```
#include <stdio.h>
 2
 3 #define MAX_STRING_LENGTH 512
4
 5 int main(void)
 6 {
7
        //function prototype
8
        void MyStrrev(char *, char *);
 9
        int MyStrlen(char *);
10
11
       //variable declarations
        char *chArray_Original = NULL, *chArray_Reversed = NULL; // A Character
12
          Array Is A String
        int original_string_length;
13
14
15
       //code
16
       // *** STRING INPUT ***
17
        printf("\n\n");
18
        chArray Original = (char *)malloc(MAX STRING LENGTH * sizeof(char));
19
20
        if (chArray_Original == NULL)
21
            printf("MEMORY ALLOCATION FOR ORIGINAL STRING FAILED !!! EXITTING
22
              NOW ... \langle n \rangle;
23
            exit(0);
24
        }
25
26
        printf("Enter A String : \n\n");
27
        gets_s(chArray_Original, MAX_STRING_LENGTH);
28
        // *** STRING REVERSE ***
29
30
        original string length = MyStrlen(chArray Original);
        chArray_Reversed = (char *)malloc(original_string_length * sizeof(char));
31
32
        if (chArray_Reversed == NULL)
33
34
            printf("MEMORY ALLOCATION FOR REVERSED STRING FAILED !!! EXITTING
              NOW ...\n\n");
            exit(0);
35
        }
36
37
38
       MyStrrev(chArray_Reversed, chArray_Original);
39
40
        // *** STRING OUTPUT ***
41
        printf("\n\n");
        printf("The Original String Entered By You (i.e : 'chArray_Original[]')
42
          Is : \n\n");
        printf("%s\n", chArray_Original);
43
44
        printf("\n\n");
45
46
        printf("The Reversed String (i.e : 'chArray_Reversed[]') Is : \n\n");
47
        printf("%s\n", chArray_Reversed);
48
49
        if (chArray Reversed)
50
51
            free(chArray_Reversed);
            chArray_Reversed = NULL;
52
```

```
... \verb|er|05-StringOperations|03-StringReverse| StringReverse.c|
```

```
2
```

```
53
             printf("\n\n");
 54
             printf("MEMORY ALLOCATED TO REVERSED STRING HAS BEEN SUCCESSFULLY
               FREED !!!\n\n");
 55
         }
 56
 57
         if (chArray_Original)
 58
 59
             free(chArray_Original);
 60
             chArray_Original = NULL;
             printf("\n\n");
 61
             printf("MEMORY ALLOCATED TO ORIGINAL STRING HAS BEEN SUCCESSFULLY
 62
               FREED !!!\n\n");
 63
         }
 64
         return(0);
 65
 66 }
 67
    void MyStrrev(char *str_destination, char *str_source)
 68
 69
 70
         //function prototype
 71
         int MyStrlen(char *);
 72
         //variable declarations
 73
 74
         int iStringLength = 0;
 75
         int i, j, len;
 76
 77
         //code
 78
         iStringLength = MyStrlen(str_source);
 79
         // ARRAY INDICES BEGIN FROM 0, HENCE, LAST INDEX WILL ALWAYS BE (LENGTH - 🤝
 80
          1)
 81
         len = iStringLength - 1;
 82
         // WE NEED TO PUT THE CHARACTER WHICH IS AT LAST INDEX OF 'str source' TO >
 83
           THE FIRST INDEX OF 'str destination'
 84
         // AND SECOND-LAST CHARACTER OF 'str_source' TO THE SECOND CHARACTER OF
           'str_destination' and so on...
         for (i = 0, j = len; i < iStringLength, j >= 0; i++, j--)
 85
 86
             *(str destination + i) = *(str source + j);
 87
 88
 89
 90
         *(str destination + i) = '\0';
 91 }
 92
 93 int MyStrlen(char *str)
 94 {
 95
         //variable declarations
 96
         int j;
 97
         int string length = 0;
 98
 99
         //code
         // *** DETERMINING EXACT LENGTH OF THE STRING, BY DETECTING THE FIRST
100
           OCCURENCE OF NULL-TERMINATING CHARACTER ( \0 ) ***
         for (j = 0; j < MAX_STRING_LENGTH; j++)</pre>
101
102
```

```
...er\05-StringOperations\03-StringReverse\StringReverse.c

103     if (str[j] == '\0')
              if (str[j] == '\0')
104
                   break;
105
              else
106
                   string_length++;
107
108
          return(string_length);
109 }
110
```

3