MODULE IV

Strings

A string constant is a one-dimensional array of characters terminated by a null ('\0'). For example,
char name[ ] = { 'H', 'A', 'E', 'S', 'L', 'E', 'R', '\0' } ;
We can also that character arrays are also called as strings. Each character in the array occupies one byte of memory and the last character is always '\0'.

H	A	E	S	L	E	R	\0
65518	65519	65520	65521	65522	65523	65524	65525

The above string can be initialized in another method also:
char name[ ] = "HAESLER" ;
Note that, in this declaration '\0' is not necessary. C inserts the null character automatically.

Example program with string:	**Same program can be implemented using pointers:**

```
main( )
{
        char name[ ] = "Klinsman" ;
        int i = 0 ;
        while ( name[i] != `\0' )
        {
                printf ( "%c", name[i] ) ;
                i++ ;
        }
}
```
	```
	main( )
	{
	        char name[ ] = "Klinsman" ;
	        char *ptr ;
	        ptr = name ; /* store base address of string */
	        while ( *ptr != `\0' )
	        {
	                printf ( "%c", *ptr ) ;
	                ptr++ ;
	        }
	}
	```

And here is the output... Klinsman	In this program, This base address is stored in the variable ptr using, ptr = name ; Once the base address is obtained in ptr, *ptr would yield the value at this address and the value is printed by the statement printf ("%c", *ptr) ; On incrementing a pointer it points to the immediately location of next character. This process is carried out till ptr doesn't point to the last character in the string,that is, '\0'.

```
main( )
{
        char name[ ] = "Klinsman" ;
        printf ( "%s", name ) ;
}
```

The %s used in printf( ) is a format specification for printing out a string. The same specification can be used to receive a string from the keyboard, as shown below.

```
main( )
{
        char name[25] ;
        printf ( "Enter your name " ) ;
        scanf ( "%s", name ) ;
        printf("The entered name is ");
        printf ( "%s", name ) ;
}
```

Output

```
Enter your name
Ram
```

```
The entered name is Ram
```

But using scanf we can read only one word, i.e, multi words are not allowed. So for reading strings with more than one word we use the library function **gets** and for printing it , we use **puts.**

```
main()
{
        char name[25];
        printf("Enter the name");
        gets(name);
        printf("The entered name is ");
        puts(name);
}
```

Output

```
Enter your name
Athira Krishnan
The entered name is Athira Krishnan.
```

Standard Library String Functions

With every C compiler a large set of useful string handling library functions are provided.

strlen	Finds length of a string
strlwr	Converts a string to lowercase
strupr	Converts a string to uppercase
strcat	Appends one string at the end of another
strncat	Appends first n characters of a string at the end of another
strcpy	Copies a string into another
strncpy	Copies first n characters of one string into another
strcmp	Compares two strings
strncmp	Compares first n characters of two strings
strcmpi	Compares two strings without regard to case ("i" denotes that this function ignores case)
stricmp	Compares two strings without regard to case (identical to strcmpi)
strnicmp	Compares first n characters of two strings without regard to case
strdup	Duplicates a string
strchr	Finds first occurrence of a given character in a string
strrchr	Finds last occurrence of a given character in a string
strstr	Finds first occurrence of a given string in another string
strset	Sets all characters of string to a given character
strnset	Sets first n characters of a string to a given character
strrev	Reverses a string

strlen( )

This function counts the number of characters present in a string.

```
main( )
{
        char arr[ ] = "Bamboozled" ;
        int len1, len2 ;
        len1 = strlen ( arr ) ;
        printf ( "length of the string is = %d", len1 ) ;
}
```

The output would be...

```
length of the string is 10
```

<u>Same program using pointer without using strlen</u>

```
main()
{
        char arr[ ] = "Bamboozled" ,*s;
        int len1,llength = 0 ;
        s=arr;
        while ( *s != '\0' )
        {
                length++ ;
                s++ ;
        }
        printf(“length of string is %d”,length);
}
```

The output would be the same,
length of the string is 10

strcpy( )

This function copies the contents of one string into another. The base addresses of the source and target strings should be supplied to this function.

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

And here is the output...

```
source string = Sayonara
target string = Sayonara
```

On supplying the base addresses, strcpy( ) goes on copying the characters in source string into the target string till it doesn't encounter the end of source string (‘\0’). It is our responsibility to see to it that the target string’s dimension is big enough to hold the string being copied into it.

<u>String copying without using strcpy</u>

```
main( )
{
        char s[ ] = "Sayonara" ;
        char t[20] ;
        while ( *s != '\0' )
        {
                *t = *s ;
                s++ ;
                t++;
        }
        *t='\0';
        printf ( "\nsource string = %s", s) ;
        printf ( "\ntarget string = %s", t) ;
}
```

And here is the output...

```
source string = Sayonara
target string = Sayonara
```

<u>strcat( )</u>

This function concatenates the source string at the end of the target string. For example, “Bombay” and “Nagpur” on concatenation would result into a string “BombayNagpur”.

```
main( )
{
        char source[ ] = "Folks!" ;
        char target[30] = "Hello" ;
        strcat ( target, source ) ;
        printf ( "\nsource string = %s", source ) ;
        printf ( "\ntarget string = %s", target ) ;
}
```

And here is the output...

```
source string = Folks!
target string = HelloFolks!
```

<u>Program of string concatenation withour using strcat</u>

```
#include <stdio.h>
main()
{
  char str1[50], str2[50], i, j;
  printf("\nEnter first string: ");
  scanf("%s",str1);
  printf("\nEnter second string: ");
  scanf("%s",str2);
   for(i=0; str1[i]!='\0'; ++i);
   for(j=0; str2[j]!='\0'; ++j, ++i)
  {
            str1[i]=str2[j];
  }
   str1[i]='\0';
  printf("\nOutput: %s",str1);

 }
```

<u>Program of string concatenation withour using strcat using pointer</u>

```
#include <stdio.h>

main()
{
   char aa[100], bb[100];

   printf("\nEnter the first string: ");
   gets(aa);   // inputting first string
```

```
    printf("\nEnter the second string to be concatenated: ");
    gets(bb);   // inputting second string

    char *a = aa;
    char *b = bb;

    // pointing to the end of the 1st string
    while(*a)!='\0'   // till it doesn't point to NULL-till string is not empty
    {
       a++;    // point to the next letter of the string
    }
    while(*b)   // till second string is not empty
    {
       *a = *b;
       b++;
       a++;
    }
    *a = '\0';  // string must end with '\0'
    printf("\n\n\nThe string after concatenation is: %s ", aa);
 }
```

strcmp( )

This is a function which compares two strings to find out whether they are same or different. The two strings are compared character by character until there is a mismatch or end of one of the strings is reached, whichever occurs first. If the two strings are identical, strcmp( ) returns a value zero. If they're not, it returns the numeric difference between the ASCII values of the first non-matching pairs of characters.

	Program to string compare without using strcmp

main( )
{
char string1[ ] = "Jerry" ;
char string2[ ] = "Ferry" ;
int i, j, k ;
j = strcmp (string1, string2)
if(j==0)
printf("Strings are equal");
else
printf("Strings are not equal");
}
	int main()
	{
	char str1[30], str2[30];
	int i;
	printf("\nEnter two strings :");
	gets(str1);
	gets(str2);
	i = 0;
	while (str1[i] == str2[i] && str1[i] != '\0')
	i++;
	if (str1[i] > str2[i])
	printf("str1 > str2");
	else if (str1[i] < str2[i])
	printf("str1 < str2");
	else
	printf("str1 = str2");
	}

Output Strings are not equal. In the call to strcmp( ), Since the two strings are not identical—"Jerry" and "Ferry"—and the value returned by strcmp( ) will not be zero.	