Day 22 Documentation

Md. Mahfuj Hasan Shohug

BDCOM0019

1. **Exercise 7-1:** Write a program that converts upper case to lower or lower case to upper, depending on the name it is invoked with, as found in argv[0].

   Answer: **Functions for Converting Strings:** Two functions, convert_lower and convert_upper, are defined in the code. They change a string's case to lowercase or uppercase, respectively. These routines transform the characters by iterating over each character in the string using the tolower and toupper functions from the ctype.h library.
   User Input and Arguments on the Command Line: The program asks the user to enter a string. Until it comes across the end of the file (EOF), a newline character, or the maximum length of the string (MAX_LENGTH), it reads the input string character by character. The str array contains the input string. The program then determines whether to change the string to lowercase or uppercase by examining the command-line option (argv[0]).
   Output and Error Handling: The program prints the converted string after converting the string. An error message is displayed if the command-line option is neither "convert_lower" nor "convert_upper," indicating that the command-line argument was incorrect.

   Test Case on the next page:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>gcc Exercise_7_1.c -o upper

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>upper
Please enter a string: mahfuj
Converted string is: MAHFUJ

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>gcc Exercise_7_1.c -o lower

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>lower
Please enter a string: MAHFUJ
Converted string is: mahfuj

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>lower
Please enter a string: ABC
Converted string is: abc

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>upper
Please enter a string: abc
Converted string is: ABC

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>upper
Please enter a string: ABcd
Converted string is: ABCD

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>lower
Please enter a string: abCD
Converted string is: abcd

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>lower
Please enter a string: 1234
Converted string is: 1234

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>upper
Please enter a string: 1234
Converted string is: 1234

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>upper
Please enter a string:
Converted string is:

D:\Repository\MdMahfujHasanShohug\C&DS\Day_22>lower
Please enter a string:
Converted string is:
```

Source code:

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

#define MAX_LENGTH 500

/*****************************************************
 * Function Name: convert_lower
 * Description: Converts a string to lowercase and prints it.
 * Parameters:
 *   - str: Pointer to the string to be converted
 *****************************************************/
void convert_lower(const char *str)
{
   int i;
            printf("Converted string is: ");
   for (i = 0; str[i] != '\0'; i++)
   {
      putchar(tolower(str[i]));
   }
   printf("\n");
}

/*****************************************************
 * Function Name: convert_upper
 * Description: Converts a string to uppercase and prints it.
 * Parameters:
 *   - str: Pointer to the string to be converted
 *****************************************************/
void convert_upper(const char *str)
{
   int i;
            printf("Converted string is: ");
   for (i = 0; str[i] != '\0'; i++)
   {
      putchar(toupper(str[i]));
   }
   printf("\n");
}

/*****************************************************
 * Function Name: main
 * Description: Entry point of the program.
 * Parameters:
 *   - argc: Number of command-line arguments
 *   - argv: Array of command-line argument strings
 * Returns: Integer indicating the exit status of the program
 *****************************************************/
int main(int argc, char *argv[])
{
   char str[MAX_LENGTH + 1];
```

```
        char c;
        int i = 0;

        printf("Please enter a string: ");

        while ((c = getchar()) != EOF && c != '\n' && i < MAX_LENGTH)
        {
          str[i] = c;
          i++;
        }
        str[i] = '\0';

        if (strcmp(argv[0], "lower") == 0)
        {
          convert_lower(str);
        } else if (strcmp(argv[0], "upper") == 0)
        {
          convert_upper(str);
        } else
        {
          printf("Wrong argument passed.\n");
        }

        return 0;
}
```

2. **Exercise 7-6:** Write a program to compare two files, printing the first line where they different.

   Answer: The code defines a function named compareFiles that accepts two file pointers (file1 and file2) as inputs. Using the fgets function, the code takes each line from both files and compares them using the strcmp function. The line number is returned if a discrepancy is discovered. The function returns -2 if one file ends before the other or if the lengths of the two files differ. The function returns -1, indicating that the files are identical, if there are no differences.
   The program expects two command-line parameters giving the paths of the two files to be compared (file1> and file2>). It checks to see if the right number of parameters is given before using the fopen method to try and open both files. An error notice is shown and the program exits with a code of 2 if any file cannot be opened.
   The program confirms the return result of the compareFiles function after comparing the files. It prints a message confirming that if the files are identical. It displays a notification regarding the varied file lengths if they are different. If not, it displays the line number where the files different.

Test case on this code:

```
D:\Repository\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_6.exe
Difference found at line 1:
W
H
The files are different.

--------------------------------
Process exited after 0.0311 seconds with return value 0
Press any key to continue . . .
```

```
D:\Repository\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_6.exe
Difference found at line 2:
e
E
The files are different.

--------------------------------
Process exited after 0.0319 seconds with return value 0
Press any key to continue . . .
```

```
Select D:\Repository\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_6.exe
Difference found at line 5:
1OThe files are different.

--------------------------------
Process exited after 0.03219 seconds with return value 0
Press any key to continue . . .
```

```
D:\Repository\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_6.exe
Difference found at line 1:
H
The files are different.

--------------------------------
Process exited after 0.03208 seconds with return value 0
Press any key to continue . . .
```

```
D:\Repository\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_6.exe
The files are Same.

--------------------------------
Process exited after 0.03245 seconds with return value 0
Press any key to continue . . .
```

Source Code:

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

#define MAX_LINE_LENGTH 100

/*****************************************************
 * Function Name: compare_files
 * Description: Compares two files line by line and returns the line number where they differ.
 * Parameters:
 *   - file1: Pointer to the first file
 *   - file2: Pointer to the second file
 * Returns:
 *   - Line number where the files differ
 *   - -1 if the files are the same
 *   - -2 if the files have different lengths
 *****************************************************/
int compareFiles(FILE *file1, FILE *file2);

/*****************************************************
 * Function Name: main
 * Description: Entry point of the program.
 * Parameters:
 *   - argc: Number of command-line arguments
 *   - argv: Array of command-line argument strings
 * Returns:
 *   - 0 if the program executed successfully
 *   - 1 if incorrect command-line arguments are provided
 *   - 2 if there are issues opening the files
 *****************************************************/
int main(int argc, char *argv[])
{
   if (argc != 3)
   {
      fprintf(stderr, "Usage: %s <file1> <file2>\n", argv[0]);
      return 1;
   }

   FILE *file1 = fopen(argv[1], "r");
   if (file1 == NULL)
   {
      fprintf(stderr, "Cannot open %s\n", argv[1]);
      return 2;
   }

   FILE *file2 = fopen(argv[2], "r");
   if (file2 == NULL)
   {
      fprintf(stderr, "Cannot open %s\n", argv[2]);
      fclose(file1);
      return 2;
```

```
    }

    int lineNum = compare_files(file1, file2);
    if (lineNum == -1)
    {
        printf("Files are Same.\n");
    }
    else if (lineNum == -2)
    {
        printf("Files have different lengths.\n");
    }
    else
    {
        printf("Files diffrent at line %d\n", lineNum);
    }

    fclose(file1);
    fclose(file2);

    return 0;
}

int compare_files(FILE *file1, FILE *file2)
{
    char line1[MAX_LINE_LENGTH];
    char line2[MAX_LINE_LENGTH];
    int lineNum = 1;

    while (fgets(line1, sizeof(line1), file1) != NULL && fgets(line2, sizeof(line2), file2) != NULL)
    {
        if (strcmp(line1, line2) != 0)
        {
            return lineNum;
        }
        lineNum++;
    }

    if (!feof(file1) || !feof(file2))
    {
        return -2; // Files have different lengths
    }

    return -1; // Files are same
}
```

3. **Exercise 7-9:** Functions like isupper can be implemented to save space or to save time. Explore both possibilities.

Answer: In my approach, three distinct techniques are used in this code to determine whether a character is uppercase:
isupper_lookup: Like the earlier method, this one makes use of a lookup table.
isupper_strchr: This method looks for the character in a string of uppercase letters using the strchr() function.
isupper_custom: By directly comparing the character with the ASCII range of uppercase letters, this method implements a custom check.
For each character entered, all three methods are invoked within **the main() function**, and the resulting output is produced.

Test Case:

```
D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_22\Exercise_7_9.exe
A
Lookup: true
Strchr: true
Custom: true

a
Lookup: false
Strchr: false
Custom: false

1
Lookup: false
Strchr: false
Custom: false

!
Lookup: false
Strchr: false
Custom: false

Aa
Lookup: true
Strchr: true
Custom: true
Lookup: false
Strchr: false
Custom: false
```

```
Hello BDCOM
Lookup: true
Strchr: true
Custom: true
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: true
Strchr: true
Custom: true
Lookup: true
Strchr: true
Custom: true
Lookup: true
Strchr: true
Custom: true
Lookup: true
Strchr: true
Custom: true
Lookup: true
Strchr: true
Custom: true

@12A
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: false
Strchr: false
Custom: false
Lookup: true
Strchr: true
Custom: true
```

Source Code:

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

// Function Declarations

/*****************************************************
 * Function Name: isupper_lookup
 * Description: Checks if a character is uppercase using a lookup table.
 * Parameters:
 *   - c: The character to check
 * Returns:
 *   - true if the character is uppercase
 *   - false otherwise
 *****************************************************/
bool isupper_lookup(char c);

/*****************************************************
 * Function Name: isupper_strchr
 * Description: Checks if a character is uppercase using the strchr function.
 * Parameters:
 *   - c: The character to check
 * Returns:
 *   - true if the character is uppercase
 *   - false otherwise
 *****************************************************/
bool isupper_strchr(char c);

/*****************************************************
 * Function Name: isupper_custom
 * Description: Checks if a character is uppercase using a custom implementation.
 * Parameters:
 *   - c: The character to check
 * Returns:
 *   - true if the character is uppercase
 *   - false otherwise
 *****************************************************/
bool isupper_custom(char c);

/*****************************************************
 * Function Name: main
 * Description: Reads characters from input until 'x' is entered and
 *          checks if each character is uppercase using different methods.
 * Returns:
 *   - EXIT_SUCCESS: If the program runs successfully
 *****************************************************/
int main(void)
{
   int c;
```

```
    while ((c = getchar()) != 'x')
        {
      if (c == '\n')
        continue;

      printf("Lookup: %s\n", isupper_lookup(c) ? "true" : "false");
      printf("Strchr: %s\n", isupper_strchr(c) ? "true" : "false");
      printf("Custom: %s\n", isupper_custom(c) ? "true" : "false");
    }

    return EXIT_SUCCESS;
  }

  // Method 1: Lookup Table
  bool isupper_lookup(char c) {
    static const bool lookup_table[256] = {
      ['A'] = true, ['B'] = true, ['C'] = true, ['D'] = true, ['E'] = true,
      ['F'] = true, ['G'] = true, ['H'] = true, ['I'] = true, ['J'] = true,
      ['K'] = true, ['L'] = true, ['M'] = true, ['N'] = true, ['O'] = true,
      ['P'] = true, ['Q'] = true, ['R'] = true, ['S'] = true, ['T'] = true,
      ['U'] = true, ['V'] = true, ['W'] = true, ['X'] = true, ['Y'] = true,
      ['Z'] = true
    };
    return lookup_table[(unsigned char)c];
  }

  // Method 2: Strchr Function
  bool isupper_strchr(char c)
  {
    return (strchr("ABCDEFGHIJKLMNOPQRSTUVWXYZ", c) != NULL);
  }

  // Method 3: Custom Implementation
  bool isupper_custom(char c)
  {
    if (c >= 'A' && c <= 'Z')
      return true;
    else
      return false;
}
```