Day 24 Documentation

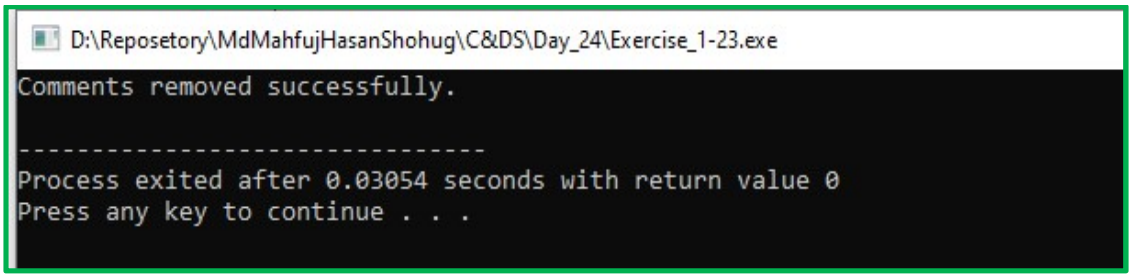
Md. Mahfuj Hasan Shohug

BDCOM0019

1. **Exercise 1-23:** Write a program to remove all comments from a C program. Don't forget to handle quoted strings and character constants properly. C comments don't nest.

   Answer: For this problem in my program that modifies a source file by removing comments and then saves the updated version without comments. It includes a function called remove_comments that accepts two file pointers: input_file for the original file with comments and output_file for the finished file without comments. Each line in the input file is iterated over by the function, which then parses the contents and writes the uncommented portions to the output file. It handles a variety of situations while retaining uncommented code, including strings and single-line and multi-line comments.
   In this program each line in the input file has comments removed using the remove_comments function. To keep tracking of whether the current character is inside a string or a comment, it use the two flags inside_string and inside_comment. Each character in the line is iterated over by the function, which checks for various circumstances and writes the uncommented characters to the output file.
   Additionally, it detects single-line comments that begin with // and skips the following line when it comes across this comment indicator. The removal of comments spanning multiple lines is possible thanks to the handling of multi-line comments beginning with /* and ending with */.
   Analysis:
   - **Benefit:** The code effectively removes comments from a source file while maintaining the original code structure. It covers a variety of situations, including strings and various comment types. The proper management of files is ensured by the use of file operations (fopen and flows).
   - **Limitation:** The code takes for granted that the /* and // comment indicators are only ever used for comments and never in strings or other contexts. When such an instance occurs, it can wrongly erase some of the code. Additionally, if the file opening fails, the code doesn't handle any input validation issues or errors.

Here is some test case for my program:



Input file

Exercise_1-23.c | test_case_1.c | test_case_1_nocomment.c

```c
/*test case 1*/
#include <stdio.h>

/**************************************************
 * Function Name: main
 * Description: test case for my program
 * Parameters:
 *    - argc: Number of command-line arguments
 *    - argv: Array of command-line arguments
 * Returns: 0 on successful execution
 **************************************************/
int main(int argc, char *argv[])
{
    printf("Hello world!"); //print hello world
}
```

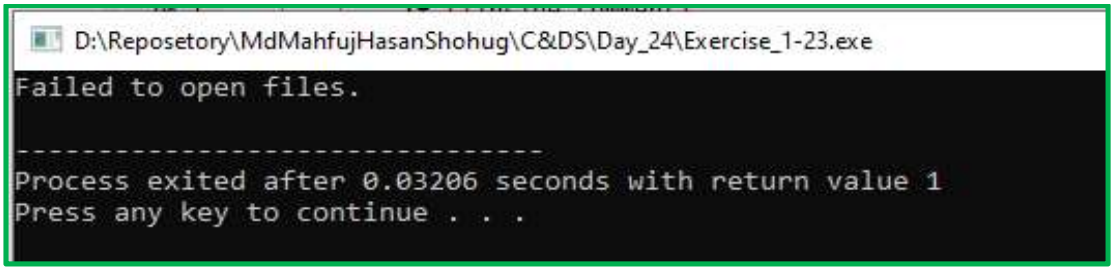
Output file

Exercise_1-23.c | test_case_1.c | test_case_1_nocomment.c

```c

#include <stdio.h>


int main(int argc, char *argv[])
{
    printf("Hello world!");
}

```

On the file of my test case I handle a big and multiple uses comment which one I provide on the folder Day_24. If run my code then the test_case_1_nocomment.c file will be updated and you can view the changes.

If file not found:



Source Code:

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

#define MAX_LINE_LENGTH 1000
/**************************************************
 * Function Name: remove_comments
 * Description: Removes comments from the input file and writes the result
to the output file.
 * Parameters:
 *    - input_file: Pointer to the input file
 *    - output_file: Pointer to the output file
 * Returns: None
 **************************************************/
void remove_comments(FILE *input_file, FILE *output_file)
{
    char line[MAX_LINE_LENGTH];
    int inside_string = 0;
    int inside_comment = 0;

    while (fgets(line, MAX_LINE_LENGTH, input_file) != NULL)
    {
        size_t line_length = strlen(line);
        size_t i = 0;

        while (i < line_length)
        {
            // Handle different cases inside the line
            // Check if not inside a comment or string
            if (!inside_comment && !inside_string)
            {
                if (line[i] == '"')
                {
                    // Check for the start of a string
                    inside_string = 1;
                    fputc(line[i], output_file);
                    i++;
                    continue;
                }

                if (line[i] == '/' && i + 1 < line_length)
                {
                    // Check for the start of a comment
```

```
                    if (line[i + 1] == '/')
                    {
                    // Skip the rest of the line, excluding the final '/'
                        while (i < line_length && line[i] != '\n')
                        {
                            i++;
                        }
                        fputc(line[i], output_file);
                        break;
                    } else if (line[i + 1] == '*')
                    {
                        // Multi-line comment
                        inside_comment = 1;
                        i++; // Skip the '*'
                        continue;
                    }
                }
            }

            if (inside_comment && line[i] == '*'
                && i + 1 < line_length && line[i + 1] == '/')
            {
                // Check for the end of a multi-line comment
                inside_comment = 0;
                i += 2; // Skip the '*/'
                continue;
            }

            if (!inside_comment)
            {
                // Check if not inside a comment
                fputc(line[i], output_file);
            }

            if (inside_string && line[i] == '"')
            {
                // Check for the end of a string
                inside_string = 0;
            }

            i++;
        }
    }
}
```

```c
/*****************************************************
 * Function Name: main
 * Description: The entry point of the program.
 * Returns:
 *    - EXIT_SUCCESS: If the program executes successfully
 *****************************************************/
int main(int argc, char *argv[])
{
    // Read with comment file
    FILE *input_file = fopen("test_case_1.c", "r");
    // Updated file create and write without comment
    FILE *output_file = fopen("test_case_1_nocomment.c", "w");
    if (input_file == NULL || output_file == NULL)
    {
        printf("Failed to open files.\n");
        return 1;
    }

    remove_comments(input_file, output_file);

    fclose(input_file);
    fclose(output_file);

    printf("Comments removed successfully.\n");

    return 0;
}
```