STRINGS

- ✓ A string is a sequence of character that is treated as single data item.
- ✓ In C language the group of characters, digits and symbols enclosed within quotation marks are called as strings.
- ✓ Character strings are often used to build meaningful and readable programs. The common operations performed on character strings include
    - o Reading and writing strings
    - o Combining strings together
    - o Copying one string to another
    - o Comparing strings for equality
    - o Extracting a portion of string

Declaration & Initialization of Strings

- ✓ C does not support strings as a data type. But it allows us to represent strings as character arrays.
- ✓ The general form of declaration of a string variable is

char string_name[size];

Size determines the number of characters in the string_name. Some examples are

char city[10];

char name[20];

- ✓ 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.
- ✓ Character arrays are initialized when they are declared. 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’,’\0’};

- ✓ C also permits us to initialize a character array without specifying the number of elements. In such cases the size of the array will be determined automatically, based on the number of elements initialized, For example the statement

char string []={‘I’,’N’,’D’,’I’,’A’,’\0};

defines the string as 6 element array.

- ✓ We can also declare the size much larger than the string size in the initializer. For example the statement,

  char str[10]=”INDIA”;

  is permitted. In this case, the computer creates a character array of size 10, places the value “INDIA” in it, terminates with null character and initializes all other elements with the null character.

Reading strings from the terminal

- ✓ Scanf can be used with **%s** format specification to read in a string of characters
  For example:

  char address [10];

  scanf(%s”, address);
- ✓ The problem with the scanf function is that it terminates its input on the first white space it finds.
- ✓ We can also specify the field width using the form **%ws** in the scanf statement for reading a specified number of character from the input string.
  For Example

  scanf (“%ws”,name);

  Here two things may happen,

  1. The width **w** is equal to or greater than the number of characters typed in. The entire string will be stored in the string variable.
  2. The width **w** is less that the number of character in the string. The excess character will be truncated and left un read

Consider the Following statements:

char name[10];

scanf (“%5s”, name);

Example1:

The input string “**HAI**” will be stored as

H	A	I	\0	?	?	?	?	?	?
0	1	2	3	4	5	6	7	8	9

Example 2:

The input string "**WELCOME**" will be stored as

W	E	L	C	O	\0	?	?	?	?
0	1	2	3	4	5	6	7	8	9

Reading a line of text

- ✓ Scanf with **%s or %ws** can read only strings without white spaces. That is they cannot be used for reading a text containing more than one word. However, C supports a format specification known as the **edit set conversion code % […]** that can be used to read a line containing a variety of characters, including white spaces.
- ✓ For example consider the following program segment

```
char line[80];
scanf("%[^\n],line];
printf("%s",line);
```

will read a line of input from the keyboard and display the same on the screen

Using getchar and gets functions

- ✓ We already discussed how to read a single character from the terminal using the function **getchar**. We can use this function repeatedly to read successive single characters from the input and place them into a character array. Thus an entire line of text can be read and stored in an array. The reading is terminated when the **newline** character('\n')is entered and the null character is inserted at the end of the string.
- ✓ The getchar function call takes the form

```
char ch;
ch=getchar();
```

- ✓ Another and more convenient method of reading a string of text containing white spaces is to use the library function **gets**. This is a simple function with one string parameter and called as

```
gets(str);
```

str is a string variable declared properly. It reads character into **str** through keyboard until a new-line character is encountered and then appends a null character to the string. It does not skip white spaces.

For example:

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

reads a line of text from the keyboard and displays it on the screen.

Writing Strings to the screen

- ✓ Printf function with %s format to print strings to the screen. The format %s can be used to display an array of character that is terminated by the null character. For example the statement

  printf("%s", name);

  can be used to display the entire contents of the array name.
- ✓ We can also specify the precision with which the array is displayed. For example the specification **%10.4s** indicates that the first four characters are to be printed in a field width of 10 columns.

Example Program: Display the string under various format specification

```
#include<stdio.h>
#include<conio.h>
main()
{
        char state[15]="Andhra Pradesh";
        printf("\n\n");
        printf("-------------------------\n");
        printf("%15s\n", state);
        printf("%5s\n", state);
        printf("%15.6s\n", state);
        printf("%-15.6s\n", state);
        printf("%15.0s\n", state);
        printf("%.3s\n", state);
        printf("%s\n", state);
        printf("-------------------------\n");
}
```

OUTPUT

Andhra Pradesh

Andhra Pradesh

Andhra

Andhra

And

Andhra Pradesh

Using Putchar and puts function

- ✓ Like getchar, C supports another character handling function putchar to output the values of character variables. It takes the following form

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

The function putchar requires one parameter. The statement is equal to **printf("%c",ch);**

- ✓ We can use this function repeatedly to output a string of character stored in an array as like the following example.

```
char name[6]="HAI"
for (i=0;i<5;i++)
putchar (name[i]);
putchar("\n");
```

- ✓ Another and more convenient way of printing string values is using the function **puts**

```
puts(str);
```

where **str** is a string variable containing a string value. This prints the value of the string variable **str** and then moves the cursor to the beginning of the next line on the screen. For example consider the following program segment

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

reads a line of text from the keyboard and displays it one screen. This syntax is very simple compared to using the scanf and printf functions

STRING LIBRARY FUNCTIONS

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

Functions	Description
Strlen()	Determines the length of a string
Strcpy()	Copies a string from source to destination
Strncpy()	Copies character of a string to another string upto specified length
Stricmp()	Compares characters of two strings.
Strcmp()	Compares two strings
Strncmp()	Compares characters of two strings upto the specified length
Strnicmp()	Compares characters of two strings upto the specified length, Ignores case
Strlwr()	Converts uppercase characters of a string to lowercase
Strupr()	Converts lowercase character of a string to uppercase
Strudp()	Duplicates a string
Strchr()	Determines the first occurrence of a given character in a string
Strrchr()	Determines the last occurrence of a given character in a string
Strstr()	Determines the first occurrence of a given string in another string
Strcat()	Appends source string to destination string
Strncat()	Appends source string to destination string upto the specified length
Strrev()	Reverses all character of a string
Strset()	Sets all character of a string with a given argument or symbol
Strnset()	Sets specified numbers of characters of a string with a given argument or symbol
Strspn()	Finds upto what length two strings are identical
Strpbrk()	Searches the first occurrence of the character in a given string and then displays the string starting from that character

1. strlen() Function

This function counts the number of character in a given string. The format of functions is

strlen(string);

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
        char text[20];
        int len;
        clrscr();
        printf("Enter the text:\n");
        gets(text);
        len=strlen(text);
        printf("length of the string=%d",len);
}
```

OUTPUT

Enter the text

Welcome

Length of the string: 7

2. strcpy()function

This function almost like a string-assignment operator. this function copies the contents of one string to another. The format of strcpy() is

strcpy(string1, string2);

Example Program

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

```
main()
{
        char original[20],duplicate[20];
        clrscr();
        printf("enter the string:");
        gets(original);
        strcpy(duplicate,original);
        printf("\n original string:%s".original);
        printf("\n  duplicate string:%s",duplicate);
        getch();
}
```

OUTPUT:

Enter the string: **Vellore institute of Technology**

Original string: **Vellore institute of Technology**

Duplicate String: **Vellore institute of Technology**

3. strncpy()function

This function performs the same task as strcpy(). The only difference between them is that the former function copies a specified length of character from source to destination string. Whereas this function copies the whole source to the destination string. The format of the function is

strncpy(destination, source, n);

Example Program

```
        #include<stdio.h>
        #include<conio.h>
        #include<string.h>
        void main
        {
                char str1[15],str2[15];
                int n;
                clrscr();
                printf("enter source string:");
                gets(str1);
```

```
        printf("enter Destination string:");
        gets(str2);
        printf("enter the number of character to replace in the destination :");
        scanf("%d",&n);
        strncpy(str2, str1, n);
        printf("source string:%s, str1);
        printf("destination string:%s", str2);
}
```

OUTPUT

Enter source string : wonderful
Enter destination string : beautiful
Enter number of characters to replace in destination: 6
Source string : wonderful
Destination string : wonderful

4. stricmp() function

This function compares two strings. The character of the strings may be in lower case or upper case. The function does not discriminate between them, that is, this function compares two strings without case. If the strings are same it returns to zero otherwise a non-zero value.

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char str1[15],str2[15];
        int diff;
        clrscr();
        printf("enter string1:");
        gets(str1);
        printf("enter string2:");
```

```
        gets(str2);
        diff=stricmp(str1, str2);
        if(diff==0)
        puts("the two strings are identical");
        else
        puts("the two strings are not identical");
        getche();
}
```

OUTPUT:

Enter string1 : **WELCOME**

Enter string2 : **welcome**

The two strings are identical

5. strcmp() function

We can also use strcmp() function instead of stricmp(). The only difference between them is that the former function discriminates between small and capital letters.

6. strncmp() function

A comparison of two strings can be made upto certain specified length. The function used for this is **strncmp()**. This function is the same as **strcmp()** but it compares the character of the string to a specified length. The format of tehis function is as follows.

strncmp(source,target, argument)

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char str1[15],str2[15];
        int n, diff;
        clrscr();
        printf("enter string1:");
        gets(str1);
        printf("enter string2:");
```

```
        gets(str2);
        printf("\n Enter length up to which comparison is to be made");
        scanf("%d",&n);
        diff=strncmp(str1,str2,n);
        printf("the two strings are identical up to %d character",n);
        else
        puts("the two strings are differenct");
        getch();
    }
```

OUTPUT

Enter string1: WELCOME

Enter string2: WELLDONE

Enter length up to which comparison is to be made: 3

The two strings are identical up to 3 characters

7. strnicmp () function

We can also use **strnicmp()** function instead of **strncmp().** The only difference between them is that the **strncmp()** discriminates between small and capital letter whereas the **strnicmp()** does not. The output of the above program with **strnicmp()** in place of **strncmp()** will be as follows.

Enter string1: WELCOME

Enter string2: welldone

Enter length up to which comparison is to be made: 3

The two strings are identical up to 3 characters

8. strlwr & strupr() function

strlwr function an be used to convert any string to a lower case. When you are passing any upper case string to this function it converts it into lower case. **Strupr()** function can be used convert lower case to upper case.

The format of strlwr() function is **strlwr(string);**

The format of strupr() function is **strupr(string);**

Example Program(Upper case to lower case):

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char str1[15];
        clrscr();
        printf("enter string in upper case:");
        gets(str1);
        printf("after strlwr(): %s", strlwr(str1));
}
```

OUTPUT

Enter string in upper case: WELCOME

After strlwr : welcome

9. strudp () function

This function is used for duplicating a given string at the allocated memory which is pointed by a pointer variable. The format of this function is as follows

text2=strudp(text1);

where text1 is a string

text2 is a pointer

Example Program: Entering the string and getting the duplicate

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text1[15], *text2;
        clrscr();
        printf("enter text:");
        gets(text1);
        text2=strudp(text1);
        printf("Original string=%s\n Duplicate String:%s", text1,text2);
```

```
}
```

OUPTUT

Enter text: Have a nice day

Original string: Have a nice day

Duplicate string: Have a nice day

10. strchr() Function

This function returns the pointer to the first occurrence of a given character in the string. The format of this function is as follows

chp=strchr(string,ch);

where **string** is a character array, **ch** is character variable, **chp** is a pointer which collects the address returned by **strchr()** function The format of this function is **strchr(string,character)**

Example Program/ Program to find the occurrence of a given character in the string

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char string[30], ch, *chp;
        clrscr();
        printf("enter text:");
        gets(string);
        printf("\n character to find");
        ch=getchar();
        chp=strchr(string,ch);
        if(chp)
        printf("character %c found in string",ch);
        else
        printf("character %c not found instring",ch);
}
```

OUTPUT

Enter text: Have a nice day

Character to find: n

Character n found in string

11. strrchr() Function

In place of **strchr()** one can use **strrchar().** The difference between them is that the **strchr()** searches for occurrence of character from the beginning of the string whereas **strrchr()** searches occurrence of character from the end.

12. strstr() Function

This function finds the second string in the first string. It returns the pointer location from where the second string starts in the first string. In case the first occurrence in the string is not observed, the function returns a NULL character. The format of this function is

strstr(string1,string2);

Example Program/ Program to find the occurrence of a second string in first string

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char line1[35],line2[35], *chp;
        clrscr();
        puts("enter line1:");
        gets(line1);
        puts("enter line2:");
        gets(line2);
        chp=strstr(line1,line2);
        if(chp)
        printf("'%s'string is present in given string",line2);
        else
        printf("'%s' string is not present in given string",line2);
}
```

OUTPUT

Enter line1 : Have a nice day

Enter lin2 : day

'day' string is present in the given string

13. strcat() Function

This function appends the target string to the source string. Concatenation of two strings can be done using this function. The format of this function is as follows

strcat(tex1,text2);

Example Program: Joining two strings

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text1[30], text2[30];
        puts("enter text1");
        gets(text1);
        puts("enter text2");
        puts(text2);
        strcat(text1," ");
        strcat(text1, text2);
        clrscr();
        printf("%s\n", text1);
}
```

OUTPUT

Enter text1 : Have a

Enter text2 : nice day

Have a nice day

14. strncat() Function

This function is the same as that of strcat(). The difference between them is that the former does the concatenation of the strings with another up to a specified length. The format of this function is

strncat(text1,text2,n);

Example Program: Joining two strings

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text1[30], text2[30],n;
        puts("enter text1");
        gets(text1);
        puts("enter text2");
        puts(text2);
        printf("enter number of characters to add:");
        gets(n);
        strcat(text1," ");
        strncat(text1, text2);
        clrscr();
        printf("%s\n", text1);
}
```

OUTPUT

Enter text1 : Have a

Enter text2 : nice day

Enter number of characters to add: 4

Have a nice

15. strrev() Function

This function simply reverses the given string. The format of this function is

strrev(string);

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
```

```
{
        char text[15];
        clrscr();
        puts("enter string1:");
        gets(text);
        puts("reverse string");
        puts(strrev(text));
}
```

OUTPUT

Enter string

Welcome

Reverse string

emoclew

16. strset() Function

This function replaces every character of a string with the symbol given by the programmer, that is the elements of the strings are replaced with the arguments given by the programmer. The format of this function is

strset(string, symbol)

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text[15];
        char symbol;
        clrscr();
        puts("enter string:");
        gets(text);
        puts("enter symbol for replacement:");
        scanf("%c", &symbol);
        printf("Before strset():%s\n", text);
        strset(string, symbol);
```

```
        printf("After strset(); %s\n", text);
    }
```

OUTPUT

Enter string : Computer

Enter symbol for replacement: x

Before strset(): Computer

After strset(): XXXXXXX

17. strnset() Function

This function is the same as that of **strset().** Here the specified length is provided. The format of this function is

strnset(string, symbol, n);

where n is the number of characters to be replaced.

Example Program

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text[15];
        char symbol;
        int n;
        clrscr();
        puts("enter string:");
        gets(text);
        puts("enter symbol for replacement:");
        scanf("%c", &symbol);
        puts("how many string character to be replaced");
        scanf("%d",&n);
        printf("Before strset():%s\n", text);
        strset(string, symbol,n);
        printf("After strset(); %s\n", text);
}
```

OUTPUT

Enter string : Computer

Enter symbol for replacement: x

How many string characters to be replaced: 4

Before strset(): Computer

After strset(): XXXXuter

18. strspn() Function

This function returns the position of the string from where the source array does not match with target one. The format of this function is

strspn(string1, string2);

Example Program: Indicate after what character the lengths of the two strings have no match

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char text1[30], text2[30];
        int length;
        clrscr();
        puts("enter text1");
        gets(text1);
        puts("enter text2");
        gets(text2);
        length= strspn(text1,text2);
        printf("After %d character there is no match\n", length);
}
```

OUTPUT

Enter text1 : GOOD MORNING

Enter text2 : GOOD BYE

After 5 characters there is no match

19. strpbrk() Function

This function searches the first occurrence of a character in the given string and then it displays the string starting from that character. This function returns the pointer position to

the first occurrence of the character **text2[2**] string in the string **text1[20].** The format of this function is

strpbrk(text1,text2)

Example Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main
{
        char *ptr;
        char text1[20], text2[2];
        clrscr();
        puts("enter text1");
        gets(text1);
        puts("enter character");
        gets(text2);
        ptr=strpbrk(text1,text2);
        puts("string from given character");
        printf(ptr);
}
```

OUTPUT

Enter text1 : Have a nice day

Enter character: n

string from given character: nice day