String in C
Computer Programming (CS1201)

Course Instructor:

Dr. Ritesh Kumar

Department of Computer Science and Engineering

National Institute of Technology, Jamshedpur, Jharkhand

Topics Covered Are:

- Introduction
- Declaration of strings
- Initialization of strings
- Reading strings
- Writing strings
- Examples

1. Intoduction

- String is sequence or collection of characters terminated by null character. Null terminates the string but not part of it. Strings are accessed through arrays/ pointer .
- String.h contains prototypes of many useful functions.

1.1 Features of strings

- C has no native string type, instead we use arrays of char.
- A special character, called a “null”, marks the end. This may be written as ’\0’.
- This is the only character whose ASCII value is zero.
- Depending on how arrays of characters are built, we may need to add the null by hand , or the compiler may add it for us.

2. Declaration of strings

- The strings can be declared as array of character. The general syntax is:

  char name_of_string[length];

- Char is data type.
- Name_of _string is user defined name given to string variable.
- [length]: defines the size of the string

Example of declaration of strings

- char a[10]; in above example ‘a’ is string variable of length 10 characters.
- char a[]={‘H’, ’E’, ’L’, ’L’ , ’O’, ’\0’,}; In memory the character array or string is

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
	H	E	L	L	O	\0
Mem Add	200	201	202	203	204	205

So total 6 bytes are allocated to the string ‘a’.

Cont…

- Notice that even though there are only five characters in the word ‘HELLO’ , six characters are stored in computer. The last character, the character’/0’ is the NULL character, which indicates the end of string.
- Therefore , if any array of characters is to be used to store a string , the array must be large enough to store the string and its terminating NULL character.

Initialization of strings

- we can initialize string variables at compile time such as: char name[10]="Arris";
- This initialization creates the following spaces in storage:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
A	R	R	i	s	\0	\0	\0	\0	\0

Cont…

Initialization of strings

- if we declare a string like this :

  char my_drink[3]= “tea”; we will get the following syntax error: error c2117:’tea’: array bound overflow .

- instead , we need to at lest declare the array with (the size of string +1) to accommodate the null terminate character’\0’.

  char my_drink[4]=“tea”;

String Input

- Use %s field specification in scanf to read string

  -> ignores leading white space

  -> reads characters until next white space encountered

  -> C stores null (\0) char after last non-white space char

  -> Reads into array (no & before name, array is a pointer)
- Example: char Name[11];

  scanf("%s",Name);
- Problem: no limit on number of characters read (need one for delimiter), if too many characters for array, problems may occur

String Input (cont)

- Can use the width value in the field specification to limit the number of characters read:

  char Name[11];

  scanf("%10s",Name);

- Remember, you need one space for the \0

  ->width should be one less than size of array

- Strings shorter than the field specification are read normally, but C always stops after reading 10 characters

String Output

- Use %s field specification in printf: characters in string printed until \0 encountered

  char Name[10] = “Rich”;

  printf(“|%s|”,Name); /* outputs |Rich| */

- Can use width value to print string in space:
  printf(“|%10s|”,Name); /* outputs | Rich| */
- Use - flag to left justify:

printf(“|-%10s|”,Name); /* outputs |Rich | */

Readind a string

- The drawback of reading a string using scanf() is that it does not read whitespaces or whole line
- If the string to be read as an input has embedded whitespace characters, use standard **gets()** function.

Printing a string

- The drawback of printing a string using printf() is that it does not print whitespaces.
- If the string to be print as an output has embedded whitespace characters, use standard **puts()** function.

Example

```
main()
{
   char s[20];
   clrscr();
   printf(“Enter your name”);
   gets(s);
   puts(s);
}
```

Commonly used functions

Function	Description
Strlen()	To calculate the length of string.
Strcpy()	To copy the contents of one string to another
Strcat()	To concatenate the two strings
Strcmp()	To compare two strings
Strstr()	To find the first occurrence of a substring in another string
Strrev()	Reverse string
Strlwr()	Converts a string to lowercase
Strupr()	Converts the string to uppercase
Strset()	Sets all characters of string to a given number

Program to calculate length of string **without using predefined function**

```
main()
{
char s[20];
int  i:
clrscr();
printf(“Enter a string”);
gets(s); //s==&s[0]
for(i=0;s[i]!=‘\0’;i++);
printf(“length is %d”, i);
getch();
}
```

Program to calculate length of string by **using predefined function**

```
#include<string.h>
main()
{
char s[20];
int  i:
clrscr();
printf(“Enter a string”);
gets(s); //s==&s[0]
i=strlen(s);
//for(i=0;s[i]!=‘\0’;i++);
printf(“length is %d”, i);
getch();
}
```

Thanks