C – String

- C Strings are nothing but array of characters ended with null character ('\0').
- This null character indicates the end of the string.
- Strings are always enclosed by double quotes. Whereas, character is enclosed by single quotes in C.

Example for C string:

- **char string[20] = { 'f' , 'r' , 'e' , 's' , 'h' , '2' , 'r' , 'e' , 'f' , 'r' , 'e' , 's' , 'h' , '\0'}; (**
- **char string[20] = "fresh2refresh";**
- **char string []   = "fresh2refresh";**

- Difference between above declarations are, when we declare char as "string[20]", 20 bytes of memory space is allocated for holding the string value.
- When we declare char as "string[]", memory space will be allocated as per the requirement during execution of the program.

Example program for C string:

```
#include <stdio.h>

int main ()
{
   char string[20] = " hello world ";

   printf("The string is : %s \n", string );
   return 0;
}
```

Output:

```
The string is : hello world
```

- **string.h header file supports all the string functions in C language.**

C – strcat() function

- strcat() function in C language concatenates two given strings. It concatenates source string at the end of destination string. Syntax for strcat( ) function is given below.

  char * strcat (char * destination, const char * source);

- Example :

  strcat (str2, str1); - str1 is concatenated at the end of str2.
  strcat (str1, str2); - str2 is concatenated at the end of str1.

- As you know, each string in C is ended up with null character ('\0').
- In strcat( ) operation, null character of destination string is overwritten by source string's first character and null character is added at the end of new destination string which is created after strcat( ) operation.

Example program for strcat( ) function in C:

- In this program, two strings "fresh2refresh" and "C tutorial" are concatenated using strcat( ) function and result is displayed as "C tutorial fresh2refresh".

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

int main( )
{
  char source[ ] = " fresh2refresh" ;
  char target[ ]= " C tutorial" ;

  printf ( "\nSource string = %s", source ) ;
  printf ( "\nTarget string = %s", target ) ;

  strcat ( target, source ) ;

  printf ( "\nTarget string after strcat( ) = %s", target ) ;
}
```

Output:

```
Source string                  = fresh2refresh
Target string                  = C tutorial
Target string after strcat( ) = C tutorial fresh2refresh
```

C – strcpy() function

- strcpy( ) function copies contents of one string into another string. Syntax for strcpy function is given below.

  char * strcpy (char * destination, const char * source);

- Example:

  strcpy (str1, str2) – It copies contents of str2 into str1.
  strcpy (str2, str1) – It copies contents of str1 into str2.

- If destination string length is less than source string, entire source string value won't be copied into destination string.
- For example, consider destination string length is 20 and source string length is 30. Then, only 20 characters from source string will be copied into destination string and remaining 10 characters won't be copied and will be truncated.

Example program for strcpy( ) function in C:

  - In this program, source string "fresh2refresh" is copied into target string using strcpy( ) function.

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

int main( )
{
  char source[ ] = "fresh2refresh" ;
  char target[20]= "" ;
  printf ( "\nsource string = %s", source ) ;
  printf ( "\ntarget string = %s", target ) ;
  strcpy ( target, source ) ;
  printf ( "\ntarget string after strcpy( ) = %s", target ) ;
  return 0;
}
```

Output:

```
source string = fresh2refresh
target string =
target string after strcpy( ) = fresh2refresh
```

C – strlen() function

- strlen( ) function in C gives the length of the given string. Syntax for strlen( ) function is given below.

size_t strlen (const char * str);

- strlen( ) function counts the number of characters in a given string and returns the integer value.
- It stops counting the character when null character is found. Because, null character indicates the end of the string in C.

Example program for strlen() function in C:

- In below example program, length of the string "fresh2refres.com" is determined by strlen() function as below. Length of this string 17 is displayed as output.

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

int main( )
{
   int len;
   char array[20]="fresh2refresh.com" ;

   len = strlen(array) ;

   printf ( "\string length  = %d \n" , len ) ;
   return 0;
}
```

Output:

```
string length = 17
```

C – strcmp() function

- strcmp( ) function in C compares two given strings and returns zero if they are same.
- If length of **string1 < string2**, it returns **< 0** value.
- If length of **string1 > string2**, it returns **> 0** value.

int strcmp (const char * str1, const char * str2);

- **strcmp( ) function is case sensitive**. i.e, "A" and "a" are treated as different characters.

Example program for strcmp( ) function in C:

- In this program, strings "fresh" and "refresh" are compared. 0 is returned when strings are equal. Negative value is returned when str1 < str2 and positive value is returned when str1 > str2.

```
#include <stdio.h>
#include <string.h>
int main( )
{
  char str1[ ] = "fresh" ;
  char str2[ ] = "refresh" ;
  int i, j, k ;
  i = strcmp ( str1, "fresh" ) ;
  j = strcmp ( str1, str2 ) ;
  k = strcmp ( str1, "f" ) ;
  printf ( "\n%d %d %d", i, j, k ) ;
  return 0;
}
```

Output:

```
0 -1 1
```

C – strchr() function

- strchr( ) function returns pointer to the first occurrence of the character in a given string.

char *strchr(const char *str, int character);

Example program for strchr() function in C:

In this program, strchr( ) function is used to locate first occurrence of the character 'i' in the string "This is a string for testing". Character 'i' is located at position 3 and pointer is returned at first occurrence of the character 'i'.

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  p = strchr (string,'i');

  printf ("Character i is found at position %d\n",p-string+1);
  printf ("First occurrence of character \"i\" in \"%s\" is" \
          " \"%s\"",string, p);

   return 0;
}
```

Output:

```
Character i is found at position 3
First occurrence of character "i" in "This is a string for testing" is "is is a string for testing"
```

If you want to find every occurrence of a character in a given string, you can use below C program.

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a string for testing";
  char *p;
  int k = 1;
  p = strchr (string,'i');
  while (p!=NULL)
  {
    printf ("Character i found at position %d\n",p-string+1);
    printf ("Occurrence of character \"i\" : %d \n",k);
    printf ("Occurrence of character \"i\" in \"%s\" is \"%s\" \ "\"\n",string, p);
    p=strchr(p+1,'i');
    k++;
  }
  return 0;
}
```

Output:

```
Character i is found at position 3
Occurrence of character “i” : 1
Occurrence of character “i” in “This is a string for testing” is “is is a string for testing”
Character i is found at position 6
Occurrence of character “i” : 2
Occurrence of character “i” in “This is a string for testing” is “is a string for testing”
Character i is found at position 14
Occurrence of character “i” : 3
Occurrence of character “i” in “This is a string for testing” is “ing for testing”
Character i is found at position 26
Occurrence of character “i” : 4
Occurrence of character “i” in “This is a string for testing” is “ing”
```

C – strstr() function

- strstr() function returns pointer to the first occurrence of the string in a given string

char *strstr(const char *str1, const char *str2);

Example program for strstr() function in C:

- In this program, strstr( ) function is used to locate first occurrence of the string “test” in the string ”This is a test string for testing”. Pointer is returned at first occurrence of the string “test”.

```
#include <stdio.h>
#include <string.h>
int main ()
{
  char string[55] ="This is a test string for testing";
  char *p;
  p = strstr (string,"test");
  if(p)
  {
    printf("string found\n" );
    printf ("First occurrence of string \"test\" in \"%s\" is"\
           " \"%s\"",string, p);
  }
  else printf("string not found\n" );
   return 0;
}
```

Output:

```
string found
First occurrence of string “test” in “This is a test string for testing” is “test string for testing”
```

C – strlwr() function

- strlwr( ) function converts a given string into lowercase.

char *strlwr(char *string);

- strlwr( ) function is non standard function which may not available in standard library in C.

Example program for strlwr() function in C:

In this program, string "MODIFY This String To LOwer" is converted into lower case using strlwr( ) function and result is displayed as "modify this string to lower".

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str[ ] = "MODIFY This String To LOwer";
   printf("%s\n",strlwr (str));
   return  0;
}
```

Output:

```
modify this string to lower
```

C – strupr() function

- strupr( ) function converts a given string into uppercase.

char *strupr(char *string);

- strupr( ) function is non standard function which may not available in standard library in C.

Example program for strupr() function in C:

In this program, string "Modify This String To Upper" is converted into uppercase using strupr( ) function and result is displayed as "MODIFY THIS STRING TO UPPER".

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

int main()
{
   char str[ ] = "Modify This String To Upper";

   printf("%s\n",strupr(str));

   return  0;
}
```

Output:

```
MODIFY THIS STRING TO UPPER
```

C – strrev() function

- strrev( ) function reverses a given string in C language.

char *strrev(char *string);

- strrev( ) function is non standard function which may not available in standard library in C.

Example program for strrev() function in C:

- In below program, string "Hello" is reversed using strrev( ) function and output is displayed as "olleH".

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

int main()
{
   char name[30] = "Hello";

   printf("String before strrev( ) : %s\n",name);

   printf("String after strrev( )  : %s",strrev(name));

   return 0;
}
```

Output:

```
String before strrev( ) : Hello
String after strrev( )     : olleH
```