C PROGRAMMING PROJECT REPORT

UPES

- **Project Title:** To-Do-List
- **Course:** Programming in C (B.Tech 1st Semester)
- **Course Code:** CSEG1041_5
- **Submitted By:** Shrishti Kumari
- **SAP ID:** 590024983
- **Submitted To:** Mohsin F. Dar
- **Date of Submission:** 02/12/20

Problem Statement:

Managing daily tasks manually can be inefficient and easy to forget. People often need a simple way to record, view, and organize their to-do items. This project provides a digital solution by creating a console-based task manager that stores tasks in a file for persistence, allows easy addition, display, and clearing of tasks, and demonstrates key C programming concepts like file handling, structures, loops, and functions.

Objective of the Project:

- Create a simple console-based task manager.
- Allow users to add new tasks.
- Enable users to view all tasks in a clear format.
- Provide an option to mark all tasks as completed (clear file).
- Demonstrate the use of C programming concepts: Structures, File handling, Functions, Loops and conditionals
- Make a menu-driven, user-friendly application.

Software / Tools Used:

Compiler: GCC compiler, VS Code

Operating System: Windows

Algorithm:

main.c –

1. Start the program.
2. Display the Task Manager menu with 4 options.
3. Ask the user to enter a choice.
4. Read the user's choice.
5. If the choice is:
    a. If choice = 1 → Call addTask()
    b. If choice = 2 → Call displayTasks()
    c. f choice = 3 → Call resetTasks()
    d. If choice = 4 → Exit program
6. Else → Show invalid input
7. Repeat until exit
8. End the program.

task.c –

1. Define a structure Task that stores one task title.

2. Create a function addTask()
    a. Open the file to-do-list.txt in append mode.
    b. Accept task title from the user.
    c. Write the task into the file.
    d. Close the file.
3. Create a function displayTasks()
    a. Open the file to-do-list.txt in read mode.
    b. If file doesn't exist, show "No tasks found".
    c. Read each task and display them with numbering.
    d. Close the file.
4. Create a function resetTasks()
    a. Open the file in write mode to clear all tasks.
    b. Close the file.
    c. Display message for successful clearing.
5. End program.

Pseudocode:

main.c –

```
START
LOOP forever
```

```
    DISPLAY "TASK MANAGER MENU"
    DISPLAY options 1 to 4
    READ choice
    IF choice = 1 THEN
        CALL addTask()
    ELSE IF choice = 2 THEN
        CALL displayTasks()
    ELSE IF choice = 3 THEN
        CALL resetTasks()
    ELSE IF choice = 4 THEN
        DISPLAY "Closing Task Manager"
        BREAK the loop
    ELSE
        DISPLAY "Invalid input"
    ENDIF
ENDLOOP
END
```

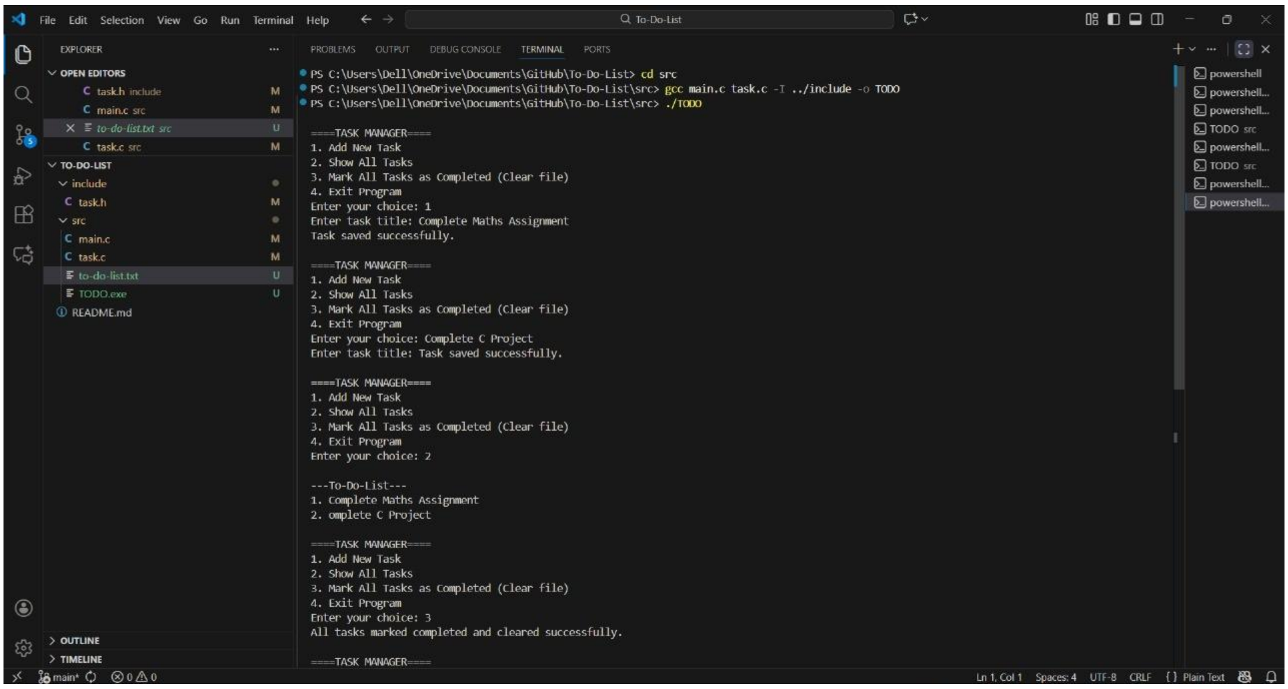
task.c –

```
DEFINE structure Task with title[100]
```

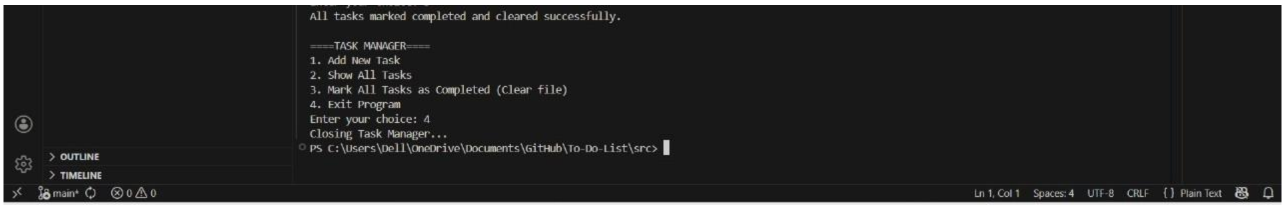
```
FUNCTION addTask:
   OPEN file "to-do-list.txt" in append mode
   IF file not opened THEN
       DISPLAY "Error opening file"
       RETURN
   ENDIF
   READ task title from user
   WRITE the structure into file
   CLOSE file
   DISPLAY "Task saved successfully"
END FUNCTION
FUNCTION displayTasks:
   OPEN file "to-do-list.txt" in read mode
   IF file not opened THEN
       DISPLAY "No tasks found"
       RETURN
   ENDIF
   SET count = 0
   WHILE reading structure successful
```

```
        INCREMENT count
        DISPLAY count and task title
    ENDWHILE
    IF count = 0 THEN
        DISPLAY "No tasks added yet"
    ENDIF
    CLOSE file
END FUNCTION
FUNCTION resetTasks:
    OPEN file "to-do-list.txt" in write mode
(clears file)
    IF file not opened THEN
        DISPLAY "Error clearing tasks"
        RETURN
    ENDIF
    CLOSE file
    DISPLAY "All tasks cleared"
END FUNCTION
```

Output Screenshots:



```
PS C:\Users\Dell\OneDrive\Documents\GitHub\To-Do-List> cd src
PS C:\Users\Dell\OneDrive\Documents\GitHub\To-Do-List\src> gcc main.c task.c -I ../include -o TODO
PS C:\Users\Dell\OneDrive\Documents\GitHub\To-Do-List\src> ./TODO

====TASK MANAGER====
1. Add New Task
2. Show All Tasks
3. Mark All Tasks as Completed (Clear file)
4. Exit Program
Enter your choice: 1
Enter task title: Complete Maths Assignment
Task saved successfully.

====TASK MANAGER====
1. Add New Task
2. Show All Tasks
3. Mark All Tasks as Completed (Clear file)
4. Exit Program
Enter your choice: Complete C Project
Enter task title: Task saved successfully.

====TASK MANAGER====
1. Add New Task
2. Show All Tasks
3. Mark All Tasks as Completed (Clear file)
4. Exit Program
Enter your choice: 2

---To-Do-List---
1. Complete Maths Assignment
2. omplete C Project

====TASK MANAGER====
1. Add New Task
2. Show All Tasks
3. Mark All Tasks as Completed (Clear file)
4. Exit Program
Enter your choice: 3
All tasks marked completed and cleared successfully.

====TASK MANAGER====
```



```
All tasks marked completed and cleared successfully.

====TASK MANAGER====
1. Add New Task
2. Show All Tasks
3. Mark All Tasks as Completed (Clear file)
4. Exit Program
Enter your choice: 4
Closing Task Manager...
PS C:\Users\Dell\OneDrive\Documents\GitHub\To-Do-List\src>
```

Source Code:

main.c –

```
#include <stdio.h>

#include <stdlib.h>

#include "task.h"

int main()

{
    int choice;
    while(1)
    {
        printf("\n====TASK MANAGER====\n");
        printf("1. Add New Task\n");
        printf("2. Show All Tasks\n");
        printf("3. Mark All Tasks as Completed (Clear
file)\n");
        printf("4. Exit Program\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if(choice==1)
        addTask();
        else if(choice==2)
        displayTasks();
        else if(choice==3)
        resetTasks();
        else if(choice==4)
        {
            printf("Closing Task Manager...\n");
            break;
        }
        else
        {
            printf("Invalid Input. Please try again.\n");
        }
    }
    return 0;
    }
```

task.c –

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "task.h"
struct Task {
    char title[100];
};
void addTask()
{
    FILE *fp = fopen("to-do-list.txt", "a");
    if (fp == NULL)
    {
        printf("Error opening file!\n");
        return;
    }
    char task[100];
    printf("Enter task title: ");
    getchar();
    fgets(task, 100, stdin);
    fprintf(fp, "%s", task);
    fclose(fp);
    printf("Task saved successfully.\n");
}
void displayTasks()
{
    FILE *fp = fopen("to-do-list.txt", "r");
    if (fp == NULL)
    {
        printf("No task found\n");
        return;
    }
    char task[100];
    int count = 0;
    printf("\n---To-Do-List---\n");
    while(fgets(task, sizeof(task), fp))
    {
        count++;
        printf("%d. %s", count, task);
    }
    if(count == 0)
    printf("No tasks added yet.\n");
    fclose(fp);
}
void resetTasks()
{
    FILE *fp = fopen("to-do-list.txt", "w");
    if (fp == NULL)
    {
        printf("Error clearing tasks!\n");
        return;
    }
    fclose(fp);
    printf("All tasks marked completed and cleared successfully.\n");
}
```

task.h –

```
#ifndef TASK_H
#define TASK_H
void addTask();
void displayTasks();
void resetTasks();
#endif
```

Conclusion:

- The project successfully implements a menu-driven To-Do List Manager using C programming concepts.

- Users can add, view, and clear tasks with file handling.
- Demonstrates understanding of functions, loops, structures, and file operations in C.

Future Enhancements:

- Add task deadlines and priorities.
- Allow editing or deleting individual tasks instead of clearing all.