

## Handling Strings

A string is stored as an array of characters. The last character of this array has to be a NULL character (`'\0'`). For example, the following is a valid way to initialize a string to store "hello";

```
char word[20];

word[0] = 'h';
word[1] = 'e';
word[2] = 'l';
word[3] = 'l';
word[4] = 'o';
word[5] = '\0';
```

Every time you make up a string by assigning individual elements, you must include the NULL character at the end. Another way to read in a string is from the keyboard directly into a character array:

```
scanf("%s", last_name);
```

where `%s` is the format to read a complete string. The computer will read everything into the string variable till it encounters a space. Note in the above statement `last_name` refers to an array of characters, and points to the first location of the string being read from the keyboard. Therefore, there is no need to put the “&” sign before the variable name, that we use while reading in integer and real variables from the keyboard. It is already a pointer to the address of the variable.

### Assigning values to strings of characters.

```
#include <stdio.h>
int main ( )
{
    int k,;
    char name[20], course[20];
    course = "C_Programming";
    printf( "\nprint a name =");
    scanf(      "%s",      name);

    printf( "\n %s is a student in %s., name, course );
}
```

**JOHN is a student of C\_Programming.**

Every string read this way is automatically terminated by a “NULL” character (`'\0'`)

J	O	H	N	\0										
---	---	---	---	----	--	--	--	--	--	--	--	--	--	--

O	P	R	A	H	\0									
---	---	---	---	---	----	--	--	--	--	--	--	--	--	--

G	A	B	R	I	E	L	\0							
---	---	---	---	---	---	---	----	--	--	--	--	--	--	--

You can operate on the individual characters by using array operations. Here is an example which converts all small case letters to upper case letters for a string. Note the last character is left untouched.

```
void to_upper(char *word) {  
    int index = 0;  
    while (word[index] != '\0') {  
        word[index] = toupper(word[index]);  
        index++;  
    }  
}
```

You can also write functions to find the length of a string, or to copy one string onto another or compare two strings to check if they are same or not. However, 'C' provides a set of library functions for manipulation of strings.

Library functions for string manipulation  
**strlen** :find length            and  
**strcat** : Join two strings ( concatenate)

```
#include <stdio.h>
#include<string.h>
int main ( ){
    int length1,length2;
    char string1[20] = "alpha";
    char string2[20] = "beta";

    length1 = strlen( string1 );
// finds length of the string
    printf( "\n length of  %s is %d", string1, length1 );

    strcat ( string1, string2);
//concatenates two strings
    printf( "\n the new string is %s", string1 );
    length2 = strlen( string1 );
    printf( "\n length of  new string is  %d", length2 );
    printf( "\n\n");
}
length of  alpha is 5
the new string is alphabeta
length of  new string is 9
```

**You can call one function within another one:**

( Using Library functions  
**strlen**                    **strcat**)

```
#include <stdio.h>
#include<string.h>
int main ( ){
    int length1,length2;
    char str1[20] = "alpha";
    char str2[20] = "beta";
    printf("\nlength is  %d", strlen(strcat(str1, str2) ));
    printf( "\n\n");
}

new length is 9
```

Always make sure **string1** has enough space in the array, whenever you are trying to join two strings.

**Library functions for string manipulation**  
**strcpy (to copy one string onto another)**

```
#include <stdio.h>
#include<string.h>
int main ( )
{
    int length1,length2;
    char string1[20] = "alpha";
    char string2[20] = "assignment";
    printf( "\n the old string is %s", string1 );

    strcpy ( string1, string2);
    //copies string2 on top of string1
    printf( "\n and the new string is %s", string1 );
    length2 = strlen( string1 );
    printf( "\n length of new string is %d", length2 );
    printf( "\n\n");
}
```

```
the old string is alpha
and the new string is assignment
length of new string is 10
```

## Library functions for string manipulation

### strcmp : (comparing two strings s1 and s2)

This function compares two strings lexicographically.

If the strings are identical, the function returns 0.

If s1 comes before s2 the function returns a negative number.

If s1 comes after s2 the function returns a positive number.

```
#include <stdio.h>
#include<string.h>
int main ( ){
    int length1,length2;
    char string1[20] = "alpha";
    char string3[10] = "alpha";
    char string2[20] = "assignment";
    if ( strcmp(string1, string2) != 0)
        printf( "\n string1 and string2 are not same");
    if ( strcmp(string1, string2) < 0)
        printf( "\n string1 comes before string2 ");
    if ( strcmp(string1, string3) == 0)
        printf( "\n string1 and string3 are same");
}
```

string1 and string2 are not same

string1 comes before string2

string1 and string3 are same

## Reading a group of names from the keyboard into an array

Each name is stored as 1-Dimensional array. A group of names would thus need 2-Dimensional array for storage. The first index would refer to number of names and the second index would refer to maximum number of characters in each name. Here is an example of storing 4 names

J	O	H	N	\0										
S	T	E	V	E	N	\0								
B	E	R	B	E	R	E	T	\0						
A	N	D	R	E	W	\0								

```
#include <stdio.h>
#include<string.h>
int main ( )
{
    int k,;
    char names[4][15];

    for(k =0; k<4; k++){
        printf( "\nprint a name =");
        scanf("%s",names[k]);
    }
    for(k =0; k<4; k++){

        printf( "\n name %d is %s",k+1, names[k]);
    }
    return 0;
}
```

**Reading a group of names into an array  
And corresponding values into another array**

```
#include <stdio.h>
#include<string.h>
int main ( )
{
    int k,;
    char names[4][20];
    int grade [4];
    for(k =0; k<4; k++){
        printf( "\nprint a name and grade =");
        scanf( "%s%d", names[k], &grade[k]);
    }
    for(k =0; k<4; k++){

        printf( "\n name %d is %s with grade %d",k+1,
                names[k], grade[k]);
    }
    return 0;
}
```