Covered Topics Under UNIT-3 and UNIT-4 of "PPS-PROGRAMMING FOR PROBLEM SOLVING (BCS101 / BCS201)"

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

Shweta Tiwari

Class Notes

Published Date: November, 2022

PPS: UNIT-3

Iteration and Looping

Array

PPS: UNIT-4

Functions

Basic of Searching and Sorting Algorithms

FALL SEMESTER, YEAR (I/II sem, 1st yr)

FALL SESSION (2022-23)
(PPS)
MS. SHWETA TIWARI
Published Date: November, 2022

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

By SHWETA TIWARI
Under On: UNIT-3 and UNIT-4

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Units Classified into Chapters

Chapter 7
ARRAYS AND STRINGS

Arrays are group of similar data types. It's used to store a large amount of data. The array elements are stored in contiguous memory locations. In other words, if an array is declared to be of an int type, it can't contain elements that are not of int type.

How to declare an Array:

DataType   variablename[size]

Example:-   int ary[5];

10	20	30	40	50

ary[0] ary[1] ary[2] ary[3] ary[4]

In the above program ary is variable name which hold the 5 elements which have a same name, same data type but with different memory locations. As the memory is allocated sequentially so that data can be easily accessed by increment or decrement by single variable. Arrays always start with index value zero.

Advantages of arrays

1. Array is the simplest data structure.
2. It is easy to create.
3. We can use one name for similar identifier.
4. Arrays use reference type so we need not to use return statement.

Limitations

1. Array may require large amount of contiguous memory, which is not available sometimes.

2.Bound checking is not done by 'C' compiler. Means in ary[5] the index value should be between 0 to 4 but if we write ary[7],ary[34] etc compiler does not show any error and display garbage value.

3. Overflow condition may occur in array if we want to store more values then its size.

ARRAY TYPES

Array is two types

1) One dimensional array
2) Multidimensional array

1) **One dimension array:-** A list of items have only one subscript and such a list is called one dimensional array.

Declaration of an array.

int a[5];

In the above example we defined an array a [5]. Here a is a variable name. Data type of a is integer and 5 is the size of an array. Size is always written in subscript []. This means that it can store only five elements. Index of a is start from 0 to 4.

Initialization of an array:- we can also assign values to an array when we declare it.

int a[]={1,2,3,4,5};

compiler automatically sets the size of the array like in this case size of a is 5 and it requires 10bytes of memory[2 for each int]

int a[10]={1,2,3,4,5,6,7,8,9,10};

In this we assign values to the ten elements of a. We can access array elements with its index, for example, if we want to print 5th element from the list we use a[4].

Write a program to print five elements of an array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[]={1,2,3,4,5};
clrscr();
printf("%d",a[0]);
printf("%d",a[1]);
printf("%d",a[2]);
printf("%d",a[3]);
printf("%d",a[4]);
getch();
}
```

In this program 1 is store at location 0,2 at location 1 similarly others. When we print a[0] it will display 1.

Write a program to input and print five elements using array.

***it is always helpful to use loop in array*

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],i;
clrscr();
for(i=0;i<=4;i++) /*it will execute scanf statement 5 times and
value in array*/
{
printf("enter array element=");
scanf("%d",&a[i]);
}
for(i=0;i<=4;i++)
{
printf(" %d",a[i]);
}
getch();
}
```

2)**Multidimensional Array: -** Array in 'C' can have more than one dimension. An array with two dimensions or more is known as multidimensional array. Multidimensional array can be of 2-Dimensional , 3-Dimensional, 4-Dimensional and so on.

```
int  a[2][2];              \\in this a is 2-D array
int  a[2][3][2];    \\in this a is 3-D array
        int  a[2][4][4][2];        \\in this a is 4-D array
```

2-D Array: - 2-Dimensional array is also known as matrix because in this elements are stored in the form of rows and columns. A[4][5] means A is an array with 4 rows and 5 columns. We can store 20 elements in A[4][5] array and size of A is 40bytes.

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]
[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]
A[3][0]	A[3][1]	A[3][2]	A[3][3]	A[3][4]

Nested loops are used to input and print data from 2-D array. Outer loop is used to represent rows and inner is used to represent column.

Write a program to print 2*3array matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][3],i,j;
clrscr();
printf("enter array matrix");
for(i=0;i<2;i++)
{
    for(j=0;j<3;j++)
            {
                    scanf("%d",&a[i][j]);
            }
}
for(i=0;i<2;i++)
{
    for(j=0;j<3;j++)
            {
                    printf(" %d",a[i][j]);
            }
                    printf("\n");
}
getch();
}
```

Write a program to add two matrices.

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

```
int a[5][5] , b[5][5],c[5][5],i,j;
clrscr();
printf("enter 1st matrix");
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
            {
                    scanf("%d",&a[i][j]);
            }
}
printf("enter 2nd matrix");
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
            {
                    scanf("%d",&b[i][j]);



for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
            {
                    c[i][j]=a[i][j]+b[i][j];
                    printf(" %d",c[i][j]);
            }
                    printf("\n");
}
getch();
}
```

a

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

b

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

c

2	3	4	5	6
2	3	4	5	6
2	3	4	5	6
2	3	4	5	6
2	3	4	5	6

Write a program to multiply two matrices.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5] , b[5][5],c[5][5],i,j,k;
clrscr();
printf("enter 1st matrix");
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
              {
                        scanf("%d",&a[i][j]);
              }
}
printf("enter 2nd matrix");
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
              {
                        scanf("%d",&b[i][j]);
              }
}
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
              {
                        c[i][j]=0;
                        for(k=0;k<=4;k++)
                                  {
                        c[i][j]=c[i][j]+a[i][k]*b[k][j];
                                  }
              }
}
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
              {
                        printf("%d",c[i][j]);
              }
                        printf("\n");
}
getch(); }
```

Write a program to print transpose of matrix.

Transpose means converting the rows into column;

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5],i,j;
clrscr();
printf("enter array matrix");
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
```

```
              {
                        scanf("%d",&a[i][j]);
              }
}
for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
              {
                        printf(" %d",a[j][i]);
              }
                        printf("\n");
}
getch();
}
```

A

1	**2**	**3**	**4**	**5**
6	**7**	**8**	**9**	**10**
11	**12**	**13**	**14**	**15**
16	**17**	**18**	**19**	**20**

B

1	**5**	**9**	**13**	**17**
2	**6**	**10**	**14**	**18**
3	**7**	**11**	**15**	**19**
4	**8**	**12**	**16**	**20**

3-Dimensonal array:- eg int A[4][5][3]. In this we can store total 4*5*3=60 elements and size of A is 120 bytes.

Write a program of 3-D Array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][5][4],i,j,k;
clrscr();
printf("enter array matrix");
for(i=0;i<=2;i++)
{
    for(j=0;j<=4;j++)
              {
                        for(k=0;k<=3;k++)
                        {
                        scanf("%d",&a[i][j][k]);
                        }
              }
}
for(i=0;i<=2;i++)
{
    for(j=0;j<=4;j++)
              {
                        for(k=0;k<=3;k++)
```

```
                    {
                    printf("%d",a[j][i][k]);
                    }
            }
}
getch();
}
```

String

C language does not provide any data type in which we can represent collection of character. So if we want to do this we use character array that is called string. As we can store more than one element in integer or float array, similarly we can store more than one character in character array. Group of characters is also known as String.

A string is a one dimensional array of characters terminated by a NULL('\0').

For example,

char name[20]={'P','R','A','T','H','A','M'};

or

char name[20]={"PRATHAM"};

The size of name is 20 bytes(1 byte for 1 char) and we use only first three bytes to store "PRATHAM" after this we have 13 bytes left that contain garbage value. But when we print it, it display first three characters only because compiler place NULL also represented as '\0' at the end of the string, so when compiler print the string it print up to NULL character. We can input and print string using %s format specifier in scanf and printf functions. We can also use gets to input a string and puts to print a string.

Write a program to input and print a string.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch[20];
clrscr();
printf("Enter name ");
scanf("%s",&ch);   // or we can use gets(ch);
printf("Name=%s",ch);
getch();
}
```

```
Output: [if we use scanf]
Enter name Charanjiv Singh
Name=Charanjiv
Output:if we use gets]
Enter name Charanjiv Singh
Name=Charanjiv Singh
```

2-D Character Array

we can use two-dimensional char array to create an array of strings.

For example; char names[10][20];
it can be used to represent ten different strings
For example,
char days[7][10]={"Monday", "Tuesday"...........”Sunday”};
can be used to store name of days.
Here 7 means we pass 7 strings and 10 means each string consist 10 charcaters.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i;
char  name[12][10]={"Jan", "feb", ",mar", "apr", "may", "jun",
"jul", "aug", "sep", "oct", "nov", "dec"};
int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
clrscr();
for(i=0;i<12;i++)
 printf(" %s has %d days:",name[i],days[i]);
getch();
}
```

String library functions are the functions, which are used regularly and stored in library file and whenever these functions are needed , you need to include the required header file in your program like as,
#include<string.h>

There are 7 inbuilt functions to perform different operations on strings.

1.. strlen: This function is used to count the characters in a string. It calculates the length of the string.

Syntax:
strlen(array variable);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s1[20];
int i;
clrscr();
printf("Enter string ");
scanf("%s",&s1);
i=strlen(s1);
printf("Length=%d",i);
getch();
}
```

2. strcpy: This function copies the contents of one string into another string.

Syntax

strcpy(target,source);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s1[20],s2[20];
clrscr();
printf("Enter string ");
scanf("%s",&s1);
strcpy(s2,s1);
printf("string =%s ",s2);
getch();
}
```

3. strcat: This function is used to concatenate the source string at the end of the target string.

Syntax

strcat(target,sourcetoadd);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s1[20],s2[20];
int i;
clrscr();
printf("Enter string ");
scanf("%s",&s1);
printf("Enter string ");
scanf("%s",&s2);
strcat(s2,s1);
printf(" concatenated string =%s ",s2);
getch();
}
```

4. strcmp: This function compares two strings to find out whether they are same or different. The two strings are compared character by character until there is a mismatch or end of the string. This function returns 0 if both the strings are identical.

Syntax

Strcmp(string1,string2);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s1[20],s2[20];
int i;
clrscr();
printf("Enter string ");
scanf("%s",&s1);
printf("Enter string ");
scanf("%s",&s2);
i=strcmp(s2,s1);
if(i==0)
printf(" Both strings are same");
else
printf(" Strings are not same");
getch();
}
```

5. **strrev() : - String reverse function** is used to reverse the string. This function also contain a single argument
6. **strupr() : - String upper case** is used to convert the string in to upper case or in capital letter. This function contains single argument.
7. **strlwr() : - String lower case** is used to convert the string into lower case or in small letter. This function contains single argument.

atoi()[define in stdlib header file: This function comes under <stdlib.h> header file. This function convert numeric string into integer

Syntax

int atoi(string);

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
char s1[20];
int i;
clrscr();
printf("Enter string ");
scanf("%s",&s1);
i=atoi(s1);
printf(" i=%d",i);
getch();
}
```

String can also be accessed as array like we can create programs without using string functions.

WAP to print length of string without using inbuilt function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char s1[20];
int i;
clrscr();
printf("Enter string ");
scanf("%s",&s1); i=0;
while(s1[i]!=NULL)
{
i++;
}
printf(" length of string =%d ",i);
getch();
}
```

**Similarly we can also create other string functions.*

Important Questions[Short]

1. Define array.
2. How we can declare and initialize an array?
3. Define 2D array.
4. Define string.
5. Explain subscripted elements.

Important Questions[Long]

1. Explain memory representation of array.
2. Explain String with example.
3. Explain inbuilt functions of string.

**** Perform Programs from 52 to 72 from program list**

Chapter 8

<u>FUNCTIONS</u>

Functions are the blocks of C programs which perform the specifc task. Functions are used to divide the long program into sub program. In general, a function is a block of code that performs one or more actions. Functions are called by other functions. A program in the C language consists of several functions. One such function we use is the main () function. Every function has a name. The name of the function ends with a pair of parenthesis. This parenthesis may or may not contain any thing. Whatever is inside the parenthesis is known as the arguments of the function.
While defining function you must know these things: -
1. Function name.
2. Function return type (if there is no return then use void)
3. Function argument(if there is no argument here also use void).

Syntax to define a function:
<Return type> <function name>(<arguments>);
Example: -
```
void  show();
```
Why we use function: -

1. Functions are used to find the errors in the program.
2. Functions are used for reusability.
3. It decreases the cost of the software.
4. It increases the efficiency.
5. It makes the programming simple.
6. It reduces the complexity of the program.

Types of Functions
There are two types of functions. First are known as built in functions and second are the user defined functions.

1. **Built in functions**: - Built in functions are those functions, which are provided to us by the programming language. But before using these functions we have to include some header files like,
<stdio.h> which are provided to us by the various input output related functions like printf() and scanf() function. Others are string handling functions, mathematical functions which are stored in
<string.h> and <math.h> header file respectively.

Math Functions
Some examples of Math functions
Basic Math functions are sqrt() ,pow(), cos(), tan() and sin() etc.

1. **sqrt ():-** This function returns the square root of a number. This function accepts a single argument. As shown in the syntax and example given below: -

The syntax for the sqrt() function is:-
```
# include <math.h>
int sqrt(int  h);
```

Here, the sqrt() function returns the non-negative square root of h in the integer data type. This function returns an error if h is negative.

Example of sqrt() function:-

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int n;
clrscr();
printf("enter any number");
scanf("%d",&n);
printf("square root of n=%d",sqrt(n));
getch();
}
```

2. **pow():-** This function returns the power of a number. This function accepts two arguments ,first is the number and the second is its power. As shown in the syntax and example: -

Syntax: -

```
#include <math.h>
int pow(int x, int y);
```

Here, the value of the int variable x is raised to the power of y. The pow() function returns the result in the int data type.

Example of pow() function:-

```
#include<stdio.h>
#include<conio..h>
#include<math.h>
void main()
{
int n,p;
clrscr();
printf("enter any number");
scanf("%d".&n);
printf("enter their power");
scanf("%d",&p);
printf("power of n=%d",pow(n,p));
getch();
}
```

String handling functions

We also use string-handling functions, which are used for string manipulations. These functions are like strcpy(), strcat(), strupr(),strlen(), strrev(), strlwr() and strcmp() etc.

2. **User defined Functions: -** These functions are defined by the user are according to their requirement, known as User define functions. User defined functions have four types: -

   a) **Without Return Without Arguments**
   b) **Without Return with Arguments**
   c) **With Return Without Argument**
   d) **With Return With Argument**

For creating user defined functions we have to follow three basic steps. These are given below: -

a) **Function Declaration / Function Prototyping**
b) **Function Definition**
c) **Function Calling**

a) **Function declaration: -** Function declaration is also known as function prototyping. Before using functions in your program it must be declared and defined. Function declaration or prototyping tells the compiler that this function is used later in the programming. It also tells the compiler their name, return type, and arguments of the function. No function can be called from any other function that hasn't first been declared. The declaration of a function is also called function prototype.A prototype always ends with a semicolon (;).Function is declared same as we declare variables before using them.
b) **Function Definition:** - In function definition we can define the body of the function. It means we can write the number of statements inside the opening and closing brace of the function, which are executed when the function is called by some other function. A function definition must match (their name, return type and parameter list) with its declaration. The definition tells the compiler how the function works.
c) **Function calling: -** Function calling means to execute the function. After declaration and definition of the function we can call the function for their execution. When you call a function, execution begins with the first statement to last statement of the function. Functions can also call other functions and can even call themselves but mostly the main function call the sub function.

Categories of user defined functions:

1. Without Return Without Argument: - This is the first type of user define function and also known as no return no argument. No return means that function start with **'void'** keyword. It tells the compiler that the function cannot return any value. No arguments mean we cannot pass anything from the calling function to called function. It means we can leave the parentheses blank. In C, the declaration of the show () function can be something like this:
Void show();
We can also write it as;

void show(void);
Note that the keyword **'void'** is used in the declaration to indicate to the compiler that no argument is passed to the function. The compiler will produce an error message if an argument is passed to show () function later in a program when this function is called.

Example1.WAP to add and subtract two numbers using function with no return no argument.

As show in the Example: -

```
#include<stdio.h>
#include<conio.h>
void sum();                              //Function declaration
void sub();
void main()
{
clrscr();
sum();                        //Function calling
sub();
getch();
}
void sum()                               ///function Definition
{
int a,b,c;
printf("enter first number");
scanf("%d",&a);
printf("enter second number");
scanf("%d",&b);
c=a+b;
printf("sum=%d",c);
}
void sub()                    ///function Definition
{
int a,b,c;
printf("enter first number");
scanf("%d",&a);
printf("enter second number");
scanf("%d",&b);
c=a-b;
printf("sub=%d",c);
}
```

2. Without Return with argument:- This is the second type of user define functions. This function starts with void keyword, which tells the compiler that function hasn't any return type. It can't return any thing to its calling program. But with argument means we must pass information to the calling function. In this we should declare variables inside the parentheses we cannot leave the parentheses blank. The argument of a function is placed between the parentheses after the function name. If a function has more than one argument then arguments must be separated by commas. We can declare a function with a fixed number of arguments, but then you need to specify the data type of each argument.

Arguments are of two types: -

1) **Formal arguments**
2) **Actual arguments**

1. Formal Arguments:-Formal arguments are those arguments which are declared at function declaration and function definition time. For example:

void sum(int a,int b);

Here a and b are the formal arguments which are passed into the sum function.

2. Actual arguments:-Actual arguments are those arguments which are passed to function at calling time. For example

sum(3,5);

Here 3 and 5 are our actual arguments which are passed into the sum() function.

As Shown in the example:-

Example 2. WAP to find the square and cube of a number using function with argument.

```
#include<stdio.h>
#include<conio.h>
void sqr(int n);              //function declaration
void cube(int n);
void main()
{
int a;
clrscr();
printf("enter any number");
scanf("%d",&a);
sqr(a);                       //calling the function
cube(a);                //here a is actual parameter
getch();
}
void sqr(int n)
{
int m;
m=n*n;
printf("\n the square=%d",m);
}
void cube(int n)
{
int m;
m=n*n*n;
printf("\n the cube=%d",m);
}
```

Local variables: - Local variables are those variables, which are declared inside a block or function. These variables are accessible only within the block and are passed as argument into the function or declared inside the function. These are also known as function level variable because they are accessible only within the function.

Global variables: - Global variables are those variables, which are declared outside the block or function. These are known as global variables. Which is accessible by each block of the program.

3. **With return without Argument: -** Return type functions are those functions, which return a value to the calling function. These functions start with a specific data type in place of void keyword. These type of functions use return keyword to send the data back to the calling function.

As shown in the example:-

Example 3. WAP to find the factorial of a given number using function with return type.

```
#include<stdio.h>
#include<conio.h>
int fact();                    //Function declaration
void main()
{
int f;
clrscr();
f=fact();             //calling the function
printf("\n the fatorial=%d",f);
getch();
}
int fact()            // Function definition
{
int a,b=1,n;
printf("enter any number");
scanf("%d",&n);
for(a=1;a<=n;a++)
{
b=b*a;
}
return b;
}
```

4. **With return with argument: -** This function contains argument and also return the value to its calling function, as show in the given example :.

Example 4. WAP to reverse a number using function with return and with argument.

```
#include<stdio.h>
#include<conio.h>
int reverse(int n);                     //Function declaration
void main()
{
int a,b;
clrscr();
printf("please enter any number");
scanf("%d",&a);
b=reverse(a);                           // Function calling
printf("\n Reverse of a number=%d",b);
getch();
}
int reverse(int n)              ///Function definition
{
int r,s=0;
while(n>0)
{
r=n%10;
s=s*10+r;
n=n/10;
}
return s; }
```

Recursive Functions

Recursive functions are those functions, which call itself again and again. Normally function is called by the other function, but in recursive function it call itself repeatedly. This process known as recursion, as shown in the example below:

Example . WAP to find the factorial of a given number using recursion.

```
#include<stdio.h>
#include<conio.h>
int fact(int n);
void main()
{
int n,f;
clrscr();
printf("enter any number");
scanf("%d",&n);
f=fact(n);
printf("factorial=%d",f);
getch();
}
int fact(int n)
{
if(a==1)
{
return   1;
}
else
{
return n*fact(n-1);
}
}
```

Example:WAP to print the fibonacci series.

```
#include<stdio.h>
#include<conio.h>
int fibo(int a,int b);
void main()
{
clrscr();
fibo(0,1);
getch();
}
int fibo(int a,int b)
{
printf("%d",a);
if(a>100)
{
return 0;
}
fibo(b,a+b);
}
```

Call by Value

In C language by default, the arguments are passed to the function by value. It means the actual arguments copy their value into the formal arguments. In call by value, we can pass the value. If any change is done into the formal arguments, it doesn’t affect the actual arguments because both the arguments use the different memory location.

As show in the example:-

```
#include<stdio.h>
#include<conio.h>
void show(int a);     //declaring the function (a is our formal
                                                     argument)
void main()
{
int x;
clrscr();
printf("enter any number");
scanf("%d",&x);
show(x);  //calling the function ( x is our actual arguments)
printf("\n now value of x=%d",x);
getch();
}
void show(int a)   // function definition ( a is our formal
                                              argument)
{
a=a+1;
printf("\n the value of a=%d",a);
}
```

Output

```
       Enter any Number = 5
The value of a = 6
       Now value of x = 5
```

Call by reference

In call by reference we can pass the address of a variable in place of value. It means we can pass the address of actual argument to the formal argument. If any change is done in to the formal arguments ,it also affects the actual arguments, because the address of the actual arguments are passed to the formal arguments .In call by reference, we can pass pointer variable as formal arguments.

As show in the example:-

```
#include<stdio.h>
#include<conio.h>
void show(int *a); //declaring the function (a is our
                              formal argument)
void main()
{
int x;
clrscr();
printf("enter any number");
scanf("%d",&x);
show(&x); ///calling the function ( x is our actual
                              arguments)
printf("\n now value of x=%d",x);
getch();
}
void show(int *a)// function definition ( a is our formal
                                   argument)
{
*a=*a+1;
printf("\n the value of a=%d",*a);
}
```

Output

```
        Enter any Number = 5
The value of a = 6
        Now value of x = 6
```

Passing array as argument into the function

Sometimes it's difficult to call a function that requires number of arguments. One way around this is to store the variables into an array. So we can pass an array as argument into the function. As shown in the example:

```
#include<stdio.h>
#include<conio.h>
void show(int a[]);          //Parameter Array Type
void main()
{
int x[5],i;
clrscr();
for(i=0;i<5;i++)
{
printf("enter any number");
scanf("%d",&x[i]);
}
show(x);
getch();
}
void show(int a[])
{
int i;
for(i=0;i<5;i++)
{
printf("\n%d",a[i]);
}
}
```

Passing structure as argument into the function

We can also pass structure as an argument into the function same as any other variable. As shown in the example:

```
#include<stdio.h>
#include<conio.h>
struct  student
```

```
{
char name[20];
int roll,marks;
};
void show(struct student);
void main()
{
struct student s;
clrscr();
printf("enter student name");
scanf("%s",&s.name);
printf("enter student roll number");
scanf("%d",&s.roll);
printf("enter student marks");
scanf("%d",&s.marks);
show(s);
getch();
}
void show(struct student s)
{
printf("\nstudent name=%s",s.name);
printf("\nstudent roll=%d",s.roll);
printf("\nstudent marks=%d",s.marks);
}
```

Nested Function

Nested function means a function inside a function. But C does not allow nested functions. But we can call one function into another function. It is also known as nested function. As shown in the example:

```
#include<stdio.h>
#include<conio.h>
void sum(int a,int b);
void input(int a,int b);
void input(int a,int b)
{
sum(a,b);
}
void sum(int a,int b)
{
int c;
c=a+b;
printf("the sum=%d",c);
}
void main()
{
int x,y;
clrscr();
printf("enter the value of x");
scanf("%d",&x);
printf("enter the value of y");
scanf("%d",&y);
input(x,y);
getch();
}
```

Important Short Answer type questions

1. What are functions?
2. What are the advantages of functions?
3. What are arguments?
4. What do you mean by built in functions?
5. Write a note on user defined functions?
6. What is recursion?
7. What are the uses of functions?
8. What are the advantages and disadvantages of recursion?

Important Long Answer Type Questions

1. Write a note on user-defined functions. Why are they used?
2. What are the various types of functions supported by C? Give examples for each type of the C functions.
3. What is the relationship between actual arguments and formal arguments? Explain with example?
4. Explain recursion in C with an example?
5. Write a function to find the factorial of a number using with return with argument.
6. Write a function to find the sum of n natural numbers using recursion.
7. Write the difference between Function prototyping and Function definition.
8. Write the two ways in which arguments can be passed to function.
9. Write a program to pass an array as an argument into the function.
10. Write a function to reverse a given integer using with return with argument?
11. Write a program to display the first n terms of Fibonacci series using recursion.