STRING MANIPULATIONS IN C

- In C language, an array of characters is known as a string. char
   st[80];
- This statement declares a string array with 80 characters.
- The control character „\0‟ which represents a null character is placed automatically at the end of any string used.

STRING HANDLING FUNCTIONS IN C:

There are four important string Handling functions in C language.

(i) **strlen( )** function

(ii) **strcpy( )** function

(iii) **strcat( )** function

(iv) **strcmp( )** function

(I) strlen( )Function:

strlen( ) function is used to find the length of a character string. Ex:

```
int  n;

char st[20] = “Bangalore”; n
= strlen(st);
```

- This will return the length of the string 9 which is assigned to an integer variable n.
- Note that the null charcter „\0‟ available at the end of a string is not counted.

(II) strcpy( )Function:

strcpy( ) function is used to copy from one string to another string. Ex :

char city[15];

strcpy(city, “BANGALORE”) ;

This will assign the string “BANGALORE” to the character variable city.

* Note that character value like

city = “BANGALORE”; cannot be assigned in C language.

(III)<u>strcat( )Function:</u>

strcat( ) function is used to join character, Strings. When two character strings are joined, it is referred as concatenation of strings.

Ex:

char city[20] = “BANGALORE”;

char pin[8] = “-560001”;

$$\text{strcat}\left(\underbrace{city}_{(L)},\underbrace{pin}_{(R)}\right);$$

- This will join the two strings and store the result in city as “BANGALORE – 560001”.
- Note that the resulting string is always stored in the left side string variable.

(iv) <u>strcmp( ) Function:</u>

strcm ( ) function is used to compare two character strings.

- It returns a 0 when two strings are identical. Otherwise it returns a numerical value which is the different in ASCII values of the first mismatching character of the strings being compared.

```
char  city[20]  =  “Madras”;  char
town[20]       =       “Mangalore”;
strcmp(city, town);
```

- This will return an integer value „- 10‟ which is the difference in the ASCII values of the first mismatching letters „D‟ and „N‟

- Note that the integer value obtained as the difference may be assigned to an integer variable as follows:

```
int  n;

n = strcmp(city, town);
```

<u>READING / WRITING STRINGS:</u>

- We use scanf ( ) function to read strings which do not have any white spaces. Ex: **scanf(“ %s ”, city);**

- When this statement is executed, the user has to enter the city name

  (eg “NEWDELHI”) without any white space (i.e not like “NEW DELHI”).

- The white space in the string will terminate the reading and only “NEW” is assigned to city.

- To read a string with white spaces gets( ) function or a loop can be used as.

  (i) gets(city);

  (ii) do … while loop i

  ```
      = 0;

     do

      {
  ```

```
        Ch  =  getchar(  );
        city[ i ] = ch; i++;

    }

   while(ch ! = „ \n ‟) ;

    i - - ;           city[ i ] = „\0‟;
```

- The copy or assign a single character to a string variable, the assignment can be written as given below;

```
city[ i ] = „N‟;
```

- When a string contains more than one character, the assignment is written as strcpy (city, "NEW DELHI");

atoi( ) Function:

atoi( ) function is a C library function which is used to convert a string of digits to the integer value.

```
char  st[10]  =  "24175"  ;
int n;

n = atoi(st);
```

This will assign the integer value 24175 to the integer variable n.

(1) Write a C program to count the occurrence of a particular character in the given string.

Solution:

Consider a string "MISSISSIPPI". Count the appearance of a character, say „S‟.

Program:

```
\* program to count a character in a string * / #
include<stdio.h>
```

```
# include<conio.h> #
include<string.h>
main( )

{

        char   st[20],   ch;
        int    count,    l,i;
        clrscr( );

        printf(“ \n Enter the string:”);
        gets(st);

        printf(“ \n which char to be counted ?”);
        scanf(“ %c”, &ch);

 /* loop to check the occurrence of the character * / l =
        strlen(st);

        count = 0;

        for( i=0; i < l; i++)
           if(st[ i ] = = ch)
           count ++;

        printf(“ \n the character %c occurs %d times”, ch, count);
        getch( ) ;

}
```

- When this program is executed, the user has to enter the string and the character to be counted in the given string.

Output:

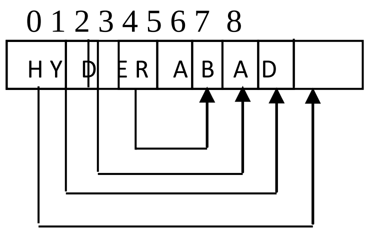
```
Enter the string : MISSISSIPPI
```

Which char to be counted? S

The character S occurs 4 times.

<u>Alternative Method:</u>

- This program can also be written to compare the first character of the string with the last, second with the second last and so on up to the middle of the string as shown.



- If all the characters are identical, then the string is a palindrome string.

- Otherwise the loop will be terminated.

(2) Write a C program to compare two strings which are given as input through keyboard and print the alphabetically greater string.

Solution

The alphabetically greater string is the one whose ASCII value of the first letter is greater than that of the second string.

Consider the two character strings.

St1 = “ALPHA”

St2 = “BEETA”

St2 is greater than st1 because the ASCII value of „B‟ in “BETA” is greater than that of „A‟ in „ALPHA‟.

- Note that when the first letters of the two strings are identical, the second letters of the strings are compared.

```
/* PROGRAM TO PRINT ALPHABETICALLY GREATER STRING * /

  # include<stdio.h>

  # include<conio.h>

   #include<string.h>

  main( )

   {

          char st1[20], st2[20];

           clrscr(  );

          printf(“  \n  Enter  string:”);
          scanf( “ %s ”,st1);

          printf(“ \n Enter string 2:”);
          scanf(    “    %s    ”,    st2);
          if(strcmp(st1,st2)> 0)

               printf(“\n  %s  is  alphabetically  greater  string”,  st1);
          else

               printf(“  \n  %s  is  alphabetically  greater  string”,  st2);
          getch(  );

   }
```

- When this program is executed, the user has to enter the two strings.
- Note that strcmp( ) function returns a positive value when string 1 is greater & a negative value when string 2 is greater.

Output:

```
Enter string 1 : ACPHA
Enter string 2 : BETA
```

BETA is alphabetically greater string.

PROGRAM TO ARRANGE NAMES IN ALPHABETICAL ORDER.

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

  {

        char names[50][20], temp[20]; int
        n,i,j;

        clrscr( ):

        printf(“ \n How many names?”);

         scanf(“%d”, &n);

        printf(“ \n Enter the %d names one by one \ n”,n);

        for( i=0; i<n; i++)

        scanf(“%s”, names[ i ]);

/* loop to arrange names in alphabetical order * /
        for( i=0; i<n-1; i++)

          for ( j =i+1; j<n; j++)

                if(strcmp(names[ i ], names[ j ]) > 0)

                  {

                     strcpy( temp, names[ i ]); strcpy(names[ i ], names[ j ]);
                     strcpy(names[ j ],temp);

/* loop to print the alphabetical list of names * / printf (“ \n names in alphabetical order”);

for( i=0; i<n; i++)
```

```
            printf("\n %s", names[ i ]);
                getch( );
      }
```

- When this program is executed, the user has to first enter the total no of names

(n) and then all the names in the list.

- The names are compared, rearranged and printed in alphabetical order.

Output:

```
How many names? 4

Enter the 4 names one by one
          DEEPAK

          SHERI
          N
          SONIK
          A
          ARUN

Names in Alphabetical order
          ARUN

          DEEPAK
          SHERIN
          SONIKA
```

(3) Write a C program to convert a line in the lower case text to upper case.

Solution

Consider ASCII values of lower and upper case letters A – 65 B – 66.................................... Z – 90 a - 97 b – 98 ....z – 122

* Note that the difference between the lower and upper case letters (ie. – 32) is used for conversion.

/* PROGRAM TO CONVERT LOWER CASE TEST TO UPPER CASE * /

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

{

        char
        st[80];   int
        i; clrscr( );

        printf(“ \ n Enter a sentence : \ n”);
        gets(st);

/* loop to convert lower case alphabet to upper case * / for(
        i=0; i<strlen(st); i++)

            if(st[ i ] >= „a‟ && st[ i ] <= „z‟)st[ i ] = st[ i ] – 32;

        printf(“ \n the converted upper case strings \n %s”, st);
        getch( );

}
```

OUTPUT:

```
Enter a sentence:

Logical thinking is a must to learn programming
The converted upper case string is
```

Additional String Handling Functions:

Some „C‟ compilers will accept the following string handling functions which are available in header files string.h and ctype.h

Functions:

(i) strupr( ): to convert all alphabets in a string to upper case letters. Ex:

strupr(“ delhi ”) “ DELHI”

(ii) strlwr( ): To convert all alphabets in a string to lower case letters. Ex:

strlwr(“ CITY ”) “city”

(iii) strrev( ): To reverse a string

Ex: strrev(“ SACHIN ”) “NIHCAS”

(iv) strncmp( ): To compare first n characters of two strings. Ex:

m = strncmp (“ DELHI ”, “ DIDAR ”, 2);

 m = -4

(v) strcmpi( ): To compare two strings with case in sensitive (neglecting upper / lower case)

Ex:

m=strcmpi(“ DELHI ”, “ delhi ”); m = 0.

(vi) **strncat( ):** To join specific number of letters to another string. Ex.

```
char  s1[10] = “New”; char
s2[10]    =    “Delhi    -41”;
strncat(s1,s2,3);
```

 s1 will be “NewDel”.

Operations with Characters:

C language also supports operations with characters and these functions are available in the header file “ctype.h”.

*** Note that the value „1‟ or positive integer will be returned when acondition is true, and value „0‟ will be returned when condition is false.**

- Some „ctype.h‟ functions are listed below.

Function:

(i) **isupper( ) :** To test if a character is a upper case letters. Ex:

isupper(“A‟) 1

isupper(“a‟) 0

isupper(“8‟) 0

(ii) **islower( ) :** To test if a charcter is a lower case letter. Ex:

islower(“n‟) 1

(iii) **isalpha( ) :** To test if a character is an alphabet. Ex:

isalpha(“K‟) 1

isalpha(“+‟) 0

(iv) **isalnum( ) :** To test if a character is an alphabet or number. Ex:

isalnum(“8‟) 1

isalnum(“y‟) 1

isalnum(“-‟) 0

(v) isdigit( ) : To test if a character is a number. Ex:

isdigit(“6‟) 1

isdigit(“a‟) 0

(vi) isxdigit( ) : To test if a character is a Hexa decimal number (0-9, A-F and a-f are Hexa decimal digits)

Ex:

isxdigit(“9‟) 1

isxdigit(“A‟) 1

isxdigit(“M‟) 0

(vii) tolower( ) : To convert an alphabet to lower case letter Ex:

tolower(“A‟) a

(viii) toupper( ) : To convert an alphabet to upper case letter Ex:

toupper(“a‟) A.

 Functions:-

 Designing:-

Functions are an essential ingredient of all programs, large and small, and serve as our primary medium to express computational processes ina programming language. So far, we have discussed the formal properties of functions and how they are applied. We now turn to the topic of what makes a good function. Fundamentally, the qualities of good functions all reinforce the idea that functions are abstractions.

 Each function should have exactly one job. That job should be identifiable with a short name and characterizable in a single line oftext. Functions that perform multiple jobs in sequence should be divided into multiple functions.

 Don't repeat yourself is a central tenet of software engineering. Theso-called DRY principle states that multiple fragments of code should not describe redundant logic.

Instead, that logic should be implemented once, given a name, and applied multiple times. If youfind yourself copying and pasting a block of code, you have probably found an opportunity for functional abstraction.

- Functions should be defined generally. Squaring is not in thePython Library precisely because it is a special case of

  the pow function, which raises numbers to arbitrary powers.

These guidelines improve the readability of code, reduce the number oferrors, and often minimize the total amount of code written.

Decomposing a complex task into concise functions is a skill that takesexperience to master. Fortunately, Python provides several features tosupport your efforts.

- **Structured programs:-**

  Structured programming is a programming paradigm aimed atimproving the clarity, quality, and development time of a

  computer program by making extensive use of the structured controlflow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines.

Function in C:-

A function is a block of statements that performs a specific task. Suppose you are building an application in C language and in one of your program, you need to perform a same task more than once. In suchcase you have two options –

a) Use the same set of statements every time you want to perform thetask

b) Create a function to perform that task, and just call it every time youneed to perform that task.

Using option (b) is a good practice and a good programmer always usesfunctions while writing codes in C.

Types of functions

1) **Predefined standard library functions** – such

as puts(), gets(), printf(), scanf() etc – These are the functions which already have a definition in header files (.h files like stdio.h), so we justcall them whenever there is a need to use them.

- 2) **User Defined functions –** The functions that we create in aprogram are known as user defined functions.

- **Inter-Function Communication:-**

When a function gets executed in the program, the execution control is transferred from calling a function to called function and executes function definition, and finally comes back to the calling function. In this process, both calling and called functions have to communicate witheach other to exchange information. The process of exchanging information between calling and called functions is called inter-function communication.

In C, the inter function communication is classified as follows...

- **Downward Communication**
- **Upward Communication**
- **Bi-directional Communication**

- **Standard library functions:-**

The standard library functions are built-in functions in C programming.These functions are

defined in header files. For example,

The printf() is a standard library function to send formatted output to the

screen (display output on the screen). This function is defined inthe stdio.h header file.

Hence, to use the printf() function, we need to include the stdio.h headerfile using #include <stdio.h>.

The sqrt() function calculates the square root of a number. The functionis defined in the math.h header file.

- **Passing Array to Functions**:-

Just like variables, array can also be passed to a function as an argument. In this guide, we will learn how to pass the array to a functionusing call by value and call by reference methods.

To understand this guide, you should have the knowledge offollowing C Programming topics:

C – Array

Function call by value in C Function call by reference in C

Passing array to function using call by value method

As we already know in this type of function call, the actual parameter iscopied to the formal parameters.

```
#include<stdio.h>
voiddisp(charch)

{

printf("%c ",ch);

}

int main()

{

chararr[]={'a','b','c','d','e','f','g','h','i','j'}; for(int x=0;
x<10; x++)

{

/* I'm passing each element one by one using subscript*/disp(arr[x]);

}



return0;

}
```

Output:

```
a b c d e f g h i j
```

Passing array to function using call by reference

When we pass the address of an array while calling a function then this is called function call by reference. When we pass an address as an argument, the function declaration should have a pointer as a parameterto receive the passed address.

```
#include<stdio.h>
voiddisp(int*num)

{

printf("%d ",*num);
```

```
}

int main()

{ intarr[]={1,2,3,4,5,6,7,8,9,0};

for(inti=0;i<10;i++)

{

/* Passing addresses of array elements*/disp(&arr[i]);

}
```

```
return0;

}
```

Output:

```
1234567890
```

How to pass an entire array to a function as an argument?

In the above example, we have passed the address of each array element one by one using a for loop in C. However you can also pass anentire array to a function like this:

Note: The array name itself is the address of first element of that array.For example if array name is arr then you can say that arr is equivalentto the &arr[0].

```
#include<stdio.h>
voidmyfuncn(int*var1,int var2)

{

        /* The pointer var1 is pointing to the first element of

         * the array and the var2 is the size of the array. In the

         * loop we are incrementing pointer so that it points to

         * the next element of the array on each increment.

         *

         */

for(int x=0; x<var2; x++)

{

printf("Value of var_arr[%d] is: %d \n", x,*var1);

/*increment pointer for next element fetch*/var1++;

}

}



int main()

{  intvar_arr[]={11,22,33,44,55,66,77};
myfuncn(var_arr,7);
return0;

}
```

Output:

```
Value of var_arr[0]is:11Value of
var_arr[1]is:22       Value        of
var_arr[2]is:33       Value        of
var_arr[3]is:44       Value        of
var_arr[4]is:55       Value        of
var_arr[5]is:66
```

```
Value of var_arr[6]is:77
```

Passing pointer to function:-

In this example, we are passing a pointer to a function. When we pass apointer as an argument instead of a variable then the address of the variable is passed instead of the value. So any change made by the function using the pointer is permanently made at the address of passedvariable. This technique is known as call by reference in C.

```
#include<stdio.h>
voidsalaryhike(int *var,
int b)

{

   *var = *var+b;

}

int main()

{

int salary=0, bonus=0;

printf("Enter the employee current
salary:");scanf("%d", &salary);

printf("Enter bonus:");
scanf("%d", &bonus);
salaryhike(&salary, bonus);
printf("Final salary: %d",
salary);return0;

}
```

Output:

```
Enter the employee current
salary:10000Enter bonus:2000

Final salary: 12000
```