UNIT-3 (STRINGS) (RAM GOPAL GUPTA- http://ramgopalgupta.com/)

PART-2

String library functions:

Sr.No.	Function & Purpose
1	**strcpy(s1, s2);** Copies string s2 into string s1.
2	**strcat(s1, s2);** Concatenates string s2 onto the end of string s1.
3	**strlen(s1);** Returns the length of string s1.
4	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);** Returns a pointer to the first occurrence of string s2 in string s1.

STRING FUNCTIONS EXPLANATION WITH SOURCE CODE: TRY THESE CODES

1. **strcpy(s1, s2):** Copies string s2 into string s1.

```
#include <stdio.h>
#include <string.h>
int main () {

   char s1[20] = "SMS";
   char s2[20] = "VARANASI";

   /* copy s2 into s1 */
   strcpy(s1, s2);                              // s1 will hold “VARANASI”
   printf("strcpy( s1, s2) :  %s\n", s1 );

 return 0;
}
```

When the above code is compiled and executed, it produces the following result −

strcpy(s1, s2) : VARANASI

Explanation: value of s2 which is “VARANASI” will copy to the array s1 therefore it is “VARANASI”

---

2. **strcat(s1, s2):** Concatenates string s2 onto the end of string s1.

```
#include <stdio.h>
#include <string.h>
int main () {

   char s1[20] = "SMS";
   char s2[20] = "VARANASI";

   /* concatenates s1 and s2 */
   strcat( s1, s2);                             // s1 will hold “SMSVARANASI”
   printf("strcat( s1, s2):   %s\n", s1 );

return 0;
}
```

When the above code is compiled and executed, it produces the following result −

strcpy(s1, s2) : SMSVARANASI

Explanation: value of s2 which is “VARANASI” will be concatenated with the value of s1 and then this concatenated value will store in s1 therefore s1 will print “SMSVARANASI”

3. **strlen(s1):** Returns the length of string s1.

```
#include <stdio.h>
#include <string.h>
int main () {
   char s1[20] = "SMS";
   int len;
   /* total length of s1 */
   len = strlen(s1);                          // len will have length of s1 i.e. 3
   printf("strlen(s1) :  %d\n", len );
return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
strlen( s1 ) :  3
```

Explanation: String s1has value “SMS”. It contains three characters therefore length of s1 is 3. Remember strlen(...) function count the all alphanumeric values including special characters and spaces.

---

4. **strcmp(s1, s2):**
   Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.

```
#include <stdio.h>
#include <string.h>
int main () {

   char s1[20] = "SMS";
   char s2[20] = "VARANASI";
   int cmp;

    /* compare str1 with str2  */
   cmp = strcmp(s1,s2);                       // cmp will have value -1; see explanation
   printf("strcmp(s1,s2) :  %d\n", cmp );

return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
strcmp( s1,s2 ) :  -1
```

Explanation: String s1 is compared with s2 character by character. First char ‘S’ of s1 is compared with first char ‘V’of s2. As we know the ASCII value of ‘S’ is less that ASCII value of ‘V’ therefore strcmp(...) function return the value -1 and don’t continue the comparison process future. ***Important Strcmp(...) function does not consider the length of the string.***

5. **strchr(s1, ch):**
   Returns a pointer to the first occurrence of character ch in string s1.

```
#include <stdio.h>
#include <string.h>
int main () {

Step 1- char s1[10] = "VARANASI ";
Step 2- char *p;

    /* find the first occurrence of the character 'A' in s1  */
Step 3- p = strchr(s1,'A');       //return address of first occurrence of character 'A' in s1
Step 4- printf ("Character A is found at position: %d\n",p-s1+1); // display exact position of 'A'

return 0;
}
```

When the above code is compiled and executed, it produces the following result –
Character A is found at position: 2

Explanation: As we know that basically string is char array in C programming therefore it will work like a normal array.

Step-1 char s1[10]; when the step1 executed an array s1 was represented into the memory as shown below:

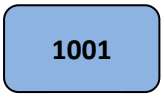
s1[10]

Index	0	**1**	2	3	4	5	6	7	8	9
Char Value	V	**A**	R	A	N	A	S	I	\0	
Address (assume)	1000	**1001**	1002	1003	1004	1005	1006	1007	1008	1009

Step-2 char *p; when this statement executed a pointer variable has been declared and does not hold the address of any variable.

p

Step-3 p = strchr(s1,'A'); when this statement executed the first address of character 'A' from s1 is stored into pointer p i.e. 1001 shown in green colour in above table.



p

Step-4 printf ("Character A is found at position: %d\n", p-s1+1);

In this statement, expression evaluated first is:

p-s1+1

Where:

p = 1001

s1 = 1000 // s1 is an array and if we write s1 it indicated address of its first element i.e. 1000

so the expression **p-s1+1** will give the result

= 1001 – 1000 + 1

= 1 + 1

= 2

The output will be:

Character A is found at position: 2

I you understand the working of strchr(...) function.

6. strstr(s1, s2):

Returns a pointer to the first occurrence of string s2 in string s1.

```
#include <stdio.h>
#include <string.h>
int main () {
    Step 1- char s[50] = "You are the best in this world.";
    Step 2- char searchString[10] = "best";
    Step 3- char *p;

  // this function returns the pointer of the first occurrence of the given string (i.e. searchString)

    Step 4- p = strstr(s, searchString);
    Step 5- printf("The substring starting from position in the given string: %d\n", p-s+1);
    Step 6- printf("The substring starting from the given string: %s\n", p);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result –

The substring starting from position in the given string: 13

The substring starting from the given string: best in this world.

Explanation:

Step-1 char s[50] = "You are the best in this world.";

It creates a char array (string) of size 50, it is represented into the memory as shown below:

s[50]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
Y	o	u		a	r	e		t	h	e		**b**	e	s	t		i	n		t	h	i	s		w	o	r	l	d	.	\0																			

Step-2 char searchString[10] = "best";

It creates a char array (string) of size 10, it is represented into the memory as shown below:

searchString[10]

0	1	2	3	4	5	6	7	8	9
b	e	s	t	\0					

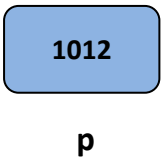
Step-3 char *p; when this statement executed a pointer variable has been declared and does not hold the address of any variable.

p

Step-4 p = strstr(s, searchString); when this statement executed the first occurrence of searchString i.e. “best” in s is transferred into pointer **p** shown in green colour in above table. Suppose the memory address scheme of s1 is as shown below:

Address of s[0] = 1000 then
Address of s[1] = 1001
Address of s[2] = 1002
Address of s[3] = 1003
Address of s[4] = 1004
.
.
.
Address of s[12] = 1012
.
.
So on

The search string “best” has its first occurrence at index 12 and the address of s[12] is 1012 which will transfer to pointer p



Step-5 printf("The substring starting from position in the given string: %d\n", p-s+1);

In this statement expression **p-s1+1** evaluated first.

Where:

p = 1012

s = 1000 // **s** is an array and if we write s it indicated address of its first element i.e. 1000

so the expression **p-s+1** will give the result

= 1012 – 1000 + 1
= 12 + 1
= 13

so output is:

The searchString starting from position in the given string: 13

Step-6 printf("The substring starting from the given string: %s\n", p);

In this statement **p** is a pointer and holding the memory address 1012 when we print the pointer using **%s** it will print the string starting from the address 1012 till null pointer i.e. ‘\0’

Therefore output is:

The substring starting from the given string: best in this world.

Try from yourself some other string functions like

strlwr()converts string to lowercase

strupr()converts string to uppercase

strcmpi ( ) Same as strcmp() function. But, this function negotiates case. "A" and "a" are treated as same.

strrchr ( ) last occurrence of given character in a string is found

strrstr ( ) Returns pointer to last occurrence of str2 in str1

strrev ( ) Reverses the given string