TCS201: Programming and Problem Solving

Strings

Strings:

- String is an array of characters.
- It is terminated by null character '\0'.
- It is usually stored as one-dimensional character array.
- '\0' character is stored to signify the end of the character array, after last character.

For ex:
"HELLO" is a string with 6 characters including the null character stored inside the computer memory.

The above string can be represented in memory as:

0	1	2	3	4	5
'H'	'E'	'L'	'L'	'O'	'\0'

Declaration of a String:

Strings are declared in C using the **char** data type.
For ex:
char name[5];
//characters including the NULL character denoted //as **'\0'**
• The above declaration can be represented as:

name[0]	name[1]	name[2]	name[3]	name[4]
				\0

|← Garbage values →|

Initialization of a String:

For ex:
char name[]={'H','E','L','L','O','\0'};
char s[]="HELLO";

The above declaration can be represented as:

name[0]	name[1]	name[2]	name[3]	name[4]	name[4]
‘H’	‘E’	‘L’	‘L’	‘O’	‘\0’

Q1. Can we declare size of string greater than no. of elements that are unitialized?

Ans. YES.

Char str[10]=”HELLO”;

str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]	str[9]
‘H’	‘E’	‘L’	‘L’	‘O’	‘\0’	‘\0’	‘\0’	‘\0’	‘\0’

Example: Program to demonstrate initialization of a string.

```
#include<stdio.h>
int main()
{
char name[5]={ ‘L’,’I’,’O’,’N’};        //or char s[5]="LION";

printf("Character in the array at First position: %c \n", name[0]);
printf("Character in the array at Second position: %c \n", name[1]);
printf("Character in the array at Third position: %c \n", name[2]);
printf("Character in the array at Fourth position: %c ", name[3]);
return 0;
}
```

Output:

Character in the array at First position: **L**
Character in the array at Second position: **I**
Character in the array at Third position: **O**
Character in the array at Fourth position: **N**

Reading & Printing of Strings:

The strings can be accepted from the user using the following formatted functions:

1. Using Scanf() and printf():

scanf() : to accept the string from the user
printf() : to print a string to the screen

Example: Program to illustrate the use of scanf() and printf().

```
#include<stdio.h>
int main()
{
char str[15];
printf("Enter a string: \n");
scanf("%s",str);
printf("You typed the string:%s", str);
return 0;
}
```

Output 1:

```
Enter a string: HELLO
You typed the string: HELLO
```

Output 2:

```
Enter a string: HELLO WORLD
You typed the string: HELLO
```

Problem with scanf in reading a string and its solution:

Output 2 shows the drawback of using scanf(). scanf can read a string from the input stream until the first occurrence of space. Hence, to enable scanf to read a string until a newline character ie '\n', the following modification can be done to the scanf :

After modifying the scanf statement:

```
#include<stdio.h>
int main()
{
char str[15];
printf("Type a string: \n");
scanf("%[^\n]s", &str);
printf("You typed the string:%s", str);
return 0;
}
```

Output:

```
Type a string:          Graphic Era Hill University
```

You typed the string: **Graphic Era Hill University**

2. Unformatted input/output String functions: gets, puts

gets()--- to read a string from the user until the user enters a newline Character ie '\n' (presses Enter key)
puts()--- to display a string to the screen

Example: Program to illustrate the use of gets() and puts().

```
#include<stdio.h>
int main()
{
char str[15];
printf("Type a string: \n");
gets(str);                                  //same as scanf("%^[\n]s", str);
printf("\nYou typed: ");
puts(str);                                  //same as printf("%s", str);
return 0;
}
```

Output:
Type a string: **Programming in C**
You typed: **Programming in C**

3. Input/output String functions: getchar(), putchar()

getchar()--- repeatedly string is read by getchar() to read sequence of single characters unless terminating character is entered.
putchar()--- to display a characters on the screen.

Example: Program to illustrate the use of getchar() and putchar().

```
#include<stdio.h>
int main()
{
        int i=0,j,k=0;
        char str[20];
        char ch=getchar();
        while(ch!='*')
        {
```

```
            str[i]=ch;
            i++;
            ch=getchar();
        }
    str[i]='\0';
    printf("The entered string is:\n");
    while(str[k]!='\0')
    {
        putchar(str[k]);
        k++;
        }
return 0;
}
```

STRING MANIPULATION FUNCTIONS

• Whenever strings needs to be manipulated in a program manually it adds the extra lines of program code and also makes it a very lengthy and difficult to understand.

• To avoid this C supports a large number of string handling functions. There are many functions defined in <string.h> header file.

Sl. No.	Function Name & its meaning
1	strlen(s1): → Returns the length of the strng s1
2	strcpy(s1,s2) → Copies the string s2 into s1
3	strncpy(s1,s2,n) → Copies first n characters of string s2 into s1
4	strcat(s1,s2) → Concatenates/Joins the string s2 to the end of s1
5	strncat(s1,s2,n) → Concatenates/Joins first n characters of string s2 to the end of s1
6	strcmp(s1,s2) → compares string s1 with s2; if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than zero; otherwise if s1>s2 it returns a value greater than zero.
7	strncmp(s1,s2,n) → compares first n characters of string s1 with s2; if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than

	zero; otherwise if s1>s2 the it returns a value greater than zero.
8	strcmpi(s1,s2) → compares string s1 with s2 by ignoring the case (uppercase or lowercase); if s1 is equal to s2 the return a value zero; if s1<s2 it returns a value less than zero; otherwise if s1>s2 the it returns a value greater than zero.
9	strchr(s1,ch)→ Returns a pointer to the first occurrence of character ch in s1
10	strstr(s1,s2)→ Returns a pointer to the first occurrence of the string s2 in s1
11	strrev(s1)→ Returns the reverse string after reversing the characters of the string s1
12	strtok(s1,delimiter): // splits str into tokens separated by the delimiters. It needs a loop to get all the tokens and it return NULL when there are no more tokens. char *strtok(char str[], const char *delims);

strlen(s1)

• The function calculates & returns the length of a string str passed to it as an argument excluding the null.

Example: Program to illustrate the use of strlen().

```
#include<string.h>
#include<stdio.h>
int main()
{
char str[20];
int len;
printf("Type a string:");
scanf("%[^\n]s",str);
len=strlen(str);
printf("\nLength of the string %s is %d ", str,len);
return 0;
```

```
}
```
Output:
Type a string: **HELLO**
Length of the string HELLO is 5

strcpy(s1,s2)

• This function copies the content of string s2 into another string s2.
- Putting text into a string:
  strcpy(S, "This is String 1.");

- Copying a whole string from S to D:
  strcpy(D, S);

- Copying the tail end of string S to D:
  strcpy(D, &S[8]);

where D is destination and S is source.

• Example: Program to illustrate the use of strcpy().

```
#include<string.h>
#include<stdio.h>
int main()
{
  char s1[20],s2[20];
  printf("Enter A string: ");
  scanf("%[^\n]s",s2);
  strcpy(s1,s2);        //Content of string s2 is copied into string s1
  printf("Copied string:");
  printf("%s",s1);
  return 0;
}
```
Output:
Enter string: **PROGRAMMING IN C**
Copied string: **PROGRAMMING IN C**

strncpy(s1,s2,n)

• This function copies first n characters of s2 into another string s1.
where n is an integer

Assume that the following statement has been executed before each of the remaining code fragments.

- Putting text into the source string:
  strcpy(S, "This is String 1.");

- Copying four characters from the beginning of S to D and placing a null at the end:
  strncpy(D, S, 4);
  D[4] = '\0';

- Copying two characters from the middle of string S to D:
  strncpy(D, &S[5], 2);
  D[2] = '\0';

- Copying the tail end of string S to D:
  strncpy(D, &S[8], 15);

which produces the same result as strcpy(D, &S[8]);

• Example: Program to illustrate the use of strncpy().

```
#include<string.h>
#include<stdio.h>
int main()
{
char s1[20],s2[20];
int n;
printf("Enter a string: ");
scanf("%[^\n]s",s2);
printf("\nHow many characters to be copied:");
scanf("%d",&n);
strncpy(s1,s2,n);          //Content of string s2 is copied into string s1
printf("Copied string: %s",s1);
 return 0;
}
```

Output:

Enter a string: **PROGRAMMING IN C**
How many characters to be copied: **7**
Copied string: **PROGRAM**

strcat(s1,s2)

Joins two strings by copying the string s2 to the end of s1. strcat() is used to concatenate a null-terminated string to end of another string variable.

Example: Program to illustrate the use of strcat().

```
#include <stdio.h>
#include <string.h>
int main()
{
char s1[10], s2[10];
printf("Enter the First String:");
gets(s1);
printf("\n Enter the Second String:");
gets(s2);
strcat(s1,s2);              //concatenates string s1 and s2 stores the final string in s1
printf("\nConcatenated String: ");
puts(s1); // final concatenated string is stored in s1
return 0;
}
```

Output:
Enter the First String: **Graphic**
Enter the Second String: **Era**
Concatenated String: **GraphicEra**

strncat(s1,s2,n)

Joins first **n** characters of **s2** joins with string s1 and stores it into s1.
Example: Program to illustrate the use of strcat().

```
#include <stdio.h>
#include <string.h>
int main()
```

```
{
char s1[10], s2[10];
int n;
printf("Enter the First String:");
gets(s1);
printf("\n Enter the Second String:");
gets(s2);
printf("\nHow many characters needs to be appended:");
scanf("%d",&n);

strncat(s1,s2,n);                    //concatenates n characters of string 2 to string s1
and
                                     //stores the final string in s1
printf("\nConcatenated String: ");
puts(s1);                            // final concatenated string is stored in s1
return 0;
}
```

Output:
Enter the First String: **Tic**
Enter the Second String: **Tac**
How many characters needs to be appended: **2**
Concatenated String: **TicTa**

strcmp(s1,s2)

• This function compares two strings s1 with s2 and
returns 0 if both the strings are equal i.e s1==s2.
returns a value < 0 if s1<s2
returns a value > 0 if s1>s2

strcmp() is used to compare two strings. The strings are compared character by character starting at the characters pointed at by the two pointers. If the strings are identical, the integer value zero (0) is returned. As soon as a difference is found, the comparison is halted and if the ASCII value at the point of difference in the first string is less than that in the second (e.g. 'a' 0x61 vs. 'e' 0x65) a negative value is returned; otherwise, a positive value is returned. Examine the following examples.

```
char s1[25] = "pat";
char s2[25] = "pet";
```

diff will have a negative value after the following statement is executed.
diff = strcmp(s1, s2);

diff will have a positive value after the following statement is executed.
diff = strcmp(s2, s1);

diff will have a value of zero (0) after the execution of the following statement, which compares s1 with itself.

diff = strcmp(s1, s1);

Example-1: Program using strcmp() function.

```
#include <string.h>
#include<stdio.h>
int main()
{
char s1[30],s2[30];
printf("Enter the first string: ");
gets(s1);
printf("Enter the second string: ");
gets(s2);
int diff= strcmp(s1,s2);
printf("%d",diff);
return 0;
}
```

Output 1:
Enter the first string: **Program**
Enter the second string: **program**
0

Output 2:
Enter the first string: **Pat**
Enter the second string: Pet
-1

Example-2: Program using strcmp() function.

```
#include <string.h>
#include<stdio.h>
int main()
{
char s1[30],s2[30];
```

```
 printf("Enter the first string: ");
 gets(s1);
 printf("Enter the second string: ");
 gets(s2);
 if(strcmp(s1,s2)==0)
        printf("Both strings are equal");
 else
        if(strcmp(s1,s2)>0)
        printf("First string is greater than second string");
 else
        printf("First string is less than second string");
return 0;

 }
```

Output:

Enter the first string: **program**
Enter the second string: **Program**
First string is greater than second string

Enter the first string: **RAINBOW**
Enter the second string: **rainbow**
First string is less than second string

strncmp(s1,s2,n)

• This function compares n characters of two strings s1 with s2 and
returns 0 if both the strings are equal i.e s1==s2 or
returns a value < 0 if s1<s2 or
returns a value > 0 if s1>s2

Examine the following examples.
char s1[25] = "pat";
char s2[25] = "pet";
diff will have a negative value after the following statement is executed.
diff = strncmp(s1, s2, 2);
diff will have a positive value after the following statement is executed.
diff = strncmp(s2, s1, 3);
diff will have a value of zero (0) after the following statement.
diff = strncmp(s1, s2, 1);

Example: Program using strncmp() function.

```
#include <string.h>
#include<stdio.h>
int main()
{
char s1[30],s2[30];
int n;
printf("Enter the first string: ");
gets(s1);
printf("Enter the second string: ");
gets(s2);
printf("\nHow many characters to be compared:");
scanf("%d",&n);

if(strncmp(s1,s2,n)==0)
        printf("Both strings are equal");
else
        printf("Strings are unequal");
return 0;
}
```

Output:
Enter the first string: **progRam**
Enter the second string: **program**
How many characters to be compared: **4**
Both strings are equal

strcmpi(s1,s2)

• This function compares two strings s1 with s2 by ignoring the cases (uppercase or lowercase) and
returns 0 if both the strings are equal i.e s1==s2.
returns a value < 0 if s1<s2
returns a value > 0 if s1>s2

Example: Program using strcmpi() function.

```
#include <string.h>
#include<stdio.h>
int main()
{
char s1[30],s2[30];
printf("Enter the first string: ");
```

```
 gets(s1);
 printf("Enter the second string: ");
 gets(s2);
 if(strcmpi(s1,s2)==0)
          printf("Both strings are equal");
 else
         printf("Strings are unequal");
 return 0;
 }
```

Output:
Enter the first string: **program**
Enter the second string: **PROGRAM**
Both strings are equal

strlwr(str):

This function is used to convert uppercase letters into lower case.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[]=”HeLLo”;
  strlwr(str);
  printf(“the string is: %s”, str);
  return 0;
}
```

OUTPUT: the string is: hello

strupr(str):

This function is used to convert lowercase letters into uppercase.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[]=”HeLLo”;
  strupr(str);
  printf(“the string is: %s”, str);
```

```
  return 0;
}
```
OUTPUT: the string is: HELLO

strrev():

function is used to reverse the string.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[]=”HELLO”;
  strrev(str);
  printf(“the string after reverse is: %s”, str);
  return 0;
}
```
OUTPUT: the string is: OLLEH

Programs on strings without using inbuilt functions:

1. Program to copy one string into another without using the inbuilt function.

```
#include<stdio.h>
int main()
{
        char s1[100], s2[100];
        int i;
        printf("Type a string:");
        gets(s1);
        i = 0;
        while (s1[i] != '\0')
        {
                s2[i] = s1[i];
                i++;
        }
        s2[i] = '\0';
        printf("Copied String is: %s ", s2);
        return 0;
}
```
Output:

Type a string : **Blueberry**
Copied String is: **Blueberry**

2. Program to compare 2 strings without using strcmp().

```
#include<stdio.h>
int main()
{
char s1[25],s2[25];
int i=0,flag=0;
printf("Enter the first string: ");
gets(s1);
printf("Enter the second string: ");
gets(s2);
while(s1[i]!='\0' && s2[i]!='\0')
{
        if(s1[i]!=s2[i])
        {       flag=1;
                break;
        }
         i++;
}
if (flag==0)
         printf("Both the strings are equal");
else
         printf("Both the strings are not equal");
return 0;
}
```

Output:
Enter first string: Graphic
Enter second string: Graphic
Both the strings are equal

3. Example: Program to concatenate two strings without using inbuilt function.

```
#include<stdio.h>
int main()
{
char s1[25],s2[25];
int i=0,j=0;
printf(" Enter the First String:");
```

```
gets(str1);
printf("\n Enter the Second String:");
gets(str2);

while(s1[i]!='\0')
        i++;
s1[i++]=' ';
while(s2[j]!='\0')
{
        s1[i]=s2[j];
        j++;
        i++;
}
s1[i]='\0';
printf("\n Concatenated String is %s",s1);
return 0;
}
```

Output:

Enter the First String: **Merry**

Enter the Second String: **goround**

Concatenated String is **Merry goround**

POINTERS AND STRINGS:

Program 1:

```
#include<stdio.h>
int main()
{
char str[]="HELLO";
char str1[10];
//str=str1; //error
char *p=str;
char *q;
q=p;
while(*q!='\0')
{
printf("%p\t%c\n",(void *)q,*q);
q++;

}
```

```
}
```
OUTPUT:
```
0x7ffe4c12674e    H
0x7ffe4c12674f    E
0x7ffe4c126750    L
0x7ffe4c126751    L
0x7ffe4c126752    O
```

// Demo Code copy text using pointers

```
#include<stdio.h>
int main()
{
 char name1[100] = "GEHU";   char name2[100];
 char *ptr1 ;
 char *ptr2 ;

 ptr1 = name1 ;
 ptr2 = name2 ;

 while (*ptr1 != '\0')
 {
 *ptr2 = *ptr1 ;
 ptr1++ ;
 ptr2++ ;
 }
 *ptr2 = '\0';
 printf("COPIED String is...\n%s",  name2) ;
}
```

OUTPUT:
```
COPIED String is...
GEHU
```

PASSING STRING TO FUNCTIONS USING POINTERS:

Program 1: User Define Function to Copy a String:

```
#include<stdio.h>
void display(char*);
int main()
```

```
 {
 char str[10];
 gets(str);
 display(str); //can be &str, &str[0]
 return 0;
 }
 void display(char *p)
 {
 while(*p!='\0')
 {
 printf("%c",*p);
 p++;
 }
 }
```

Program 2: User Define Function to Compare two Strings

```
// Case Sensitive Version
#include <stdio.h>

int compare(char *, char *) ;

int main()
{
   char org[50] = "Arjun";
   char copy[50]        ;

   printf("Enter the second String to Compare\n");
   scanf("%s", copy);

   if (compare(org,copy) == 0)
     printf("The two Strings are Equal \n");
   else
     printf("The Two Strings are Un-equal \n");

   return 0 ;

}

int compare(char *org, char *copy)
{
   int i = 0 ;
```

```
   while ( (*org == *copy) && (*org != '\0' && *copy != '\0'))
     {
      org++  ;
      copy++ ;
     }

   if (*org == '\0' && *copy == '\0')
     return 0 ;
   else
     return -1 ;
}

// Sample Output
Enter the second String to Compare
Arjun
The two Strings are Equal

Enter the second String to Compare
arjun
The Two Strings are Unequal
```

Two Dimensional Array of Characters:

```
#include <stdio.h>
int main()
{
 char  months [12][25] =
{"January","February","March","April","May","June","July","August",
"September","October","November",  "December"} ;

char *dow[] =
  {"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday" };

printf("Months of a Year\n") ;
for (int i = 0; i < 12; i++)
printf("%s\n", months[i]);
printf("Days of the Week\n") ;
for (int i = 0; i < 7; i++)
printf("%s\n", dow[i]);
return 0;
```

```
}
```

strtok():

The C library function char *strtok(char *str, const char *delim) breaks string str into a series of tokens using the delimiter delim.

- Following is the declaration for strtok() function:

char *strtok(char *str, const char *delim)

- Parameters:

str − The contents of this string are modified and broken into smaller strings (tokens).

delim − This is the C string containing the delimiters. These may vary from one call to another.

- Return Value:

This function returns a pointer to the first token found in the string. A null pointer is returned if there are no tokens left to retrieve.

The following example shows the usage of strtok() function:

```
#include <string.h>
#include <stdio.h>

int main () {
   char str[80] = "Graphic-Era-Hill-University";
   const char s[2] = "-";
   char *token;

   /* get the first token */
   token = strtok(str, s);

   /* walk through other tokens */
   while( token != NULL ) {
      printf( " %s\n", token );

      token = strtok(NULL, s);
   }
```

```
  return(0);
}
```

OUTPUT:

```
Graphic
Era
Hill
University
```

atoi():

The C library function int atoi(const char *str) converts the string argument str to an integer (type int).

- atoi() converts a given string into its corresponding integer.
- This function returns that integer to the calling function.
- Therefor string should start with a number.
- This function will stop reading from the string as it gets a non-numerical character.

Syntax: int atoi(const char*str);

The following example shows the usage of atoi() function:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   char str[10]="12345";
   char str1[10]="12345.57";
   char str2[10]="12345a678";
   int i=atoi("12");
   int a=atoi(str);
   int b=atoi(str1);
   int d=atoi(str2);
   printf("%d\t",i);        //12
   printf("%d\t",a);        //12345
   printf("%d\t",b+5);      //12350
   printf("%d\t",d);        //12345

}
```