Strings in C

Rab Nawaz Jadoon

Assistant Professor

COMSATS IIT, Abbottabad

Pakistan

Department of Computer Science

DCS

COMSATS Institute of
Information Technology

What are Strings

- The way a group of integers can be stored in an integer array,
  - Similarly a group of characters can be stored in a character array.
  - Character arrays are many a time also called strings. Many languages internally treat strings as character arrays.
    - Character arrays or strings are used by programming languages to manipulate text such as words and sentences.

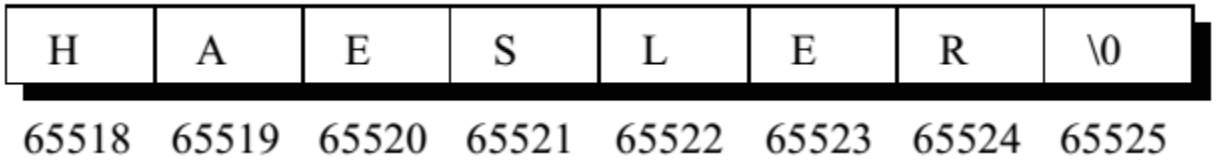
What are Strings

- 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' } ;
    - Each character in the array occupies one byte of memory and the last character is always '\0'.
    - What character is this? It looks like two characters, but it is actually only one character, with the \ indicating that what follows it is something special.
    - '\0' is called null character.
    - Note that '\0'and '0'are not same.
    - ASCII value of '\0' is 0, whereas ASCII value of '0' is 48.

What are Strings

- The terminating null ('\0') is important, because it is the only way the functions that work with a string can know where the string ends.



  - For example, the string used above can also be initialized as, char name[ ] = "HAESLER" ;
    - Note that, in this declaration '\0' is not necessary.
    - C inserts the null character automatically.

Program 1

```
/* Program to demonstrate printing of a string */
main( )
{
      char  name[ ] = "Klinsman" ;
      int  i = 0 ;

      while ( i <= 7 )
      {
            printf ( "%c", name[i] ) ;
            i++ ;
       }
 }
```

And here is the output...

Klinsman

Alternate way

```
main( )
{
        char  name[ ] = "Klinsman" ;
        int  i = 0 ;

        while ( name[i] != `\0' )
        {
                printf ( "%c", name[i] ) ;
                i++ ;
        }
}
```

This program doesn't rely on the length of the string (number of characters in it) to print out its contents

And here is the output...

Klinsman

Another alternative

```
#include<stdio.h>
#include<conio.h>
main( )
{
 char name[ ] = "Klinsman" ;
 char *ptr ;
 ptr = name ; /* store base address of string */
 while ( *ptr != `\0' )
 {
  printf ( "%c", *ptr ) ;
  ptr++ ;
 }
  getche();
}
```

And here is the output...

Klinsman

Strings

- printf( ) function has got a sweet and simple way of doing it, as shown below.
  - Note that printf( ) doesn't print the '\0'.
  - The %s used in printf( ) is a format specification for printing out a string.

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

Strings

- 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 ( "Hello %s!", name );
}
```

Strings

- scanf( ) is not capable of receiving multi-word strings.
  - Therefore names such as 'Abdullah Sikandar' would be unacceptable.
  - The way to get around this limitation is by using the function gets( ) and puts( ).

```
main( )
{
 char name[25];
 printf ( "Enter your full name " );
 gets ( name );
 puts ( "Hello!" );
 puts ( name );
}
```

Standard Library String Functions

Function	Use
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)

string.h is used as a header file

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 string

strlen( )

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main( )
{
 char arr[ ] = "jadoon" ;
 int len1, len2 ;
 len1 = strlen ( arr ) ;
 len2 = strlen ( "Abdullah Sikandar" ) ;
 printf ( "\nString = %s length = %d", arr, len1 ) ;
 printf ( "\nString = %s length = %d", "Abdullah Sikandar", len2 ) ;
 getche();
}
```

Output is as under!!!

String = jadoon length = 6
String = Abdullah Sikandar length = 17

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.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main( )
{
 char source[ ] = "Abdullah" ;
 char target[20] ;
 strcpy ( target, source ) ;
 printf ( "\nSource String = %s", source ) ;
 printf ( "\nTarget String = %s", target ) ;
 getche();
}
```

Output!!!
Source String = Abdullah
Target String = Abdullah

strcat( )

- This function concatenates the source string at the end of the target string.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main( )
{
 char source[ ] = "Sikandar!" ;
 char target[30] = "Abdullah" ;
 strcat ( target, source ) ;
 printf ( "\nSource String = %s", source ) ;
 printf ( "\nTarget String = %s", target ) ;
 getche();
}
```

Output !!!

Source String = Sikandar!
Target String = AbdullahSikandar!

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 (strcmp)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main( )
{
 char string1[ ] = "Jerry" ;
 char string2[ ] = "Ferry" ;
 int i, j, k ;
 i = strcmp ( string1, "Jerry" ) ;
 j = strcmp ( string1, string2 ) ;
 k = strcmp ( string1, "Jerry boy" ) ;
 printf ( "\n%d %d %d", i, j, k ) ;
 getche();
}
```

The output!!!

0 4 -32

Output illustration

- In the first call to strcmp( ),
  - the two strings are identical—"Jerry" and "Jerry"—and the value returned by strcmp( )is zero.
- In the second call,
  - the first character of "Jerry" doesn't match with the first character of "Ferry" and the result is 4, which is the numeric difference between ASCII value of 'J' and ASCII value of 'F'.
- In the third call to strcmp( )
  - "Jerry" doesn't match with "Jerry boy", because the null character at the end of "Jerry" doesn't match the blank in "Jerry boy".
  - The value returned is -32, which is the value of null character minus the ASCII value of space, i.e., '\0' minus ' ', which is equal to -32.


Thank You!