Jai Sri Gurudev

Module-3

Swetha 7
Swetha K.R
Dept of CS

Strings

→ A string is a sequence of characters enclosed with double quotes.

A string constant is always terminated by a character ['\0'].

eg:- string is "BGS"

```
#include<std
void main()
{
Printf("BG
}
```

B	G	S	\0
0	1	2	3

→ NULL-character.

Array starts from zero and each locations hold one character starting from index 0 to 2. The string always ends with NULL character denoted by '\0'.

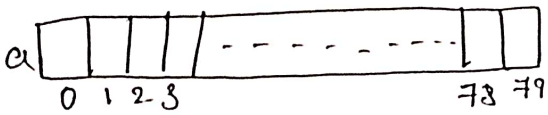
Declaring a string Variable :-

String variable has to be declared before it is used in any of the functions.

→ string is declared as an array of characters.

char a[80];

using this declaration, the compiler allocates 80 memory locations for the variable a ranging from 0 to 79 as shown below.



Initializing strings :-

Initialization is the process of assigning value to a variable before doing manipulation.

Initialization can be done in various ways

1) Initializing locations character by character
2) Partial array Initialization

4) Array initialization with a string.

Initializing locations character by character :-

Here, compiler will allocate all memory location with a character and last location with '\0'. i.e.

eg:- Char bgsit = {'C','O','M','P','U','T','E','R'}

bgsit	C	O	M	P	U	T	E	R	\0
	0	1	2	3	4	5	6	7	8

Partial array initialization :-

If the number of characters to be initialized is less than the size of the array then all characters are stored sequentially from left to right. The remaining locations will be initialized to NULL character automatically.

eg:- char a[6] = {'p','c','d'}

a	p	c	d	\0	\0	\0
	0	1	2	3	4	5

Initialization without size :-

If the size is not specified, then the compiler will set the array size to the total number of initial values i.e

eg:- char str[] = {'p','c','d'}

str	p	c	d
	0	1	2

For this declaration, the compiler will set the array size to total number of initial values i.e 3.

Array Initialization with a string :-

Consider the following declaration with string initialized

```
char b[]="computer";
```

Here the string length is 8 bytes. So, the compile reserves 8+1 memory locations, i.e.

b	c	o	m	p	u	t	e	r	\0
	0	1	2	3	4	5	6	7	8

String input/output functions :-

The various input and output functions are:-

I/O →
- Formatted Input function — eg:- scanf()
- Formatted output function — eg:- printf()
- Un-formatted Input function — eg:- gets()
- Un-formatted output functions — eg:- puts()

1) **Formatted input function - scanf() :-**

Consider the declaration char bgs[100];

→ The string can be read from using scanf() by writing the following conversion code %s

```
scanf("%s", bgs);
```

→ The main drawback of this function is that it terminated as soon as it finds a blank space.

→ If white space (blank space) are present

eg eg:- If the user enters hello world, then bgs will contain only hello.. i.e

bgs

h	e	l	l	o	\0	?	?	..	?
0	1	2	3	4	5	6	7	...	99

→ All other subsequent characters are still in the keyboard buffer. All these characters can be removed using fflush(stdin)

2) Unformatted input function :- gets(), getch(), getche()

Refer module 1 notes for gets(), getch(), getche().

3) Formatted output function like printf() :-

→ The string can be displayed using printf() by writing printf("%s", bgs);

→ we may also use width and precision specifications along with %s

→ The width specifies the minimum output field width. If the string is short, extra space is either left padded or right padded.

→ The precision specifies the maximum number of characters to be displayed.

→ The -ve width left pads short strings rather tho the default right justification. If the string is long the extra characters are truncated. For example.

printf("%5.3s", bgs);

		h	e	l

→ This statement would print only the first 3 charact in a total field of 5 characters. all these characte right justified in the allocated width. To ma

eg:- printf("%-5.3s", bgs)

H	e	l		

4) Unformatted output functions :- puts(), putch(), Putche()

refer module 1 notes for puts, putch(), putche() functions.

String handling functions :-

String manipulation functions :-

1) strlen (str) - string length :-

It is defined in header file "string.h"

Syntax :- int strlen (char str[]);

This function returns the length of the string str i.e it counts all the characters upto '\0' except '\0'

eg:-

```
#include <stdio.h>
#include <string.h>
Void main()
{
  char str[] = "Hello";
  Printf("Length = %d \n", strlen(str));
}
```

output :-

Length = 5

H	e	l	l	o	\0
0	1	2	3	4	5

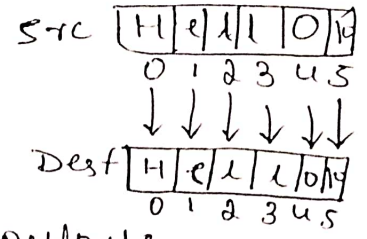
2) strcpy (dest, src) - string copy :-

Syntax :- strcpy (char dest[], char src[]);

→ The strcpy function copies the source string src to destination string dest including '\0'.

→ The size of destination string dest should be greater ~~than~~ or equal to the size of source string src. After copying, the original contents of destination string dest are lost.

eg:-
```
#include <stdio.h>
#include <string.h>
Void main ()
{
  Char src[] = "Hello";
  Char dest[6];
  Strcpy (dest, src);
  Printf("Destination string = %s", dest);
}
```



Output:-
Destination string = Hello

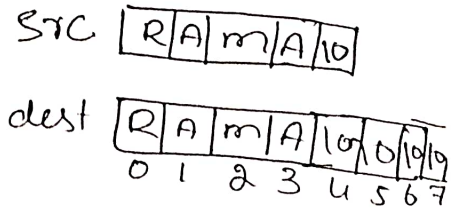
→ strncpy (dest, src, n) – string number copy :-

Syntax:- strncpy (char dest[], char src[], int n);

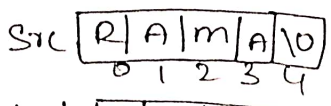
where
- dest is the destination string
- src is the source string
- n indicates number of characters copied from source to destination

→ This function will copy "n" characters from source to destination string as follows
→ If source string is less than n, the entire string is copied to destination and null character is placed at end.
→ If source string src is longer than n characters only n characters are copied into destination. Then destination string will not have a delimiter "\0" and it is invalid string.

eg:- 1) char src[5] = "RAMA";
char dest[8];
Strncpy (dest, src, 5);



2) char src[5] = "RAMA";
char dest[3];
Strncpy (dest, src, 4);



dest R A m → Invalid dest because no \0
0 1 2

5) **Strcat (s1, s2) – String concatenation :-**

Syntax :- strcat (char s1[], char s2[]);

where
- s1 is the first string.
- s2 is the second string.

→ This functions copies all characters from s2 to the end of string s1. The delimiter of string s1 is replaced by the first character of s2. only condition is that of size of s1 is sufficiently large to store a string of size s1 + s2

eg :-
```
# include <stdio.h>
# include < string.h>
Void main ()
{
  char s1[20] = "RAma";
  char s2 [15] = " krishna";
 if(strlen(s1) + strlen(s2) < size of (s1))
    strcat (s1, s2);
 else
 Printf (" error \n");
}
```

output :-

s1	R	A	m	A	k	r	i	s	h	n	a
	0	1	2	3	4	5	6	7	8	9	10

6) **Strncat (s1, s2, n) → string – number concatenate :-**

Syntax :-

strncat (char s1[], char s2[], int n);

where
- s1 is the first string.
- s2 is the second string.
- n is the number of characters of string s2 to be concatinated.

This function will copy only n bytes or n characters

char S1[10] = "VIVEK"; char S2[8] = "RAMAPPA"

eg:- S1

V	I	V	E	K	\0	...	\0
0	1	2	3	4	5	....	9.

S2

R	A	M	A	P	P	A	\0
0	1	2	3	4	5	6	7

After strncat (S1, S2, 4);

S1

V	I	V	E	K	R	A	M	A	\0
0	1	2	3	4	5	6	7	8	9

S2

R	A	M	A	P	P	A	\0

7) **strcmp (S1, S2) - string compare :-**

Syntax:- int strcmp (char S1[], char S2[]);

Where
- S1 is the first string
- S2 is the second string

This function is used to compare 2 strings. The comparision starts with first character of each string. The comparision continues till the corresponding characters differ or until the end of the character is reached.

The following values are returned after comparision

1) if S1 == S2, it returns 0.
2) if S1 > S2, it returns +ve value.
3) if S1 < S2, it returns -ve value.

eg:-
```
#include<stdio.h>
#include<string.h>
int main ()
{
    char S1[] = "RAMA";
    char S2[] = "KRISHNA";
    char S3[] = "RAMA";
    char S4[] = "APPLE";
```

```
int res1, res2, res3;
res1 = strcmp (s1, s2);
res2 = strcmp( s1, s3);
res3 = strcmp ( s4, s1);
```

// s1[0] # s2[1]
82 - 75 = 7

// s1[A] # s2[e]
65 - 82 = 17

```
Printf(" %d \n %d \n %d \n", res1, res2, res3);
}
```

output
7
0
-17

strncmp (s1, s2, n) — string number compare :-

Syntax :- int strcmp (char s1[], char s2[], int n);

* This function is used to compare first n number of characters in the two strings. The comparision, starts with first characters & ends when corresponding character differ, 'n' number of characters is tested or when the end of string is encountered.
* The function returns following values.
  - if s1 == s2, it returns 0.
  - if s1 > s2, it returns +ve value.
  - if s1 < s2, it returns -ve value.

eg:- s1 = BHIKA s2 = BHIHI

strncmp (s1, s2, 4) returns 0
strncmp (s1, s2, 5) returns -ve value.
strncmp (s1, s2, 3) returns 0

8) strspn (S1, S2) → string span :-

Syntax :- strspn (char S1[], char S2[]);

* The function keeps comparing each character of string 1 i.e S1 with string S2. If the character of S1 is present in S2 comparision continues with next character until a different character is encountered which is not present in S2, and returns following values.

- returns number of character S1 matched with S2.
- returns 0 (zero) if no character is matched.

eg:- strspn (str1, str2) return 6 as follows:

str1

B	H	I	S	H	M	A	O	H	A	R	I	\0
0	1	2	3	4	5	6	7	8	9	10	11	12

str2

I	S	H	B	H	\0

→ A is character in str1, which is not present in str2, so comparision stops. and returns 6 i.e no. of characters present in str1 scanned successfully.

9) strcspn (S1, S2) → string complement span :-

* This function is complement of strspn. This function keeps comparing character of string S1 from left to right with contents of S2 as long as the character is not available in S2.

* Function returns following values :-
  - Returns the number of characters ~~passed~~ scanned in S1 which is not present in S2.
  - Returns the length of string S1 if no character of

eg:- S1 = "BHIMA", S2 = "VASU", S3 = "GOD"

Strcspn(S1, S2) = 4.

Strcspn(S1, S3) = 5.

10) **Strrev (str) – string reverse :-**

This function reverse all the characters present in str except '\0'. The original string str is lost

syntax:- void strrev (char str[]).

eg:- Check whether the given string is palindrome or n

```
#include <stdio.h>
# include <string.h>
main ()
{
    char s1[] = "INDIA";
    char s2[];
    Strcpy (s2, s1);
    Strrev (s1);
    if (strcmp (s1, s2) == 0)
    {
      Printf ("given string is palindrome \n");
    }
    else
    {
      Printf("given string is not a palindrome\n");
    }
}.
```

Output :-
given string is not a palindrome

▷ **Strlwr (str)**

This function converts all characters in string str to lower case.

eg:- char str[] = "RAMA";
Strlwr (str);

str

R	A	M	A	\0

12) strupr(char str[]);

This function converts all characters in the string str to uppercase.

eg:- char str[] = "apple"

strupr(str);

str:

a	p	p	l	e	\0

output:

A	P	P	L	E	\0

3) strchr (str, c);

This function returns the first occurance of character c in the string str.

eg:- char str[] = "APPLE";

strchr (str, 'L'); = 4

A	P	P	L	E	\0
0	1	2	3	4	5

4) strrchr (str, c);

This function returns the last occurance of character c in the string str.

eg:- char str[] = "APPLE";

strchr (str, 'P'); = 3.

5) strstr (str1, str2);

This function finds the first occurance of str2 in str1.

eg:- char str1[] = "single";

char str2[] = "sing";

strstr (str1, str2); = 1

Programming examples.

1) Binary search -> Refer lab program.

2) No of vowels & consonants -> Refer lab program

3) Copy the src string to destination -> Refer lab program