Unit 4 Strings and Library Functions

String

"String is an array of character."

A string variable is a variable declared as array of character. The general format of declaring string is:

char string_name [size];

Here string_name is a variable name and size indicates your string contains maximum how many characters. When the compiler assigns the string to the character array, it automatically put the Null Character ('\0') at the end of string. Null character indicates the end of string. So the size of string should be maximum number of characters plus one. For example, in following case we have string name "city" and size of string is "10".

char city[10];

Initialization of string variable can be done by two ways. One is a by simple method which we are using in initialization of array.

char string_name[size] = { list of the character in string within single cot};

For example, in following case string "name" has assigned value "Hello".

char name[10] = {'H', 'e', 'l', 'l', 'o', '\0'};

Another method is in which we assign the string directly within double cot.

char string_name[size] = "String";

For example, in following case we are assigning directly string "Hello" to variable "name".

char name[10] = "Hello";

We can also initialize the string variable without specifying the size of variable. For example,

char name[ ] = "Hello";

Printing string

To print the string we are using same printf function but we have to specify the output format as %s.

printf("%s", string_name);

For example, we have the following string variable "name" having value "Hello World!". Then we are printing the string with printf function.

```
char name[20] = "Hello World!";
printf("%s\n", name);
```

We can use formatting in printing string. In formatting we have to specify the total width (w) and number of characters we want to print from string (p).

%w.ps

For example, in above example of string "Hello World!" we want to specify the total width of print is 10 and from string we want to print string "Hello" then

```
printf("%10.3s", name);
```

We can also use puts( ) function to print the string.

```
puts(string_name);
```

Here there is no need to put the new line character after the string variable but function itself automatically put the new line after the string. For example, in above example if we use puts( ) function then output will be "Hello World!" with new line at end.
puts(name);

Reading String

To read we can use *scanf( )* function. To read string value we have to specify formatting character *%s*. But one thing in string reading, we have to not specify the address operator (&) in *scanf( )* because string variable automatically determine its address. The format for the *scanf( )* function is as under:

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

Before reading string variable by using *scanf( )* function we have to clear the input buffer. For that in Turbo C/C++ we have to use *fflush( )* function.

```
fflush(stdin);          // Clear standard input buffer
```

There is one limitation of string reading by using simple *scanf( )* function. It cannot read white space (spacebar, tab, new line). That means you cannot read entire string "Hello World!" by using simple function since it contains white space but we can read only "Hello".

Solution to read string variable with white space:

1. We have to read the entire string character by character in loop up to we is not entering new line character. At the end of string we have to explicitly put the null character to indicate the end of string.

```
char c, name[100];    int i=0;
do {   c=getchar( ) ;         // getchar() function read one character
       name[i] = c;
       i++;
   } while (c != '\n');
name[i] = '\0';
```

2. We have to use *gets( )* function which read the string up to new line.

```
char name[100];
gets(name);          // this will read the string up to new line
```

3. We can use same *scanf( )* function with formatting to read the string.

```
scanf("%[^\n]", name);
```

Here we have used "%[^character]" format means your computer read the string up to you are not entering the new line character.

String Handling Function

They are

1. strlen()
2. strcat()
3. strcpy()
4. strcmp()
5. strrev()

To use these functions, the header file string.h must be included in the program using statement

[1]. String Length: strlen()

Syntax - int strlen()

Description - It is used to find length of a given string. Length means total number of characters.

Example - Write a program to find length of a string.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char   s1[50];
        int      len;
                 clrscr();
                 printf("Enter your name…");
                 scanf("%s",s1);
                 len  =  strlen(s1) ;
                 printf("\n Length = %d", len);
                 getch();
}
```

Output:

```
Enter your name… Anjali
Length = 6
```

[2]. String Copy: strcpy()

Syntax - void strcpy(s2, s1)
Where s1, s2 both are string.

Description - It is used to copies the contents of one string to another. Here, the contents of s1 are copied into s2. s1 is source string and s2 is destination string.

Example - The following program shows the use of strcpy() function.

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

```
#include<conio.h>
#include<string.h>
void main()
{    char    s1[50], s2[50];
          clrscr();
           printf("Enter string…");
           scanf("%s",s1);
           strcpy(s2,s1) ;
           printf("\n Original string = %s & Copied string = %s", s1,s2);
           getch();
}
```

<u>Output</u>:

```
Enter string… Anjali
Original string = Anjali & Copied string = Anjali
```

[3]. String Concatenation: strcat()

Syntax	-	void strcat(s1, s2) Where s1, s2 both are string.
Description	-	It is used to append the contents of one string to another. Here, the contents of s2 are appended to s1. s1 is resultant string.
Example		

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{           char    s1[50], s2[50];
           clrscr();
           printf("Enter string  s1…");
           scanf("%s",s1);
           printf("Enter string  s2…");
           scanf("%s",s2);
           strcat(s1,s2) ;
           printf("\n Resultant string = %s ", s1);
           getch();
}
```

<u>Output</u>:

```
Enter string  s1… Tushar
Enter string s2… Patel
Resultant string = TusharPatel
```

[4]. String Comparision: strcmp()

Syntax	-	int strcmp(s1, s2) Where s1 and s2 are string.
Description	-	It is used to compares two string. It returns 0 (zero) value if both s1 equal to s2, positive value if s1 greater than s2 and negative value if s1 less than s2.

Example -
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{     int      ans;      char    s1[50], s2[50];
              clrscr();
              printf("Enter string  s1…");
              scanf("%s",s1);
              printf("Enter string  s2…");
              scanf("%s",s2);
              ans = strcmp(s1,s2) ;
              if  ( ans = = 0 )     printf("\n Both strings are equal");
              if  ( ans > 0 )     printf("\n String1 is greater");
              if  ( ans < 0 )     printf("\n String2 is gerater");
              getch();
}
```
<u>Output</u>:
```
Enter string  s1… mamu
Enter string s2… mamu
Both strings are equal
```

[5]. String reverse: strrev()

Syntax - char *strrev(s1)
Where s1 is string.

Description - It is used to reverse the contents of a string.

Example -
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{        char    s1[50];                clrscr();
              printf("Enter string…");
              scanf("%s",s1);
              printf("\n Original string = %s", s1);
              strrev(s1) ;
              printf("\n Reverse string = %s", s1);                getch();
}
```
<u>Output</u>:
```
Enter string… xyz
Original string = xyz
Reverse string = zyx
```

Common Standard Library Functions

abs()

Syntax - int abs(n)
where n is an integer.

Header file - math.h

Description - It returns absolute value of an argument.

Example -

```
//Example on abs() function.
#include<stdio.h>
#include<conio.h>
#include<math.h>
void  main()
{       int   n = -123 ;
              clrscr( ) ;
              printf("\n Given number : %d", n);
              printf("\n Absolute value : %d", abs(n));
              getch( );
}
```

Output -

```
Given number : - 123
Absolute value : 123
```

pow()

Syntax - double pow(a, b)
where a and b are double.

Header file - math.h

Description - It returns value of a raised to b.

Example -

```
//Example on pow() function.
#include<stdio.h>
#include<conio.h>
#include<math.h>
void  main()
{       float   a = 2, b = 3 ;
              clrscr( ) ;
              printf("\n 2 raised to 3 is : %0.2f", pow(a,b));
              getch( );
}
```

Output -

```
2 raised to 3 is 8.00
```

sqrt()

Syntax - double sqrt(n)
where n is double.

Header file - math.h

Description - It returns square root of an argument.

Example - //Example on sqrt() function.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void  main()
{       float   a = 25;
             clrscr( ) ;
             printf("\n Square root of 25 is : %0.2f", sqrt(a));
             getch( );
}
```

Output - Square root of 25 is 5.00

---

toupper( )

Syntax - returnvalue = toupper(character) ;

Header file - *ctype.h*

Description - This function is used to convert your character into uppercase from lowercase. Here the returnvalue should be character type.

Example - char returnvalue;
returnvalue = toupper('a') ;

Output the value of the returnvalue is 'A' after execution.

---

tolower( )

Syntax	-	returnvalue = tolower(character) ;
Header file	-	*ctype.h*
Description	-	This function is used to convert your character into lower case from uppercase. Here the returnvalue should be character type.

Example	-	char returnvalue;
		returnvalue = tolower('A') ;

Output		Here the value of the returnvalue is 'a' after execution.

---

isupper()

Syntax - int isupper(c)
where c is a character.

Header file - ctype.h

Description - It returns true if an argument is an upper case letter (A to Z), otherwise false.

Example -

```
//Example on isupper() function.
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void  main()
{       char   c ;
                clrscr( ) ;
                printf("Enter any character :");
                scanf("%c",&c);
                if ( isupper(c) )
                        printf("It is an upper case letter.");
                getch( );
}
```

Output -

```
Enter any character : A
It is an upper case letter.
```

---

getchar( )

Syntax - variablename = getchar( )

Header file - Stdio.h

Description - This function is used to read simply one character from standard input device.

Example -

```
#include<stdio.h>
#include<conio.h>
void  main()
{       char   name ;
                clrscr( ) ;
                printf("Enter name :");
                name = getchar ( ) ;
                getch( );
}
```

---

islower()

Syntax - int islower(c)
where c is a character.

Header file - ctype.h

Description - It returns true if an argument is a lower case letter (a to z), otherwise false.

Example -

```
//Example on islower() function.
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void  main()
{          char   c ;
                     printf("Enter any character :");
           scanf("%c",&c);
           if ( islower(c) )
                  printf("It is an lower case letter.");
           getch( );
}
```

Output -

```
Enter any character : a
It is an lower case letter.
```

---

isapha()

Syntax - int isalpha(c)
where c is a character.

Header file - ctype.h

Description - It returns true if an argument is an upper case (A to Z) or lower case letter (a to z), otherwise false.

Example -

```
//Example on isalpha() function.
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void  main()
{      char   c ;
           clrscr( ) ;
           printf("Enter any character :");
           scanf("%c",&c);
           if ( isalpha(c) )
                printf("It is an alphabets.");
           getch( );
}
```

Output -

```
Enter any character : A
It is an alphabets.
```

---

isdigit()

Syntax - int isdigit(c)
where c is a character.

Header file - ctype.h

Description - It returns true if an argument is a digit, otherwise false.

Example -

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void  main()
{       char   c ;
              clrscr( ) ;
              printf("Enter any character :");
              scanf("%c",&c);
              if ( isdigit(c) )
                    printf("It is a digit.");
              getch( );
}
```

Output -

```
Enter any character : 5
It is a digit.
```

---

getche( )

Syntax - variablename = getche( ) ;

Header file - *conio.h.*

Description - This function is used to read the character from the standard input device.

Example -

```
#include<stdio.h>
#include<conio.h>
void  main()
{       char   name ;
              clrscr( ) ;
              printf("Enter name :");
               name = getche( ) ;
              getch( );
}
```

---

isalnum( )

Syntax - returnvalue = isalnum(character) ;

Header file - *ctype.h*

Description - This function is used to check whether the character is digit (0 to 9) or alphabet (A to Z). Here the returnvalue should be integer type. If your character is digit or alphabet then it will return true (non-zero value) otherwise it will return false (zero value). Here you can pass either the character variable or character constant as argument.

Example -

```
int returnvalue;
returnvalue = isalnum('3') ;
```

Output Here the value of the returnvalue is non-zero since the '3' is digit.

---

getch( )

Syntax - variablename = getch( )

Header file - *conio.h*.

Description - This function is used to read the character from the standard input device but it will not echo (display) the character, which you are inputting.

Example -
```
#include<stdio.h>
#include<conio.h>
void  main()
{      char   name ;
            clrscr( ) ;
            printf("Enter name :");
            name = getch( ) ;
            getch( );
}
```

---

putchar( )

Syntax - putchar(variablename)

Header file - *stdio.h*

Description - This function is used to print one character on standard output device.

Example -
```
#include<stdio.h>
#include<conio.h>
void  main()
{
      char name;
       name='P';
       putchar(name);
       getch( );
}
```

Output P

---

ispunct( )

Syntax - returnvalue = ispunct(character) ;

Header file - *ctype.h*

Description - This function is used to check whether the character is punctuation mark (special character. Not space, digit, alphabet, and non-printable character). Here the returnvalue should be integer type. If your character is punctuation mark then it will return true (non-zero value) otherwise it will return false (zero value). Here you can pass either the character variable or character constant as argument.

Example -
```
int returnvalue;
returnvalue = ispunct('$') ;
```

Output Here the value of the returnvalue is non-zero since the '$' is punctuation mark.

---

isspace( )

Syntax	-	returnvalue = isspace(character) ;
Header file	-	*ctype.h*
Description	-	This function is used to check whether the character is space character (horizontal tab, new line, vertical tab, form feed, carriage return, space) or not. Here the returnvalue should be integer type. If your character is space character then it will return true (non-zero value) otherwise it will return false (zero value). Here you can pass either the character variable or character constant as argument.

Example	-	int returnvalue;
		returnvalue = isspace(‘ ’) ;

Output		Here the value of the returnvalue is non-zero since the ‘ ’ is space character.

isprint( )

Syntax	-	returnvalue = isprint(character) ;
Header file	-	*ctype.h*
Description	-	This function is used to check whether the character is printable or not. Printable characters are alphabet (A to Z), digit (0 to 9), and special character ($, %, #, &, etc.) while control, alter, shift, bell, null character are non-printable character. Here the returnvalue should be integer type. If your character is printable then it will return true (non-zero value) otherwise it will return false (zero value). Here you can pass either the character variable or character constant as argument.

Example	-	int returnvalue;
		returnvalue = isprint(‘3’) ;

Output		Here the value of the returnvalue is non-zero since the ‘3’ is printable character.