```
#include <stdio.h>
 2
 3 #define MAX_STRING_LENGTH 512
4
 5 int main(void)
 6 {
7
        //function prototype
8
        void MyStrcat(char *, char *);
        int MyStrlen(char *);
 9
10
       //variable declarations
11
        char *chArray_One = NULL, *chArray_Two = NULL; // A Character Array Is A →
12
         String
13
14
       //code
15
       // *** STRING INPUT ***
16
        printf("\n\n");
17
       chArray One = (char *)malloc(MAX_STRING_LENGTH * sizeof(char));
18
19
        if (chArray_One == NULL)
20
        {
21
           printf("MEMORY ALLOCATION TO FIRST STRING FAILED !!! EXITTING NOW...\n →
              \n");
22
           exit(0);
        }
23
24
25
        printf("Enter First String : \n\n");
26
        gets_s(chArray_One, MAX_STRING_LENGTH);
27
        printf("\n\n");
28
29
        chArray Two = (char *)malloc(MAX STRING LENGTH * sizeof(char));
30
        if (chArray Two == NULL)
31
           printf("MEMORY ALLOCATION TO SEOND STRING FAILED !!! EXITTING NOW...\n →
32
              \n");
33
           exit(0);
34
        }
35
        printf("Enter Second String : \n\n");
36
37
        gets_s(chArray_Two, MAX_STRING_LENGTH);
38
        // *** STRING CONCAT ***
39
        printf("\n\n");
40
        printf("****** BEFORE CONCATENATION ******");
41
        printf("\n\n");
42
43
        printf("The Original First String Entered By You (i.e : 'chArray_One[]') 🤝
         Is : \n\n");
44
        printf("%s\n", chArray_One);
45
46
        printf("\n\n");
        printf("The Original Second String Entered By You (i.e : 'chArray_Two[]') >
47
         Is : \n\n");
        printf("%s\n", chArray_Two);
48
49
50
       MyStrcat(chArray_One, chArray_Two);
51
```

```
52
        printf("\n\n");
        printf("***** AFTER CONCATENATION ******");
53
        printf("\n\n");
54
        printf("'chArray_One[]' Is : \n\n");
55
        printf("%s\n", chArray_One);
56
57
58
        printf("\n\n");
        printf("'chArray_Two[]' Is : \n\n");
59
        printf("%s\n", chArray_Two);
60
61
62
        if (chArray_Two)
63
        {
            free(chArray_Two);
64
            chArray_Two = NULL;
65
            printf("\n\n");
66
67
             printf("MEMORY ALLOCATED TO SECOND STRING HAS BEEN SUCCESSFULLY
               FREED !!!\n\n");
68
        }
69
        if (chArray One)
70
71
            free(chArray_One);
72
73
            chArray_One = NULL;
74
            printf("\n\n");
            printf("MEMORY ALLOCATED TO FIRST STRING HAS BEEN SUCCESSFULLY
75
               FREED !!!\n\n");
76
        }
77
        return(0);
78
79 }
80
81 void MyStrcat(char *str destination, char *str source)
82
        //function prototype
83
84
        int MyStrlen(char *);
85
        //variable declarations
86
        int iStringLength_Source = 0, iStringLength_Destination = 0;
87
        int i, j;
88
89
90
        //code
        iStringLength Source = MyStrlen(str source);
91
        iStringLength Destination = MyStrlen(str destination);
92
93
94
        // ARRAY INDICES BEGIN FROM Ø, HENCE, LAST VALID INDEX OF ARRAY WILL
          ALWAYS BE (LENGTH - 1)
        // SO, CONCATENATION MUST BEGIN FROM INDEX NUMBER EQUAL TO LENGTH OF THE
95
          ARRAY 'str_destination'
        // WE NEED TO PUT THE CHARACTER WHICH IS AT FIRST INDEX OF 'str_source' TO >
96
            THE (LAST INDEX + 1) OF 'str destination'
97
        for (i = iStringLength_Destination, j = 0; j < iStringLength_Source; i++, →</pre>
           j++)
98
        {
99
            *(str destination + i) = *(str source + j);
100
        }
101
```

```
\dots Operations \verb|\| 04-StringConcatenation \verb|\| StringConcatenation.c
```

```
102
        *(str_destination + i) = '\0';
103 }
104
105 int MyStrlen(char *str)
106 {
107
        //variable declarations
108
        int j;
109
        int string_length = 0;
110
        //code
111
        // *** DETERMINING EXACT LENGTH OF THE STRING, BY DETECTING THE FIRST
112
          OCCURENCE OF NULL-TERMINATING CHARACTER ( \0 ) ***
113
        for (j = 0; j < MAX_STRING_LENGTH; j++)</pre>
114
115
             if (str[j] == '\0')
116
                 break;
117
             else
118
                 string_length++;
119
120
        return(string_length);
121 }
122
```