PPS:

Page No.
Date

Unit - 4

Strings

⇒ Topic Name:-

- String
- String Storage
- Built-in- String functions

Prepared By:-
Ms. Jindl Jala
(MECE), LJIET

↦ Introduction:-

Def^{n}:- A string is a sequence of characters that is treated as a single data item.

↦ Any group of characters or string defined bet^{n} double quotation marks.

Example:- " Well done "

Printf c" Well done "c);

↦ Character strings are often used to build meaningful & readable programs.

↦ Different operations on strings:-

- Reading & writing strings
- Combining strings together
- Copying one string to another
- Comparing strings for quality
- Extracting a portion of a string

↦ Declaring & Initializing String Variables:-

↦ C doesn't support strings as a data-type. However, it allows us to represent strings as character arrays.

↦ The general form of declaration of a string variable is :-

```
char string-name [size];   ← Declaration
```

Ex :-
```
char city [10];
char name [30];
```

↦ Remember :- When the compiler assigns a character string to a character array, it automatically supplies a null character ('\0') at the end of the string. Therefore, the size should be equal to the maximum number of characters in the string plus one.

↳ Initialization :-

C permits a character array to be initialized in either of the following two forms :-

```
char city[9] = "New York";
char city[9] = {'N','e','w',' ','Y','o','r','k'};
```

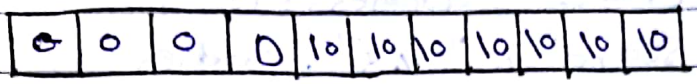
C also permits us to initialize a character array without specifying the number of elements.

```
char string[] = {'G','o','o','d','\0'};
```

⇒ Rules for string:-

1. We can declare the size much larger than the string size in the initializer.

```
Char str[10] = "Good"
```



2. The following declaration is illegal.

```
Char str[3] = "Good"
```

Because - Good \0

size 5 elements & str[3] = 4 elements

This will result in a compile time error.

3. We can't separate the initialization from declaration.

```
Char str[5];
str = "Good";
```

is not allowed.

4. An array name can't be used as the left operand of an assignment operator.

```
Char S1[4] = "abc";
char S2[4];
S2 = S1;   (// Error)
```

Not allowed.

⇒ Reading Strings from Terminal:-

(i) Using Scanf function:-

- Scanf can be used with %s format.

```
Char address[10];
Scanf("%s", address);
```

Remember:- The problem with the Scanf function is that it terminates its input on the first white space it finds.

Ex:- If we enter the string

New York

then only the string "New" will be read into the array address, since the blank space after the word 'New' will terminate the reading of string.

N	e	w	\0	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9

The unused locations are filled with garbage.

⇒ Example:-

1. Write a program to illustrate how to read string & write on the screen.

```
void main()
{
    char name[20];
    clrscr();
    printf("Enter name:");
    scanf("%s", name);
    printf("Your name is %s", name);
    getch();
}
```

O/P :- Enter name : Jinal
Your name is Jinal.

(ii) Using getchar function :-

↦ getchar() function is mostly used for reading single character if we want to read multiple character using getchar() function then we can do it with using loop.

↦ Example :-

```
void main()
{
    char line[80], character;
    int c=0;
    printf("Enter text:");
    while(character != '\n')
    {
        character = getchar();
        line[c] = character;
        c++;
    }
```

```
    c = c-1;
    line[c] = '\0';
    printf("\n The text entered by You:\n%s", line);
    getch();
}
```

(iii) Using gets function :-

↦ Another & more convenient method of reading a string of text containing white space is to use the library function gets available in the <stdio.h> header file.

```
char line[80];
gets(line);
printf("%s", line);
```

↦ Writing Strings to Screen :-

(i) Using Printf function :-

↦ We have extensively the printf function with %s format to print string to the screen.

```
printf("%s", name);
```

(ii) Using Putchar & puts functions:-

↦ Like getchar, C supports another character handling function putchar to o/p the values of character variable.

```
Char ch = 'A';
putchar(ch);
```

↳ Example :-

```
Char name[6] = "PARIS";
for(i=0; i<5; i++)
{
   Putchar(name[i]);
}
```

↳ ~~Putchar~~ Puts() :-

Another & more convenient way of Printing String values is to use the function puts declared in the header file <stdio.h>.

```
puts(str);
```

↳ Example :-

```
char line[80];
gets(line);
puts(line);
```

⇒ **String - Handling functions (Built-in String functions) :-**

↦ C library supports a large number of string-handling functions that can be used to carry out many of the string manipulations.

	Function		Action
1.	strcat()	-	concatenates two strings
2.	strcmp()	-	Compares two things
3.	strcpy()	-	Copies one string over another
4.	strlen()	-	finds the length of a string

1. Strcat( ) function:-

↦ The Strcat function joins two strings together.

```
Strcat(string1, string2);
```

↦ String1 & String2 are character arrays. When the function strcat is executed, String2 is appended to String1.

↳ Example :-

Part1 =

0	1	2	3	4	5	6	7	8	9	0	1
V	E	R	Y		\0						

Part2 =

0	1	2	3	4	5	6
G	o	o	D	\0		

After execution of the statement

```
Strcat(part1, Part2);
```

will result in:

Part1 =

0	1	2	3	4	5	6	7	8	9	0	1	2
V	E	R	Y		G	o	o	D	\0			

Part2 =

0	1	2	3	4	5	6
G	o	o	D	\0		

Another method Strcat function may also append a string constant to a string variable.

```
Strcat(Part1, "Good");
```

2. Strcmp() function:-

↦ The Strcmp function compares two strings identified by the arguments & has a value 0 if they are equal. If they are not, it has the numeric difference betn the first nonmatching characters in the strings.

```
strcmp (String1, String2);
```

↳ Example:-

```
Strcmp(name1, name2);
strcmp(name1, "John");
strcmp("Rom", "Ram");
```

Suppose we take ASCII "i" and ASCII "r" then i - r = -9 is difference

105 - 114 = -9

3. Strcpy() function:-

↦ The strcpy function works almost like a string-assignment operator.

```
strcpy (String1, String2);
```

↳ Example:-

```
strcpy (city, "DELHI");
strcpy (city1, city2);
```

(will assign the contents of the string city2 to city1).

4. Strlen() function :-

↦ This function counts & returns the number of characters in a string.

```
n = strlen(string);
```

↦ where n is an integer variable, which receives the value of the length of the string.
The argument may be a string constant, The counting ends at the first null character.

↳ Examples :-

1. WAP to read string & find string length with using inbuilt function & without using inbuilt function.

↳
```
void main()
{
    int len = 0;
    char s[10];
    printf("Enter a string:");
    gets(s);
    while(s[len] != '\0')
    {
        len++;
    }
printf("Length : %d", len);
getch();
}
```

```
Void main()
{
    int n;
    char s[10];
    printf("Enter a string:");
    gets(s);
    n = strlen(s);   ← Inbuilt function
    printf("Length: %d", n);
    getch();
}
```

2. WAP to count total words in the given string.

```
Void main()
{
    int len = 0, c=1;
    char s[40];
    printf("Enter a string:");
    gets(s);
    while(s[len] != '\0')
    {
        if(s[len] == ' ')
        {c++;
        }
        len++;
    }
    printf("Total Words: %d", c);
    getch();
}
```

3. WAP to count no. of vowels in given string.

↦
```
void main()
{
    char s[80];
    int i, vowels = 0;

    Printf("Enter a sentence\n");
    gets(s);
    for(i=0; s[i]!='\0'; i++)
    {
      if((s[i]=='a'||s[i]=='e'||s[i]=='i'
    ||s[i]=='o'||s[i]=='u')||(s[i]=='A'
    ||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='U'))
        {
            vowels++;
        }
    Printf("No of vowels in sentence
            = %d", vowels);
    getch();
}
```

4. WAP to append one string to another.

↦
```
Void main()
{
    char s1[25], s2[25];
    int i=0, j=0;
    Printf("Enter first string:");
    gets(s1);
    Printf("\n Enter second string:");
    gets(s2);
```

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

WAP to compare two strings & check that both are equal or not.

```
Void main()
{
    Char str1[20] = "Beginners Book";
    char str2[20] = "BeginnersBook.com";
    if ( Strcmp (Str1, Str2) == 0 )
    {
      Printf ("String 1 and string 2 are
      equal");
    }
    else
    {
      Printf ("string 1 and string 2 are
          different");
    }
    getch();
}
```

WAP to concat two different string using String concat() function.

```
Void main()
{
    Char S1[10] = "Hello";
    char S2[10] = "World";
    strcat (S1, S2);
    Printf ("Output string after concatenation
                : %s", S1);
    getch();
}
```

⇒ CAP to that read two string & compare them using the function strcmp() & print the message that first string is equal, less, greater than the second one.

```
Void main()
{
    char S1[20], S2[20];
    pfc"Enter string 1:");
    gets(S1);
    pfc"Enter string 2:");
    gets(S2);

    if(strcmp(S1,S2)==0)
    pfc"Both strings are equal");
    else if(strcmp(S1,S2)>0)
    pfc"first string is greater than second");
    else
    pfc"Second string is greater than first");
    getch();
}
```

O/P:- Enter string 1: Ramesh
Enter string 2: Rakesh
First string is greater than second

Enter string 1: Rakesh
Enter string 2: Ramesh
Second string is greater than first.

Rakesh Rakesh
Both strings are equal.

Page No
Date

⊟ WAP which accepts a long string & print only characters at positions which are multiple of 3.

⊟
```
Void main()
{
    Char S[100];
    int i;
    pfc"Enter string:");
    gets(s);

    for(i=0; S[i]!='\0'; i++)
    {
        if(i%3==0)
        pfc"%c\t", S[i]);
    }

    getch();
}
```

O/P:- Enter string: We are Engineers

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

W a g e s

space

⊟ WAP to accept a string & print every third character from string only if it is lower case.

```
void main()
{
```

```
char s[100];
int i=0;

Pfc"Enter string:");
getscs);

while (s[i]!='\0')
{
    if(s[i]>='a' && s[i]<='z')
    pfc"%c\t", s[i]);

    i= i+3;
}
getch();
}
```

o/p:- Enter string:

We are ENGINEERS and we can do everything
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18 20 21 23 24 25 27 28 ....

read....

⇒ WAP in C to reverse the input string. (with using inbuilt function)

```
void main()
{
    char s[100];
    pf c"Enter string:");
    getscs);
```

```
    strrev(s);  ← Inbuilt function
    printf("After using function:");
    puts(s);
    getch();
}
```

O/P :- Enter string:- abc
After using function: cba

WAP to print reverse string of given string. (without using inbuilt function)

```
void main()
{   char s[10], rev[10];
    int i, j, l=0;
    printf("Enter the string");
    gets(s);
    while(s[i]!='\0')
    {   l++;  }
    for(j=l-1; i=0; j>=0; j--, i++)
    {
        rev[i]=s[j];
    }
    rev[i]='\0';
    printf("%s", rev);
    getch();
}
```

⇒ WAP convert characters into Toggle character into given String.

```
⇒ Void main()
{
    char str[100];
    int i=0
    pf("Enter string:");
    gets(str);

    while (str[i]!='\0')
    {
        if (str[i]>=65 && str[i]<=90)
        {
            str[i] = str[i]+32;
        }
        else if(str[i]>=97 && str[i]<=122)
        {
            str[i] = str[i]-32;
        }
        i++;
    }
pf(" The toggled string is:");
puts(str);
getch(); }
```

8. WAP to copy one string into another using strcpy() function.

↦
```
Void main()
{
    Char S1[20] = "ABC";
    Char S2[20] = "XYZ";
    Strcpy(S1, S2); // Copied S2 into S1.
    Printf("String S1 is %S", S1);
    getch();
}
```

O/P:-

String S1 is XYZ

↳ Other String functions:-

↦ The header file <string.h> contains many more string manipulation functions.

(i) strncpy():-

↦ 'strncpy' copies only the left-most n characters of the source string to the target string variable.

```
strncpy(S1, S2, 5);
```

↦ This statement copies the first 6 characters of the S2 into S1. 6th position of S2 is '\0'.

S1[6] = '\0'.

(ii) strncmp() :-

↦ A variation of the function strcmp is the function strncmp.

```
strncmp(s1, s2, n);
```

this compares the left-most n characters of s1 to s2 & returns.

(a) 0 if they are equal.
(b) negative number, if s1 sub-string is less than s2
(c) Positive number, otherwise.

(iii) strncat() :-

```
strcat(s1, s2, n);
```

↦ This call will concatenate the left-most n characters of s2 to the end of s1.

(iv) strstr() :-

↦ It is a two-parameter function that can be used to locate a sub-string in a string.

```
strstr(s1, s2);
str str(s1, "ABC");
```

↦ The function strstr searches the string S1 to see whether the string S2 is contained in S1. If yes, the function returns the position of the first occurrence of the sub-string.

↦ We also have functions to determine the existence of a character in a string.

```
strchr(S1, 'm');
```

will locate the first occurrence of the character 'm' and the call

```
Strrchr(S1, 'm');
```

will locate the last occurrence of the character 'm' in the string S1.

9. Write a program to check whether given String is Palindrome or not.

↦
```
Void main()
{
    Char S[20];
    int i, length, flag=0;
    printf("Enter a string:");
    gets(S);
    length = strlen(S);
    for(i=0; i<length; i++)
    {
        if(S[i] != S[length-i-1])
        {   flag=1;
            break;
        }
    }
}
```

```
if (flag == 1)
{
  printf("%s is not palindrome", s);
}
else
{
  printf("%s is a palindrome", s);
}
getch();
}
```

⇒ Find given string is Palindrome or not using string library function.

```
Void main()
{
    char str[100], rev[100];

    Pfc"Enter string:");
    getsc str);
    strcpy (rev, str);          // Copied one string
    str.rev(str);               // Revesse one string

    if (Strcmp(str, rev)==0) // Compare Both Str,
    Pfc"\nThe string is Palindrome");
    else
    Pfc"\nString is not Palindrome");
    getch();
}
```

O/P :- Enter string: madam
The string is Palindrome

Enter string: abc
String is not Palindrome

Prepared By:-
Ms. Jinal Jala
(MECE), LJIET