UNIT-1

- Strings – Declaration of strings, Initialization of strings using arrays and pointers,
- Standard library functions of <string.h>header file,
- Null-terminated strings,
- Char arrays and pointers,
- Pointers and Strings,
- comparing two strings,
- find substring in a string,
- tokenizing a string with strtok() function, pointer-based string-conversion function – atoi()

Introduction To Strings

- In C, strings are treated as arrays of type char and are terminated by a null character('\0').

“Array or string of characters terminated by null character ‘\0’.”

This null character has ASCII value zero.

For ex:

"LION" is a string with 5 characters including the null character stored inside the computer memory.

This string can be represented as:

0	1	2	3	4
‘L’	‘I’	‘O’	‘N’	‘\0’

- There is no separate data type for strings in C. They are treated as arrays of type char.

Declaration of a String:

Strings are declared in C using the **char** data type.

For ex:

char name[5];

//string variable name can store maximum of 5

//characters including the NULL character denoted as **'\0'**

name[0]	name[1]	name[2]	name[3]	name[4]
				\0

|← Garbage values →|

Initialization of a String:

char str[10] = {'I', 'n', 'd', 'i', 'a', '\0' };

char str[10] = "India"; /*Here the null character is automatically placed at the end*/

For ex:

char name[5]={‘L’,’I’,’O’,’N’};

char s[9]="LION";

The above declaration can be represented as:

name[0]	name[1]	name[2]	name[3]	name[4]
‘L’	‘I’	‘O’	‘N’	‘\0’

string constant is a sequence of characters enclosed in double quotes. It is sometimes called a literal.
The double quotes are not a part of the string. Some examples of string constants are-

“V”

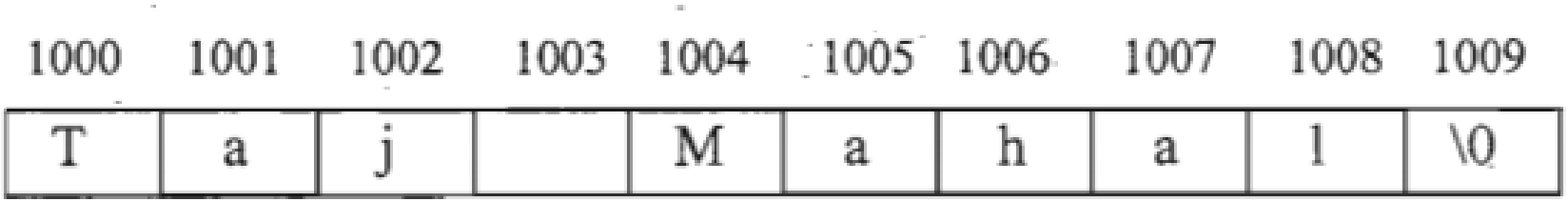
“Taj MahaI”

“2345”

“Subhash Chandra Bose was a great leader”

“My age is %d and height is %f\n” *(Control string used in printf)*

For example the string "Taj MahaI" will be stored in memory as-



• Each character occupies one byte and compiler automatically inserts the null character at the end

Example: Program to demonstrate initialization of a string.

```
#include<stdio.h>
void main()
{
    char name[5]={ ‘L’,’I’,’O’,’N’};                    //or char s[5]="LION";
    printf("Character in the array at First position: %c \n", name[0]);
    printf("Character in the array at Second position: %c \n", name[1]);
    printf("Character in the array at Third position: %c \n", name[2]);
    printf("Character in the array at Fourth position: %c ", name[3]);
}
```

```
Character in the array at First position: L
Character in the array at Second position: I
Character in the array at Third position: O
Character in the array at Fourth position: N
--------------------------------
```

Reading & Printing of Strings

The strings can be accepted from the user using the following formatted functions:

scanf()	to accept the string from the user
printf()	to print a string to the screen

Example: Program to illustrate the use of scanf() and printf().

```
#include<stdio.h>
void main()
{
char str[15];
printf("Type a string: \n");
scanf("%s", &str);
printf("You typed the string:%s", str);
}
```

Output:

Type a string: LION

You typed the string: LION

- **Problem with scanf in reading a string and its solution:**

  scanf( ) can read a string from the input stream until the first occurrence of space. Hence, to enable scanf( ) to read a string until a newline character ie '\n', few modification can be done to the scanf :

Before modifying the scanf() statement:

```
#include<stdio.h>
void main()
{
    char str[15];
    printf("Type a string: \n");
    scanf("%s", &str);
    printf("You typed the string:%s", str);
}
```

Output:

Type a string: **LION THE KING OF JUNGLE**
You typed the string: **LION**

After modifying the scanf() statement:

```
#include<stdio.h>
void main()
{
    char str[15];
    printf(“Type a string: \n”);
    scanf(“%^[\n]s”, &str);
    printf(“You typed the string:%s”, str);
}
```

Output:

Type a string: **LION THE KING OF JUNGLE**

You typed the string: **LION THE KING OF JUNGLE**

Unformatted input/output String functions: gets, puts

- **gets()** to read a string from the user until the user enters a newline Character ie '\n' (presses Enter key)

gets() takes the starting address of the string which will hold the input. The string inputted using gets() is automatically terminated with a null character.

The string can also be read by calling the **getchar()** repeatedly to read a sequence of single characters (unless a terminating character is entered) and simultaneously storing it in a character array.

- **puts()** to display a string to the screen.

The string can also be written by calling the **putchar()** repeatedly to print a sequence of single characters

Example: Program to illustrate the use of gets() and puts().

```
#include<stdio.h>
void main()
{
    char str[15];
    printf(“Type a string: \n”);
    gets(str);                          //same as scanf(“%^[\n]s”, str);
    printf(“\nYou typed: ”);
    puts(str);                          //same as printf(“%s”, str);
}
```

Output:
Type a string: **Programming in C**
You typed: **Programming in C**

String Library Functions

- There are several library functions used to manipulate strings.
- The prototypes for these functions are in header file **string.h**.

#include< string.h >

STRING MANIPULATION FUNCTIONS

- Whenever strings needs to be manipulated in a program manually it adds the extra lines of program code and also makes it a very lengthy and difficult to understand.
- To avoid this C supports a large number of string handling functions. There are many functions defined in <string.h> header file.

Sl. No.	Function Name & its meaning
1	strlen(s1): → Returns the length of the strng s1
2	strcpy(s1,s2) → Copies the string s2 into s1
3	strncpy(s1,s2,n) → Copies first n characters of string s2 into s1
4	strcat(s1,s2) → Concatenates/Joins the string s2 to the end of s1
5	strncat(s1,s2,n) → Concatenates/Joins first n characters of string s2 to the end of s1
6	strcmp(s1,s2) → compares string s1 with s2; if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than zero; otherwise if s1>s2 it returns a value greater than zero.
7	strncmp(s1,s2,n) → compares first n characters of string s1 with s2; if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than zero; otherwise if s1>s2 the it returns a value greater than zero.

8	strcmpi(s1,s2) → compares string s1 with s2 by ignoring the case (uppercase or lowercase); if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than zero; otherwise if s1>s2 the it returns a value greater than zero.
9	strchr(s1,ch)→ Returns a pointer to the first occurrence of character ch in s1
10	strstr(s1,s2)→ Returns a pointer to the first occurrence of the string s2 in s1
11	strrev(s1)→ Returns the reverse string after reversing the characters of the string s1

12	strtok(s1,delimiter): // splits str into tokens separated by the delimiters. It needs a loop to get all the tokens and it return NULL when there are no more tokens.
	char *strtok(char str[], const char *delims);

strlen( )

- The function calculates & returns the length of a string passed to it as an argument.

  This function returns the length of the string i.e. the number of characters in the string excluding the terminating null character.

It accepts a single argument, which is pointer to the first character of the string.

For example:

strlen("suresh") returns the value **6**.

Similarly if s1 is an array that contains the name "deepali" then strlen(s1) returns the value **7** .

Example 1: Program to understand the work of strlen () function

```
#include<stdio.h>
#include<string.h>
int  main( )
{
char str[20];
int length;

printf ("Enter the string: \n") ;
scanf ("%s", str);

length=strlen(str) ;

printf("Length of the string is  %d\n", length);
}
```

Example 2: Program to illustrate the use of strlen().

```
#include<string.h>
#include<stdio.h>
void main()
{
    char str[20];
    int len;
    printf("Type a string:");
    gets(str);
    len=strlen(str);
    printf("\nLength of the string %s is %d ", str,len);
}
```

Output:

Type a string: **COLOR**

Length of the string COLOR is 5

strcpy( )

- This function is used for copying one string to another string.

  strcpy(str1, str2) copies str2 to str1**.**

Here str2 is the source string and str1 is destination string.

Similarly:

strcpy(s1,s2)

Function will copy the content of string s2 into another string s1.

- If str2 = "suresh" then this function copies "suresh" into str1.
- *This function takes pointers to two strings as arguments and returns the pointer to first string.*

Program to understand the work of strcpy () function

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[10], str2[10];
printf ("Enter the first string: " );
scanf("%s",str1);
printf ("Enter tne second string: " ) ;
scanf("%s",str2) ;
strcpy(str1,str2);
printf ("First string : %s \t\t Second string : %s\n",str1,str2);
strcpy(str1,"Delhi");
strcpy(str2,"Calcutta");
printf ("First string : %s \t\t Second string : %s\n",str1,str2);
}
```

• Example: Program to illustrate the use of strcpy().

```
#include<string.h>
#include<stdio.h>
void main()
{
char s1[20],s2[20];
printf("Enter A string: ");
gets(s2);
strcpy(s1,s2); //Content of string s2 is copied into string s2
printf("Copied string:");
puts(s1);
}
```

Output:

Enter string: **PROGRAMMING IN C**

Copied string: **PROGRAMMING IN C**

strcmp( )

- This function is used for comparison of two strings.
- If the two strings match, strcmp() returns value 0,
- otherwise it returns a non-zero value.
- This function compares the strings character by character.
- The non-zero value returned on mismatch is the difference of the ASCII value of the non-matching characters of the two strings.

strcmp(s1, s2) returns a value-

< 0 when s1 < s2

= 0 when s1 = = s2

> 0 when s1 > s2

Program to understand the work of strcmp() function

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[10], str2[10];

    printf ("Enter the first string ") ;
    scanf("%s",str1);

    printf ("Enter the second string :");
    scanf("%s",str2);
          if( (strcmp(str1,str2) )==0)
          printf ("Stririgs are same\n");
                    else
                    printf("Strings are not same\n");
}
```

strcat( )

- This function is used for concatenation of two strings.
- If first string is “Graphic" and second string is “Era” then after using this function the first string becomes “GraphicEra".

strcat(str1, str2); /*concatenates str2 at the end of str1 */

- The null character from the first string is removed, and the second, string is added at the end of first string. The second string remains unaffected.
- This function takes pointer to two strings as arguments and returns a pointer to the first(concatenated) string.

Program to understand the work of strcat( ) function.

```
#include<stdio.h>
#include<string.h>
int main( )
{
char str1[20],str2[20];
printf ("Enter the first string: ");
scanf("%s",str1);
printf ("Enter the second string: ");
scanf("%s",str2) ;
strcat(str1,str2);
printf( "First string %s \t Second string %s \n", str1, str2);
strcat(str1, "_one");
printf ("Now first string is:  %s \n", str1) ;
}
```

Output:

```
Enter the first string : data
Enter the second string : base
First string : database          Second string : base
Now first string is  : database_one
```