FUNCTION

Q.What is function ?why is it necessary in programming? Exam:ACCE-2013

Ans: Function is a group of statements that together perform task.

Or

Functions are "self contained" modules of code that accomplish a specific task. Functions usually "take in" data, process it, and "return" a result. Once a function is written, it can be used over and over and over again. Functions can be "called" from the inside of other functions.

Necessary of function:

Functions allow us to conceive of our program as a bunch of sub-steps. (Each sub-step can be its own function. When any program seems too hard, just break the overall program into sub-steps!)

Functions are used because of following reasons –

a) To improve the readability of code.

b) Improves the reusability of the code, same function can be used in any program rather than writing the same code from scratch.

c) Debugging of the code would be easier if you use functions as errors are easy to be traced.

d) Reduces the size of the code, duplicate set of statements are replaced by function calls.

Functions allow us to test small parts of our program in isolation from the rest. This is especially true in interpreted langaues, such as Matlab, but can be useful in C, Java, ActionScript, etc.

Q.What is a function prototype and what are the benefits of it? Exam:ACCE-2013

Ans:

a function prototype declares a function before it is used and prior to its definition. A prototype consists of a functions' name, its return type and its parameter list. It is terminated by a semicolon. The compiler needs to know this information in order for it to properly execute a call to the function.

For example:

```
Int myfunc()
 {
Printf("this is a test");
}
Its prototyps is
Int myfunc();
```

The only function that does not need a prototype is main() since it is predefined by the c language.

Q. Distinguish between automatic and static variable?

Ans:

Static variable	Auto variable
We have to specify the storage class to make a variable static.	It is the default storage class.
If it is not assigned any value then it will give 0 as out put.	If it is not assigned any value then it will give garbage value as output.
It is visible to the block in which it is declared and also in the function where it will passed.	It is visible to the block in which the variable is declared.
It retains its value between different function calls. It holds its last value.	It retains its value till the control remains in the block in which the variable is declared.
Static variable should be compile by compiler first.	Auto variable will compile by the compiler after the static variable.

February 1, 2016

(2)

Q. Distinguish between Global and extern variable.

Ans:

Global variable: A variable whose value can be accessed and modified by any statement in a program, not merely within a single routine in which is defined.

Extern variable: Global variable known to all functions in the file.

Q. What is difference between local and global variable? Exam.Acce 2014,13

Ans:

Local	Global
1.Locals are defined inside a function.	1. Globals are defined out side a function.
2.Local variables are known only to the block in which it is defined.	2.Global variables are known through out the entire program.
3.To storage for local variable is on the stack.	3. storage for global variable in a fixed region of memory set.

Q. Describe about the function definition?

Ans:

The general form of a function definition in C programming language is as follows:

```
return_type function_name( parameter list )
{
body of the function
}
```

A function definition in C programming language consists of a function header and a function body.

Q.Describe about the several part of function?

Ans:

Here are all the parts of a function:

- **Return Type:** A function may return a value. The return_type is the data type of the value the function returns.
- **Function Name:** This is the actual name of the function. The function name and the parameter list together
- **Parameters:** A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument.
- **Function Body:** The function body contains a collection of statements that define what the function does.

Q.Describe about function Declaration,function arguments and function calling with example?

Ans:

Function Declarations

A function declaration tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts:

```
return_type function_name( parameter list );
```

Suppose we have a function name is max(), following is the function declaration:

```
int max(int num1, int num2);
```

Parameter names are not important in function declaration only their type is required, so following is also valid declaration:

```
int max(int, int);
```

Calling a Function

When a program calls a function, program control is transferred to the called function. A called function performs defined task, and when its return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

Q.Write a program that explain user define function and calling that function from main function?

```
#include <stdio.h>
..... .....function declaration.....
int max(int num1, int num2);
int main ()
{
..... local variable definition .....
int a = 100;
int b = 200;
int ret;
..... calling a function to get max value .....
ret = max(a, b);
printf( "Max value is : %d\n", ret );
return 0;
}
..... function returning the max between two numbers .....
int max(int num1, int num2)
{
int result;
if (num1 > num2)
result = num1;
else
result = num2;
return result
}
```

Function Arguments:

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the formal parameters of the function.

Q.What are the different ways of passing parameters to the functions? Which to use when?
Ans:

- **Call by value** – We send only values to the function as parameters. We choose this if we do not want the actual parameters to be modified with formal parameters but just used.
- **Call by reference** – We send address of the actual parameters instead of values. We choose this if we do want the actual parameters to be modified with formal parameters.

Q. Distinguish between function call by value and function call by value and function call by reference. Explain with example? Exam-acce-2014
Q.What is Call by value and Call by reference describe with example?
Ans:

Function call by value:

The call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
It is the default way of calling a function in C

Example:

```
int increment(int var)
{
  var = var+1;
  return var;
}

int main()
```

```
{
 int num1=20;
 int num2 = increment(num1);
 printf("num1 value is: %d", num1);
 printf("num2 value is: %d", num2);
 return 0;
}
Output:
num1 value is: 20
num2 value is: 21
```

Why did it happen? num1 is still have 20 even after increment operation, why?
The reason is simple, function is called by value in above program, which means num1's value gets copied into var and the variable var got incremented (not variable num1), which later stored in num2, via call.

Function call by reference:

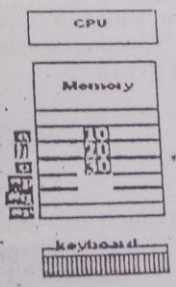
The call by reference method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.

```
int increment(int *var)
{
   *var = *var+1;
   return *var;
}
int main()
{
   int num1=20;
   int num2 = increment(&num1);
   printf("num1 value is: %d", num1);
   printf("num2 value is: %d", num2);
  return 0;
}Output:
num1 value is: 21
num2 value is: 21
```

Unlike "call by value", in this method the value of num1 got changed because the address of num1 is passed as an argument so the increment operation is performed on the value stored at the address.

Figure another example of call by reference:

```
#include <stdio.h>
#include <conio.h>
void add(int *p,int *q,int *s);
void main()
{
   int a,b,c;
   clrscr();
   printf("Enter two numbers\n");
   scanf("%d%d",&a,&b);
   add(&a,&b,&c);
   printf("Sum of %d and %d is %d",a,b,c);
   getch();
}
void add(int *p,int *q,int *s)
{
   *s = *p + *q;
}
Enter two numbers
10
20
Sum of 10 and 20 is 30
```



Q.What is the difference between actual and formal parameters?

Ans:

The parameters sent to the function at calling end are called as actual parameters while at the receiving of the function definition called as formal parameters.

Q.What is the purpose of built-in stricmp() function.

Ans:

It compares two strings by ignoring the case.

Q. Define recursive function with example, state the rules that a recursive function must have to satisfy.
Marks:2.750 Exam-ACCE-2014,2013,ICE-2013,15 CSE-2011,APPE

Ans:

Recursive function:

A function that calls itself is known as recursive function and this technique is known as recursion in C programming

A **recursive function** (DEF) is a **function** which either calls itself or is in a potential cycle of **function** calls. As the definition specifies, there are two types of recursive functions. Consider a function which calls itself: we call this type of recursion **immediate** recursion.

One can view this mathematically in a directed call graph.

```
A ---|
^    |
|    |
|----|
```

```
void A() {
  A();
  return;
} Or.
```

A recursive function is a function that calls itself during its execution. This enables the function to repeat itself several times, outputting the result and the end of each iteration. Below is an example of a recursive function.

```
function Count (integer N)
   if (N <= 0) return "Must be a Positive Integer";
   if (N > 9) return "Counting Completed";
   else return Count (N+1);
end function
```

For better visualization of recursion in this example series sum 1+2+3+4+5:

```
sum(5)
=5+sum(4)
=5+4+sum(3)
=5+4+3+sum(2)
=5+4+3+2+sum(1)
=5+4+3+2+1+sum(0)
=5+4+3+2+1+0
=5+4+3+2+1
=5+4+3+3
=5+4+6
=5+10
=15
```

Rules /condition of Recursive function:

- Every recursive function must have two or more paths.
- One or more of these paths must be non-recursive, which corresponds to the absolute definition.
- Every recursive path must modify its parameter, in order that the recursive call is closer to not recursing than the current call.
- These paths are usually controlled by If statements.

The Factorial Example

The recursive factorial function is quite simple:

```
int fact(int n) {
if(n < 2)
return 1;
return fact(n-1) * n;
}
```

Q. Write a recursive function in C programming that will receive an integer n as input and return the nth number of Fibonacci series. Marks:3 Exam-ACCE-2014,ICE-2013

Ans:

```
#include <stdio.h>

int fibonaci(int i) {

  if(i == 0) {
    return 0;
  }
  if(i == 1) {
    return 1;
  }
  return fibonaci(i-1) + fibonaci(i-2);
}
int main() {
  int i;
  for (i = 0; i < 10; i++) {
    printf("%d\t\n", fibonaci(i));
  }
  return 0;
}
```

When the above code is compiled and executed, it produces the following result –

0 1 1 2 3 5 8 13 21 34

Q.Write a recursive function in C programming that print factorial number?

Ans:

```
#include <stdio.h>
 int fact(int n) {
 if(n < 2)
 return 1;
 return fact(n-1) * n;
 }
int main() {
  int i = 15;
  printf("Factorial of %d is %d\n", i, fact (i));
  return 0;
}
```

Q.Discuss the Advantages and Disadvantages of Recursion?

Ans:

Advantage:

Recursion is more elegant and requires few variables which make program clean. Recursion can be used to replace complex nesting code by dividing the problem into same problem of its sub-type.

Disadvantage:

In other hand, it is hard to think the logic of a recursive function. It is also difficult to debug the code containing recursion.

Q.What is the difference between function declaration and function definition with example?

Ans:

A declaration provides basic attributes of a symbol: its type and its name.

int func();

This is a function declaration; it does not provide the body of the function, but it does tell the compiler that it can use this function and expect that it will be defined somewhere.

. Defining a function means providing a function body;

You can write code like this:

```
int func();
int main()
{
  int x = func();
}
int func()
{
  return 2;
}
```

Since the compiler knows the return value of func, and the number of arguments it takes, it can compile the call to func even though it doesn't yet have the definition. In fact, the definition of the method func could go into another file!

A definition provides all of the details of that symbol--if it's a function, what it does; if it's a class, what fields and methods it has; if it's a variable, where that variable is stored

. If no source file ever defines a symbol, but it is declared, you will get errors at link time complaining about undefined symbols

Q.What are enumerations?

Enumerations are list of integer constants with name. Enumerators are defined with the keyword *enum.*

Q. Describe different categories of function based on their arguments and return type.

Ans:

For better understanding of arguments and return type in functions, user-defined functions can be categorised as:

1. Function with no arguments and no return value
2. Function with no arguments and return value
3. Function with arguments but no return value
4. Function with arguments and return value.

Example of each category:

1.Function with no arguments and no return value

```
#include <stdio.h>
void prime();
int main(){
  prime();    //No argument is passed to prime().
```

```
    return 0;
}
void prime(){
        /* There is no return value to calling function main(). Hence, return
type of prime() is void */
  int num,i,flag=0;
  printf("Enter positive integer enter to check:\n");
  scanf("%d",&num);
  for(i=2;i<=num/2;++i){
    if(num%i==0){
      flag=1;
    }
  }
  if (flag==1)
    printf("%d is not prime",num);
  else
    printf("%d is prime",num);
      }
```

Function prime() is used for asking user a input, check for whether it is prime of not and display it accordingly. No argument is passed and returned form prime() function.

2.Function with no arguments but return value

C program to check whether a number entered by user is prime or not using function with no arguments but having return value

```
#include <stdio.h>
int input();
int main(){
   int num,i,flag = 0;
   num=input();   /* No argument is passed to input() */
   for(i=2; i<=num/2; ++i){
   if(num%i==0){
     flag = 1;
     break;
   }
   }
   if(flag == 1)
     printf("%d is not prime",num);
   else
     printf("%d is prime", num);
   return 0;
}
int input(){    /* Integer value is returned from input() to calling function */
  int n;
  printf("Enter positive integer to check:\n");
  scanf("%d",&n);
  return n;
}
```

There is no argument passed to input() function But, the value of *n* is returned from input() to main() function.

Function with arguments and no return value

Program to check whether a number entered by user is prime or not using function with arguments and no return value

```
#include <stdio.h>
void check_display(int n);
int main(){
```

```
  int num;
  printf("Enter positive enter to check:\n");
  scanf("%d",&num);
  check_display(num);
  return 0;
}
void check_display(int n){
/* There is no return value to calling function. Hence, return type of function
is void. */
  int i, flag = 0;
for(i=2; i<=n/2; ++i){
  if(n%i==0){
    flag = 1;
    break;
  }
  }
  if(flag == 1)
    printf("%d is not prime",n);
  else
    printf("%d is prime", n);
}
```

Here, check_display() function is used for check whether it is prime or not and display it accordingly. Here, argument is passed to user-defined function but, value is not returned from it to calling function.

Function with argument and a return value

Program to check whether a number entered by user is prime or not using function with argument and return value

```
#include <stdio.h>
int check(int n);
int main(){
  int num,num_check=0;
  printf("Enter positive enter to check:\n");
  scanf("%d",&num);
  num_check=check(num); /* Argument num is passed to check()
function. */
  if(num_check==1)
    printf("%d is not prime",num);
  else
    printf("%d is prime",num);
  return 0;
}
int check(int n){
/* Integer value is returned from function check() */
  int i;
  for(i=2;i<=n/2;++i){
  if(n%i==0)
    return 1;
}
  return 0;
}
```

Q.What is function? Discuss different type of function?why need user defined function? Exam:ICE-2013ACCE-2011,CSE

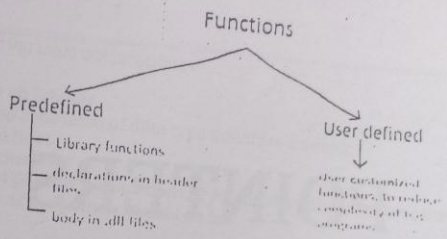
A function is a block of code that performs a particular tas.

February 1, 2016

C functions can be classified into two categories,

- Library functions
- User-defined functions

Library functions are those functions which are defined by C library, example printf(), scanf(), strcat() etc. You just need to include appropriate header files to use these functions. These are already declared and defined in C libraries.



User-defined functions are those functions which are defined by the user at the time of writing program. Functions are made for code reusability and for saving time and space.

Or

The functions which we can create by ourselves, for example in the above code I have created a function abc and I called it in main() in order to use it.

Benefits of Using Functions

1. It provides modularity to the program.
2. Easy code Reuseability. You just have to call the function by its name to use it.
3. In case of large programs with thousands of code lines, debugging and editing becomes easier if you use functions.