String:

String is an array of characters. Data of the same type are stored in an array for example, Integers can be stored in an integer array, similarly, a group of characters can be stored in a character array. **Character arrays are also called strings**. A string is a one-dimensional array of characters that is terminated by a null (‘\0’).

Declaration of strings:

Declaring a string is very simple, same as declaring a one-dimensional array. Below is the syntax for declaring a string.

```
char string_name[size];
```

In the above syntax, string_name is any name given to the string variable, and size is used to define the length of the string, i.e the number of characters that the strings will store. Keep in mind that there is an extra terminating character which is the null character (‘\0’) that is used to indicate the termination of string.

```
char name[ ] = { 'F', 'E', 'N', 'I', 'L', '\0' } ;
```

Each character in the array, like “H”, occupies one byte of memory and the last character is always ‘\0’. The null character ‘\0’ looks like two characters, but it is actually only one character, with the \ indicating that what follows it is something special. Character array elements are stored in contiguous memory locations.

Equally we can make the string by assigning character values to each member.

```
name[0]='F';
name[1]='E';
name[2]='N';
name[3]='I';
name[4]='L';
name[5]='\0';
```

The placeholder for string variables is %s.

Note: There is difference between ‘\0’ and ‘0’. The ASCII value of ‘\0’ is 0, and for ‘0’, the value is 48. The terminating null ‘\0’ and ‘0’ are not same.

The null (‘\0’) is important in C programming because it is the only way the functions that work with a string can know where the string ends. When a string not terminated by a ‘\0’, then it is not really a string but merely a collection of characters.

Example of string in C:-

```c
#include<stdio.h>
int main()
{
    // declare and initialize string
    char str[] = "Linef";
    printf("%s",str);
    return 0;
}
//Output:- Linef
```

In order to read a string that contains the spaces, we use the gets() function. The purpose of gets is to ignore the whitespaces. When a newline is reached, gets stops reading. For example:

```c
#include <stdio.h>
int main() {
char name[50];
printf("Enter your name: ");
gets(name);
printf("My name is %s ",name);
return 0;}
```

Conclusion:-

Today we have learned about how to use strings in C programming. Each member of the array contains one of the characters in the string. By using scanf() we are not capable of receiving multi-word strings. The way to get a multi-word string is by using the function gets().

String Functions & string.h Library:

strcat( ):-

This function is used to concatenates the source string at the end of the target string. For example, “Hello” and “World” on concatenation would result into a string “HelloWorld”. Here is an example of strcat( ):

```c
int main( ) {
```

```
char  s[ ] = "Hello" ;
char  t[30] = "World" ;
strcat ( t, s ) ;
printf ( "String = %s", t ) ;
}
//Output: string = HelloWorld
```

strlen( ):-

This function is used to counts the number of characters present in a string. Its example is given below:

```
int main( ) {
char  str[ ] = "Fenil" ;
int  str_length;
str_length= strlen ( str ) ;
printf ( " length = %d", str_length );
}
//Output: length = 5
```

strcpy( ):-

This function is used to copies the contents of one string into another. The base addresses of the source and target strings should be given to this function. Here is an example of strcpy( ):

```
int main( ) {
char  s[ ] = "Linef" ;
char  t[20] ;
strcpy ( t, s ) ;
printf ( "\n Source string = %s", s ) ;
printf ( "\n Target string = %s", t ) ; }
//And here is the output...
//Source string = Linef
//Target string = Linef
```

strcmp( ):-

This function is used to compares two strings to find out whether they are same or different. The strcmp() will compare two strings character by character until there is a

mismatch or end of one of the strings is reached. If both of the strings are identical, strcmp( ) returns a value zero. If they are not identical, it will return the numeric difference between the ASCII values of the first non-matching pairs of characters. Here is a example of strcmp( ).

```
#include <stdio.h>
#include <string.h>
int main()
{
char string1[ ] = "Fenil" ;
char string2[ ] = "Code" ;
int a;
a= strcmp ( string1, string2 ) ;
printf ("\n%d", a) ;
return 0;
}
//Output:5
```

strrev():-

This function is used to show the reverse of the string. Following are the example of strrev():

```
#include<stdio.h>
#include<string.h>
int main()
{
char str[50] = "12345";
printf("After reversing string is =%s",strrev(str));
return 0;}
//Output: After reversing string is = 54321
```

There are many other built-in string funtions like **strlwr , strupr ,strcat , strdup and strset,** these functions are also very useful in manipulating the strings. In today's tutorial we have discussed only few of the string functions. You can explore more about string functions by searching it on the internet.