

LAB REPORT

CSE332: Compiler Design Lab

|  |
| --- |
| 03 |

Topic: Solving Problem Using C.

Submitted To

Shadman Rabby (SHR)

Lecturer

Department of CSE,

Daffodil International University

Submitted By

Student ID: 221-15-5400

Section: 61\_A2

Student Name: M. B. Mahir Tanzim

Date of Assignment Submission: 27 March 2025

|  |  |  |
| --- | --- | --- |
| **Experiment No:** 03 | | **Mapping:** CO1 and CO2 |
| **Experiment Name** | Solving Problem Using C | |

**Experiment Details:**

**Problem 01:** Write a C program that will take multiple lines as input and count the number of Lines.

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

    string line;

    int count = 0;

    while (getline(cin, line)) {

        count++;

    }

    cout << "Total number of lines: " << count << endl;

    return 0;

}

**Problem 02:** Write a C program that will take multiple lines as input and identify the comments if there any.

**Solution**

#include <stdio.h>

#include <string.h>

void detect\_comments(char \*line){

    if (strstr(line, "//") != NULL){

        printf("Single-line comment detected: %s", line);

    }

    else if (strstr(line, "/\*") != NULL){

        printf("Multi-line comment detected: %s", line);

    }

}

int main(){

    char line[1000];

    printf("Enter the C code (Press Ctrl+Z to stop input):\n");

    while (fgets(line, sizeof(line), stdin) != NULL){

        detect\_comments(line);

    }

    return 0;

}

**Problem 03:** Write a C program that will take multiple lines as input and remove the single line/multiple line comments if there any.

**Solution:**

#include <stdio.h>

#include <string.h>

void remove\_comments(char \*code) {

    int i = 0, j = 0;

    int in\_string = 0, in\_char = 0;

    int in\_single\_comment = 0, in\_multi\_comment = 0;

    char result[1000];

    while (code[i]) {

        if (in\_single\_comment && code[i] == '\n') {

            in\_single\_comment = 0;

            result[j++] = code[i];

        }

        else if (in\_multi\_comment && code[i] == '\*' && code[i + 1] == '/') {

            in\_multi\_comment = 0;

            i++;

        }

        else if (in\_single\_comment || in\_multi\_comment) {

            // Skip characters inside comments

        }

         else if (code[i] == '/' && code[i + 1] == '/') {

            in\_single\_comment = 1;

        }

         else if (code[i] == '/' && code[i + 1] == '\*') {

            in\_multi\_comment = 1;

            i++;

        }

        else {

            result[j++] = code[i];

        }

        i++;

    }

    result[j] = '\0';

    printf("Code without comments:\n%s", result);

}

int main() {

    char code[1000], line[200];

    printf("Enter the C code (Press Ctrl+D to stop input on Linux/Mac, Ctrl+Z on Windows):\n");

    code[0] = '\0';

    while (fgets(line, sizeof(line), stdin) != NULL) {

        strcat(code, line);

    }

    remove\_comments(code);

    return 0;

}

**Obtained Output:**

After taking the following demo code as input in every Solution, we obtain the desired output:

#include <iostream>  // This is a single-line comment

using namespace std;

int main() {

    cout << "Hello, World!" << endl;  // Print message

    /\*

       This is a multi-line comment.

       It spans multiple lines.

    \*/

    int x = 10; // Variable declaration

    int y = 20; /\* Another variable \*/

    /\* Multi-line comment

       int z = x + y;

    \*/

    cout << "Sum: " << (x + y) << endl; // Output sum

    return 0;

}

|  |  |
| --- | --- |
| Problem 01:    Problem 02:    Problem 03: | Desired Output? |
| YES  YES |

**Conclusion:**

In this lab, we successfully implemented programs to count the number of lines, detect comments, and remove comments from multiple lines of input. We explored different methods for each task and analyzed their advantages and limitations. The character-by-character approach proved to be efficient in memory usage, while line-by-line processing using fgets() provided better readability and ease of implementation.

For comment detection and removal, maintaining flags helped in accurately identifying single-line (//) and multi-line (/\* ... \*/) comments. By applying structured logic, we ensured that comments were correctly removed while preserving the rest of the input.

Overall, this lab helped in understanding input processing techniques, string manipulation, and efficient handling of text-based data in C.