The familiarity with arrays enables us to understand and work with strings-a special case of character arrays. The strings like any other data type may be constants or variables. The entire concept of arrays is directly applicable to strings. However C provides special features for handling strings.

A string is a single dimension array of characters terminated by a \0 (zero byte)

Consider the following program segment –(line numbers are for reference only.)

```
1. main()
2. {
3. char name[10];
4. printf("\nEnter your name ");
5. scanf("%s",name);
6. printf("\nYour name is %s ",name);
7. getch();
8. }
```

In this program a string is declared at #3 as a character array. The value to be read using scanf (line #5) the format specified used is **%s**. The printf also uses the same format specifier **%s** to print a string.

Consider another program that gives a value to a string variable

```
main()
{
   char name[]= "C PROGRAMMING";
  ……..
}
```

the memory map of this is

C		P	R	O	G	R	A	M	M	I	N	G	**\0**

Observe the null byte '\0' is inserted at the end of the array. This is the only way C can know when the string is terminated. If this '\0' is lost somehow, C has no way to tell where the string terminates.

Reading the string with **scanf** enables the user to read the text but the limitation with scanf is that once a blank space is encountered it terminates the reading process. So while entering strings to a variable using **scanf** if the user enters

Computer Programming in C

Only "Computer" will be stored in the string.

The alternative to be used is **gets**

gets - Get a line from the stdin stream.

char *gets(char *_buffer_);

Return Value

Returns its argument if successful i.e. the string it just read. A NULL pointer indicates an error.

Parameters

buffer Storage location for input string

Remarks

The gets function reads a line from the standard input stream stdin (in simple words keyboard) and stores it in buffer. The line consists of all characters up to and including the first newline character ('\n'). gets then replaces the newline character with a null character ('\0') before returning the line.

strcpy Copy a string.

char ***strcpy(char** **strDestination***, **const char** **strSource* **);**

Return Value

Returns the destination string.

Parameters

strDestination Destination string

strSource Null-terminated source string

Remarks

The **strcpy** function copies *strSource*, including the terminating null character, to the location specified by *strDestination*. No overflow checking is performed when strings are copied or appended. The behavior of **strcpy** is undefined if the source and destination strings overlap.

strlen Get the length of a string.

int strlen(const char ***string* **);

Return Value

Returns the number of characters in string, excluding the terminal NULL. No return value is reserved to indicate an error.

Parameter

string Null-terminated string