Analyzer.**

**Theory:** This program will tokenize simple expressions involving operators, integers, and identifiers. It will identify:

- Keywords: `int`, `if`
- Identifiers: Variable names (e.g., `x`, `y`)
- Operators: `+`, `-`, `*`, `/`
- Numbers: Integers (`10`, `100`)
- Punctuation: Semicolons (`;`)

### **Code:**

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

void identifyToken(char *token) {
    const char *keywords[] = {"int", "float", "if", "else", "while", "return"};
    int isKeyword = 0;

    for (int i = 0; i < sizeof(keywords) / sizeof(keywords[0]); i++) {
        if (strcmp(token, keywords[i]) == 0) {
            isKeyword = 1;
            break;
        }
    }

    if (isKeyword) {
        printf("Keyword: %s\n", token);
    } else if (isdigit(token[0])) {
        printf("Number: %s\n", token);
    } else if (isalpha(token[0]) || token[0] == '_') {
        printf("Identifier: %s\n", token);
    }
}
```

```
    } else {  
        printf("Unknown Token: %s\n", token);  
    }  
}  
  
void lexicalAnalyzer(const char *code) {  
    char token[100];  
    int i = 0, j = 0;  
  
    while (code[i] != '\0') {  
        if (isspace(code[i])) {  
            i++;  
            continue;  
        }  
  
        // Identify operators  
        if (strchr("+-*/=;", code[i])) {  
            printf("Operator or Separator: %c\n", code[i]);  
            i++;  
            continue;  
        }  
  
        // Extract identifiers, keywords, or numbers  
        if (isalpha(code[i]) || isdigit(code[i]) || code[i] == '_') {  
            j = 0;  
            while (isalnum(code[i]) || code[i] == '_') {  
                token[j++] = code[i++];  
            }  
            token[j] = '\0';  
            identifyToken(token);  
        } else {  
  
            printf("Unknown Character: %c\n", code[i]);  
            i++;  
        }  
    }  
}
```

```
int main() {  
    const char code[] = "int x = 10; float y = 20.5; if (x > y) return x + y;";  
    printf("Lexical Analysis of the code:\n");  
    lexicalAnalyzer(code);  
    return 0;  
}
```

### Output:

**Output** Clear

```
Lexical Analysis of the code:  
Keyword: int  
Identifier: x  
Operator or Separator: =  
Number: 10  
Operator or Separator: ;  
Keyword: float  
Identifier: y  
Operator or Separator: =  
Number: 20  
Unknown Character: .  
Number: 5  
Operator or Separator: ;  
Keyword: if  
Unknown Character: (  
Identifier: x  
Unknown Character: >  
Identifier: y  
Unknown Character: )  
Keyword: return  
Identifier: x  
Operator or Separator: +  
Identifier: y  
Operator or Separator: ;  
  
=== Code Execution Successful ===
```

## **PRACTICAL NO: 2**

### **AIM: Program to count digits, vowels and symbols in C.**

**Theory:** The program to count digits, vowels, and symbols in C is designed to analyze a given string and classify its characters into three categories: numerical digits (0-9), vowels (both uppercase and lowercase), and symbols (special characters, punctuation, or spaces). It processes the input character by character, incrementing counters for each category based on predefined rules. This program is particularly useful for text processing tasks where detailed string analysis is required, helping to extract specific patterns and insights from the data.

### **Code:**

#### **With user input**

```
#include <stdio.h>
#include <string.h>

void count(char* str)
{
    int vowels = 0, digits = 0, symbols = 0;

    int i;
    char ch;

    for (i = 0; str[i] != '\0'; i++) {
        ch = str[i];
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
            vowels++;
        }
        else if (ch >= '0' && ch <= '9') {
            digits++;
        }
        else if (!(ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || ch == ' ') {
            symbols++;
        }
    }
}
```

```
    }  
}  
  
printf("\nVowels: %d", vowels);  
printf("\nDigits: %d", digits);  
printf("\nSymbols: %d", symbols);  
}
```

```
int main()  
{  
    char str[100];  
  
    printf("Enter a string: ");  
    fgets(str, sizeof(str), stdin);  
  
    str[strcspn(str, "\n")] = '\0';  
  
    printf("\nString: %s", str);  
    count(str);  
  
    return 0;  
}
```

### **Output:**

**Write input file/string:** He!!o @ W0rld, I am saurabl911

**Screenshot of output:**



### Output

[Clear](#)

```
Enter a string: He!!o @ wOrld, I am saurabl911
Digits: 4
Vowels: 7
Symbols: 4
```

```
=== Code Execution Successful ===
```

### Code : For Text file input

```
#include <stdio.h>
#include <string.h>
void count(char* str, int *vowels, int *digits, int *symbols)
{
    int i;
    char ch;
    for (i = 0; str[i] != '\0'; i++) {
        ch = str[i];
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
            (*vowels)++;
        }
        else if (ch >= '0' && ch <= '9') {
            (*digits)++;
        }
        else if (!((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || ch == ' ' || ch ==
'\n')) {
            (*symbols)++;
        }
    }
}
int main()
{
```

```
char str[1000];  
FILE *file;  
int vowels = 0, digits = 0, symbols = 0;  
  
file = fopen("counter.txt", "r");  
  
while (fgets(str, sizeof(str), file) != NULL) {  
    count(str, &vowels, &digits, &symbols);  
}  
fclose(file);  
printf("\nTotal Vowels: %d", vowels);  
printf("\nTotal Digits: %d", digits);  
printf("\nTotal Symbols: %d", symbols);  
return 0;  
}
```

### **Output:**

#### **Write input file/string:**

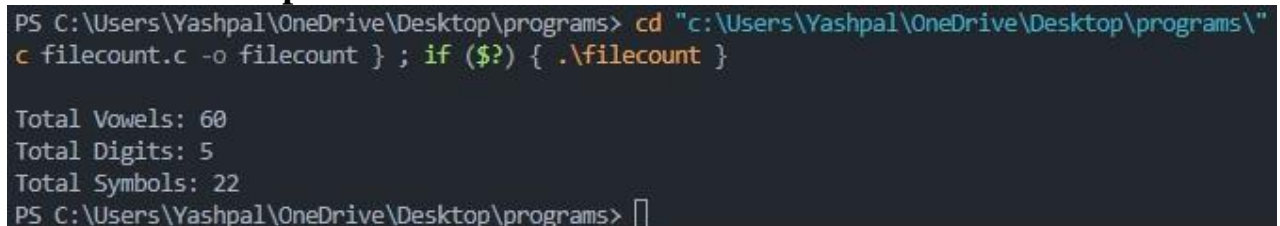
My n@me is X!zY@546

@parul!University!

The compiler is software !!

that converts a program written in a high-level language (Source Language)  
to a low-level language (Object/Target/Machine Language/0, 1's).

#### **Screenshot of output:**



```
PS C:\Users\Yashpal\OneDrive\Desktop\programs> cd "c:\Users\Yashpal\OneDrive\Desktop\programs\  
c filecount.c -o filecount } ; if ($?) { .\filecount }  
  
Total Vowels: 60  
Total Digits: 5  
Total Symbols: 22  
PS C:\Users\Yashpal\OneDrive\Desktop\programs> █
```

## **PRACTICAL NO: 3**

### **AIM: Program to check validation of User Name and Password in C**

**Theory:** This program ensures that a user's username and password meet predefined validation criteria. It takes input from the user for both fields and verifies them against specific rules such as length, character composition, and uniqueness. For example:

- Username: Must follow specific rules like minimum/maximum length, containing only letters, digits, or underscores.
- Password: Should meet security standards, such as including uppercase letters, lowercase letters, digits, and special symbols.

The program helps enforce secure login practices and highlights any invalid entries for correction.

### **Code:**

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int isValidUsername(char username[]) {

    int len = strlen(username);
    if (len < 5 || len > 15) {
        printf("Username must be between 5 and 15 characters.\n");
        return 0;
    }

    for (int i = 0; i < len; i++) {
        if (!isalnum(username[i])) {
            printf("Username must contain only letters and numbers.\n");
            return 0;
        }
    }
}
```

```
    return 1;
}

int isValidPassword(char password[]) {
    int len = strlen(password);

    if (len < 8) {
        printf("Password must be at least 8 characters long.\n");
        return 0;
    }
    int hasUpper = 0, hasLower = 0, hasDigit = 0;
    for (int i = 0; i < len; i++) {
        if (isupper(password[i])) hasUpper = 1;
        if (islower(password[i])) hasLower = 1;
        if (isdigit(password[i])) hasDigit = 1;
    }

    if (!hasUpper || !hasLower || !hasDigit) {
        printf("Password must contain at least one uppercase letter, one lowercase
letter, and one digit.\n");
        return 0;
    }
    return 1;
}

int main() {
    char username[50];
    char password[50];

    printf("Enter Username: ");
    scanf("%s", username);
    printf("Enter Password: ");
    scanf("%s", password);
    int usernameValid = isValidUsername(username);
    int passwordValid = isValidPassword(password);

    if (usernameValid && passwordValid) {
        printf("Username and Password are valid!\n");
    }
}
```

```
} else {  
    printf("try again.\n");  
}  
return 0;  
}
```

### **Output:**

#### **Input string 1:**

Enter Username: vish@123  
Enter Password: abc@1234

#### **Screenshot of output:**

```
PS C:\Users\Yashpal\OneDrive\Desktop\programs> cd "c:\Users\Yashpal\OneDrive\Desktop\programs\" ;  
f ($?) { .\valid }  
Enter Username: vish@123  
Enter Password: abc@1234  
Username must contain only letters and numbers.  
Password must contain at least one uppercase letter, one lowercase letter, and one digit.  
try again.
```

#### **Input string 2:**

Enter Username: Vish1234  
Enter Password: Abc@4793

#### **Screenshot of output:**

```
PS C:\Users\Yashpal\OneDrive\Desktop\programs> cd "c:\Users\Yashpal\OneDrive\Desktop\programs\" ;  
f ($?) { .\valid }  
Enter Username: Vish1234  
Enter Password: Vish1234  
Username and Password are valid!  
PS C:\Users\Yashpal\OneDrive\Desktop\programs> █
```