String Handling

Strings in C

Strings are defined as an array of characters.
The difference between a character array and a string is the string is terminated with a special character '\0'.

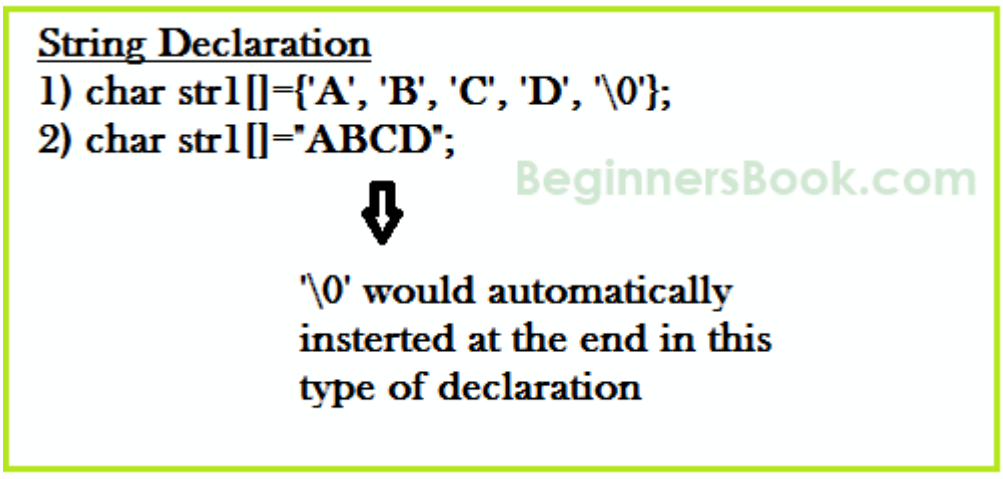
Declaration of string:

Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.

char str_name[size];

In the above syntax,
str_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 strings will store.
there is an extra terminating character which is the ***Null character ('\0')*** used to indicate ***end (termination)*** of string ,
which differs strings from normal character arrays.

String Declaration



Initializing a String:

A string can be initialized in different ways. Below is an example to declare a string with name as str and initialize it with “HVPM”.

```
1. char str[] = " HVPM ";
2. char str[10] = " HVPM ";
3. char str[] = {'H' , 'V' , 'P' , 'M' , '\0'};
4. char str[5] = {'H' , 'V' , 'P' , 'M' , '\0'};
```

Below is the memory representation of a string “HVPM”

str	H	V	P	M	\0
Address	2001	2002	2003	2004	2005

Program for declaring and initializing a string in C :-

```
// C program for declaration and initialization of string
#include<stdio.h>
#include<conio.h>
void main()
{
   char str[10] = "HVPM";   // declare and initialize string
   printf("%s",str);         // print string
   getch( );
}
```

Output:

```
HVPM
```

In the above program the string can be printed using a normal printf statements.
The C language does not provide an inbuilt data type for strings but it has an access specifier “**%s**” which can be used to directly print and read strings.

Below is a sample program to read a string from user:

```
// C program to read strings
 #include<stdio.h>
#include<conio.h>
void main()
{
   char str[50];          // declaring string
   scanf("%s",str);       // reading string
   printf("%s",str);      // print string
   getch( );
}
```

Output:-

"HVPM"
"HVPM"

in the above program the string can also be read using a single scanf statement.

Note:-[Also you might be thinking that why we have not used the '&' sign with string name 'str' in scanf statement! To understand this you will have to recall your knowledge of scanf. We know that the '&' sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces '[' and ']' will give the base address of this string. That's why we have not used '&' in this case as we are already providing the base address of the string to scanf.]

Array of Pointer to string

- Pointer stores the address of the variable.
- An "array of pointer" stores the addresses of all the elements of the array.
- An *"array of pointer to string"* stores the addresses of the string present in the array.
- The array contains the base address of every string element in the array.

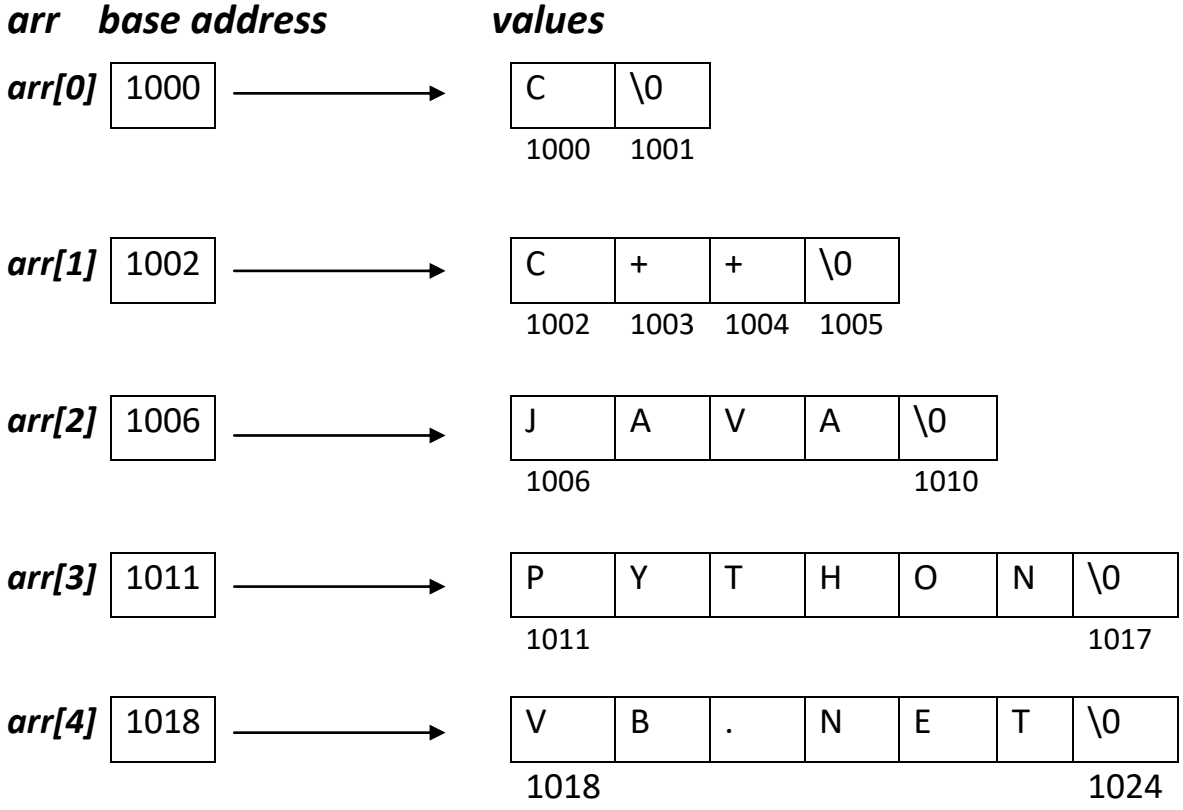
Ex. char *arr[5] = { "C", "C++", "JAVA", "PYTHON", "VB.NET" }

The arr[5]= stores 5-string type values and

The array variables as below:-

Char arr[5]	C	C++	JAVA	PYTHON	VB.NET
(Size/subscript)	*arr[0]*	*arr[1]*	*arr[2]*	*arr[3]*	*arr[4]*

The memory representation of *array of pointer to string* as below-



```
/* Program for array of pointer to string*/
#include<stdio.h>
#include<conio.h>
void main( )
{
 char  *arr[ ]= { “C”, “C++”, “JAVA”, “PYTHON”, “VB.NET” };
 int  i;
     for(i=0 ; i<5 ; i++)
    {
     printf(“%s  \t” arr[i] );
    }
  getch( );
}
```

Output:-

C C++ JAVA PYTHON VB.NET

String Handling Functions in C

- C programming language provides a set of pre-defined functions called **string handling functions** to work with string values.
- The string handling functions are defined in a header file called **string.h**
- Whenever we want to use any string handling function we must include the header file called **string.h**.

The following table provides most commonly used string handling function and their use:-

Function	Syntax (or) Example	Description
strcpy()	strcpy(string1, string2)	Copies string2 value into string1
strncpy()	strncpy(string1, string2, 5)	Copies first 5 characters string2 into string1
strlen()	n=strlen(string1)	returns total number of characters in string1 and store in variable n
strcat()	strcat(string1,string2)	Appends string2 to string1
strncat()	strncpy(string1, string2, n)	Appends first ‘n’ characters of string2 to string1
strcmp()	strcmp(string1, string2)	Returns 0 if string1 and string2 are the same; -ve value if string1<string2; +ve value if string1>string2
strncmp()	strncmp(string1, string2, n)	Compares first ‘n’ characters of both string1 and string2
strrev()	strrev(string1)	It reverses the value of string1
strset()	strset(string1, 'ch')	Sets all the characters of string1 to given character 'ch'.
strnset()	strnset(string1, 'ch', 5)	Sets first 5 characters of string1 to given character 'ch'.

Function	Syntax (or) Example	Description
strlwr()	strlwr(string1)	Converts all the characters of string1 to lower case.
strupr()	strupr(string1)	Converts all the characters of string1 to upper case.
strcmpi()	strcmpi(string1,string2)	Compares two strings, string1 and string2 by ignoring case (upper or lower)
stricmp()	stricmp(string1, string2)	Compares two strings, string1 and string2 by ignoring case (similar to strcmpi())
strdup()	string1 = strdup(string2)	Duplicated value of string2 is assigned to string1
strchr()	strchr(string1, 'b')	Returns a pointer to the first occurrence of character 'b' in string1
strrchr()	'strrchr(string1, 'b')	Returns a pointer to the last occurrence of character 'b' in string1
strstr()	strstr(string1, string2)	Returns a pointer to the first occurrence of string2 in string1

strcpy( )

strcpy(str1, str2)

It copies the string str2 into string str1, including the end character (terminator character or null charater '\0').

Example of strcpy:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str1[30] = "string 1";
    char str2[30] = "Amravati";
    /* this function has copied s2 into s1*/
    strcpy(str1,str2);
    printf("String str1 is: %s", str1);
    getch( );
}
```

Output:

```
String str1 is: Amravati
```

strncpy( )

strncpy(str1, str2, n)

n= integer type value

Case1: If length of str2 > n then it just copies first n characters of str2 into str1.

Case2: If length of str2 < n then it copies all the characters of str2 into str1 and appends several terminator chars('\0') to accumulate the length of str1 to make it n.

Example of strncpy:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str1[30] = "Nagpur";
    char str2[30] = "Amravati";
    /* this function has copied first 4 chars of s2 into s1*/
    strncpy(str1,str2, 4);
    printf("String str1 is: %s", str1);
    getch( );
}
```

Output:

```
String str1 is: Amar
```

Strlen( )

Syntax:

```
n = strlen(str)
```

n = is integer type value

It returns the length of the string without including end character **(terminating char '\0')**.

Example of strlen:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str1[20] = "Amravati";
    n= strlen(str1);
    printf("Length of string str1: %d",n);
    getch( );
}
```

Output:

```
Length of string str1: 8
```

strlen vs sizeof

strlen returns you the length of the string stored in array, however sizeof returns the total allocated size assigned to the array. So if I consider the above example again then the following statements would return the below values.

strlen(str1) returned value 13.

sizeof(str1) would return value 20 as the array size is 20 (see the first statement in main function).

strcat( )

syntax:-

strcat(str1, str2)

It concatenates two strings and returns the concatenated string.

Example of strcat:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char s1[10] = "Hello";
    char s2[10] = "World";
    strcat(s1,s2);
    printf("Output string after concatenation: %s", s1);
    getch( );
}
```

Output:

```
Output string after concatenation: HelloWorld
```

strncat( )

syntax:-

strncat(str1, str2, n)

It concatenates n characters of str2 to string str1. A **terminator char ('\0')** will always be appended at the end of the concatenated string.

Example of strncat:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char s1[10] = "Hello";
    char s2[10] = "World";
    strncat(s1,s2, 3);
    printf("Concatenation using strncat: %s", s1);
    getch( );
}
```

Output:

```
Concatenation using strncat: HelloWor
```

strcmp( )

n = strcmp(str1, str2)

It compares the two strings str1 and str2 and returns an integer value which is then assign to variable 'n'.

If both the strings are same (equal) then this function would return 0 otherwise it may return a negative or positive value based on the comparison.

If string1 < string2 then it would return a **-ve value.**

If string1 > string2 then it would return ***+ve value***.

If string1 == string2 then you would return **0(zero)**

Example of strcmp:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char s1[20] = "Amravati";
    char s2[20] = "Nagpur";
    n = strcmp( );
    if (n ==0)
    {
      printf("string 1 and string 2 are equal");
    }
   else
    {
       printf("string 1 and 2 are different");
    }
    getch( );
}
```

Output:

```
string 1 and 2 are different
```

How strcmp( ) function works:-

Ex. char str1[10] = “ab” ;
char str2[10] = “ac” ;
n = strcmp(str1, str2);

In above example, the strcmp( ) compares the str1 and str2 as below:-
str1 = ab
str2 = ac
It compares chacters of str1 and str2 one by one and find the ASCII difference

Character			ASCII value			
str1		str2	str1		str2	
a	-	a	97	-	97	= 0
b	-	c	98	-	99	= -1 (which is the final output)

strncmp( )

n = strncmp(str1, str2, n)

It compares first ‘n’ characters of both the strings.

Example of strncmp:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char s1[20] = "Amravati city";
    char s2[20] = " Amravati";
    int  n;
    /* below it is comparing first 8 characters of s1 and s2*/
    n = strncmp(s1, s2, 8);
    if (n==0)
    {
       printf("string 1 and string 2 are equal");
    }
   else
    {
       printf("string 1 and 2 are different");
    }
    getch( );
}
```

Output:

```
string1 and string 2 are equal
```

strrev( )

strrev(str1)

It reverse the given inputted string.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char s1[10] = "Hello";
    strrev(s1);
    printf("Reverse of string Hello is = %s", s1);
    getch( );
}
```

Output:

```
Reverse of string Hello is = olleH
```

strset( )

strset(str1 , 'ch')

It sets all the characters of string1 to given character 'ch'. Or

It replace all the charaters of string1 by character 'ch'.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( )
{
    char s1[10] = "Hello";
    strset( s1, '*');
    printf("String1 is = %s", s1);
    getch( );
}
```

Output:

String1 is = *****

(Note :- In above example all the charaters of word 'Hello' are replaced by '*'. Hence the output becomes 5-times * i.e. '*****'

strnset( )

strnset(str1 , 'ch' , n);

It sets all the characters of string1 to given character 'ch'. Or

It replace all the charaters of string1 by character 'ch'.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( )
{
    char s1[10] = "Hello";
    strnset( s1, '*' , 4);
    printf("String1 is = %s", s1);
    getch( );
}
```

Output:

String1 is = ****o

(Note :- In above example first 4-charaters of word 'Hello' are replaced by '*' and the last character 'o' remains as it is. Hence the output becomes 4-times * and character 'o' i.e. '****o'

strlwr( )

strlwr(str1)

It converts all the characters of string1 to lower case i.e small case.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( )
{
    char s1[10] = "HELLO";
    strlwr( s1, '*');
    printf("String1 is = %s", s1);
    getch( );
}
```

Output:

String1 is = hello

(Note :- In above example all the charaters of word 'HELLO' are converted to small case letters i.e hello.

strupr( )

strupr(str1)

It converts all the characters of string1 to lower case.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( )
{
    char s1[10] = "hello";
    strupr( s1, '*');
    printf("String1 is = %s", s1);
    getch( );
}
```

Output:

String1 is = HELLO

(Note :- In above example all the charaters of word 'hello' are converted to upper case letters i.e 'HELLO'.

gets( )

```
gets( str )
```

It reads a line from standard input and stores it into the string pointed by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.

- **str** − This is the string i.e array of characters where the C string is stored.

This function returns str on success, and NULL on error or when end of file occurs, while no characters have been read.

Example of strrev:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( )
{
   char s1[50];
printf(“Enter a string = ”);
   gets(s1);
   printf("String is = %s", s1);
   getch( );
}
```

Output:

```
Enter a string = Amravati city
String is = Amravati city
```