JECRC Foundation

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

Year & Semester - B.Tech I year (I Semester)

Subject - Programming for Problem Solving

Presented by - Ms. Abhilasha /Ms. Yogita Punjabi/Mr. Gajendra Sharma

Designation - Asst. Professor

Department - Computer Science (First Year)

VISSION OF INSTITUTE

To become a renowned centre of outcome based learning, and work towards academic, professional, cultural and social enrichment of the lives of individuals and communities.

MISSION OF INSTITUTE

❖**Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.**

❖**Identify, based on informed perception of Indian, regional and global needs, the areas of focus and provide platform to gain knowledge and solutions.**

❖**Offer opportunities for interaction between academia and industry.**

❖**Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders may emerge.**

Programming for Problem Solving : Course Outcomes

Students will be able to:

CO1: Understand concept of low-level and high-level languages, primary and secondary memory. Represent algorithm through flowchart and pseudo code for problem solving.

CO2: Represent and convert numbers & alphabets in various notations.

CO3: Analyze and implement decision making statements and looping.

CO4: Apply pointers, memory allocation and data handling through files in 'C' Programming Language.

Function

A function is a group of statements that together perform a task. Every C program has at least one function, which is **main()**.

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 such case you have two options –

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

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

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

Types of Functions

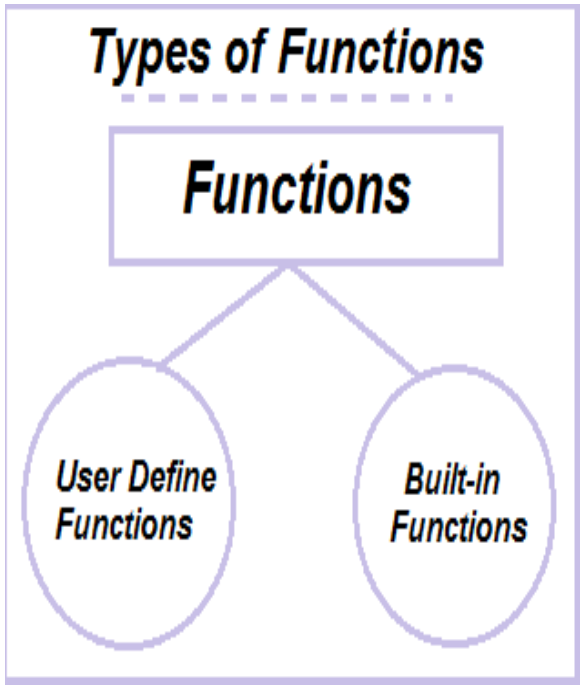
1) Built in function –

puts(), gets(), printf(), scanf() etc –

These are the functions which already have a definition in header files (.h files like stdio.h), so we just call them whenever there is a need to use them.

2) User Defined functions –

The functions that we create in a program are known as user defined functions.



Why we need functions in C

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.

Syntax of a function

return_type function_name (argument list)
{
Set of statements – Block of code
}
Example
int add(int,int);

return_type: Return type can be of any data type such as int, double, char, void, short etc. Don't worry you will understand these terms better once you go through the examples below.
function_name: It can be anything, however it is advised to have a meaningful name for the functions so that it would be easy to understand the purpose of function just by seeing it's name.
argument list: Argument list contains variables names along with their data types. These arguments are kind of inputs for the function. For example – A function which is used to add two integer variables, will be having two integer argument.
Block of code: Set of C statements, which will be executed whenever a call will be made to the function.

How to call a function in C?

Example1: Creating a user defined function addition()

```
#include <stdio.h>
 int addition(int num1, int num2)
{
int sum;
sum = num1+num2; /
 return sum;
}
int main() {
int var1, var2;
printf("Enter number 1: ");
 scanf("%d",&var1);
printf("Enter number 2: ");
scanf("%d",&var2);
int res = addition(var1, var2);
printf ("Output: %d", res);
return 0;
 }
```

Output:

Enter number 1: 100 Enter number 2: 120 Output: 220

Example2: Creating a void user defined function that doesn’t return anything.

```
#include <stdio.h>
void introduction()
 {
printf("Hi\n");
printf("My name is Abhilasha\n");
printf("How are you?");
}
int main()
{
introduction();
 return 0;
}
```

Output:

Hi My name is Abhilasha How are you?

Category of Function

1. NO ARGUMENT NO RETURN VALUES
2. ARGUMENT BUT NO RETURN VALUES
3. NO ARGUMENT WITH RETURN VALUES
4. WITH ARGUMENT WITH RETURN VALUES

NO ARGUMENT NO RETURN VALUES

- In this method, We won't pass any arguments to the function while defining, declaring, or calling the function.
- This type of functions in C will not return any value when we call the function from main() or any sub-function.
- When we are not expecting any return value, but we need some statements to print as output. Then, this type of function in C is very useful.

Example: No Argument No Return Values Addition of Two number.

```
#include<stdio.h>
void Addition();
void main()
 {
printf("\n ..... \n");
Addition();
}
 void Addition()
{
int Sum, a = 10, b = 20;
Sum = a + b;
 printf("\n Sum of a = %d and b = %d is = %d", a, b, Sum);
}
```

Output:
Sum of a=10 and b=20 is 30

Function with No arguments and with Return value

- In this method, We won't pass any arguments to the function while defining, declaring, or calling the function.
- This type of function will return some value when we call the function from main() or any sub function.
- The Data Type of the return value will depend upon the return type of function declaration.
- For instance, if the return type is int then return value will be int.

Example: No Arguments and with Return Value Multiplication of Two number

```
#include<stdio.h>

 int Multiplication();

int main()

{

int Multi;

Multi = Multiplication();

printf("\n Multiplication of a and b is = %d \n", Multi );

return 0;

}

int Multiplication()

{

int Multi, a = 20, b = 40;

Multi = a * b;

return Multi;

 }
```

Output

Multiplication of a and b is = 800

Function with No arguments and with Return value

- In this method, We won't pass any arguments to the function while defining, declaring, or calling the function.
- This type of function will return some value when we call the function from main() or any sub function.
- The Data Type of the return value will depend upon the return type of function declaration.
- For instance, if the return type is int then return value will be int.

Example: No Arguments and with Return Value Mulitiplication of Two no

```
#include<stdio.h>
 int Multiplication();
int main()
{
int Multi;
Multi = Multiplication();
printf("\n Multiplication of a and b is = %d \n", Multi );
return 0;
}
int Multiplication()
{
int Multi, a = 20, b = 40;
Multi = a * b;
return Multi;
 }
```

Output
Multiplication of a and b is = 800

C Function with argument and No Return value

- This method allows us to pass the arguments to the function while calling the function.
- But, This type of function will not return any value when we call the function from main () or any subfunction.
- If we want to allow the user to pass his data to the function arguments, but we are not expecting any return value, this type of function is very useful.

Example: C Function with argument and No Return value Addition of Two number.

```
#include<stdio.h>
void Addition(int, int);
 void main()
{
int a, b;
printf("\n Please Enter two integer values \n");
scanf("%d %d",&a, &b);
Addition(a, b);
}
 void Addition(int a, int b)
{
 int Sum; Sum = a + b;
 printf("\n Additiontion of %d and %d is = %d \n", a, b, Sum);
}
```

OUTPUT

Please enter two integer value
10
20
Addition of 10 and 20 is 30

C Function with argument and Return value

- This method allows us to pass the arguments to the function while calling the function.
- This type of function will return some value when we call the function from main () or any sub function.
- Data Type of the return value will depend upon the return type of function declaration. For instance, if the return type is int then return value will be int.
- This type of user-defined function is called a fully dynamic function, and it provides maximum control to the end-user.

Example :Function with arguments and Return value Multiplication of Two number.

```
#include<stdio.h>
 int Multiplication(int, int);
 int main()
 {
int a, b, Multi;
 printf("\n Please Enter two integer values \n");
scanf("%d %d",&a, &b);
Multi = Multiplication(a, b);
 printf("\n Multiplication of %d and %d is =
%d \n", a, b, Multi);
return 0;
}
 int Multiplication(int a, int b)
{
int Multi; Multi = a * b;
return Multi;
 }
```

OUTPUT

Please Enter two integer values 10 ,20

Multiplication of 10 and 20 is=200

Two ways in which arguments can be passed to a Function

1. Call by Value
2. Call by Reference

PARAMETERS	CALL BY VALUE	CALL BY REFERENCE
Basic	A copy of the variable is passed.	A variable itself is passed.
effect	Change in a copy of variable doesn't modify the original value of variable.	Change in a copy of variable modify the original value of variable.
Syntax	function_name(variable_name1, variable_name2...)	function_name(&variable_name1, &variable_name2...)
Default calling	Primitive type are passed using "call_by_value".	Objects are implicitly passed using "call_by_reference".

Function call by value

Function call by value is the default way of calling a function in C programming.

Actual parameters: The parameters that appear in function calls.

Formal parameters: The parameters that appear in function declarations.

```
                                    // Formal parameters
void add(int num1, int num2) // Function definition
{
    // Function body
}


int main()
{
            // Actual parameters
    add(10, 20); // Function call

    return 0;
}
```

Example of Function call by Value

```
#include <stdio.h>
 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("\nnum2 value is: %d", num2);
return 0; }
```

Output:

num1 value is: 20 num2 value is: 21

Function call by reference

Before we discuss function call by reference, lets understand the terminologies that we will use while explaining this:
Actual parameters: The parameters that appear in function calls.
Formal parameters: The parameters that appear in function declarations.

For example:
int sum(int a, int b);The a and b parameters are **formal parameters**.
We are calling the function like this:
int s = sum(10, 20); //Here 10 and 20 are **actual parameters.**
or
int s = sum(n1, n2); //Here n1 and n2 are **actual parameters.**

Function call by reference

```
#include <stdio.h>
void increment(int *var)
{
*var = *var+1;
 }
 int main()
 {
 int num=20;
increment(&num);
 printf("Value of num is: %d", num);
return 0;
}
```

Output:Value of num

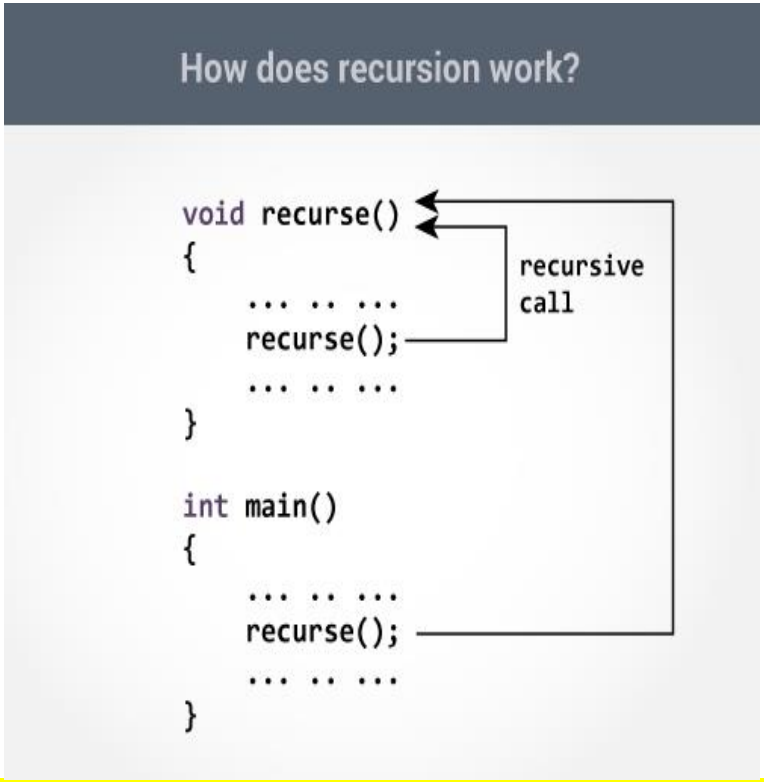
Recursion

How recursion works?

A function that calls itself is known as a recursive function. And, this technique is known as recursion.

How recursion works?

```
void recurse()
{
... .. ... recurse();
... .. ...
}
int main()
{ .
.. .. ... recurse();
... .. ...
}
```



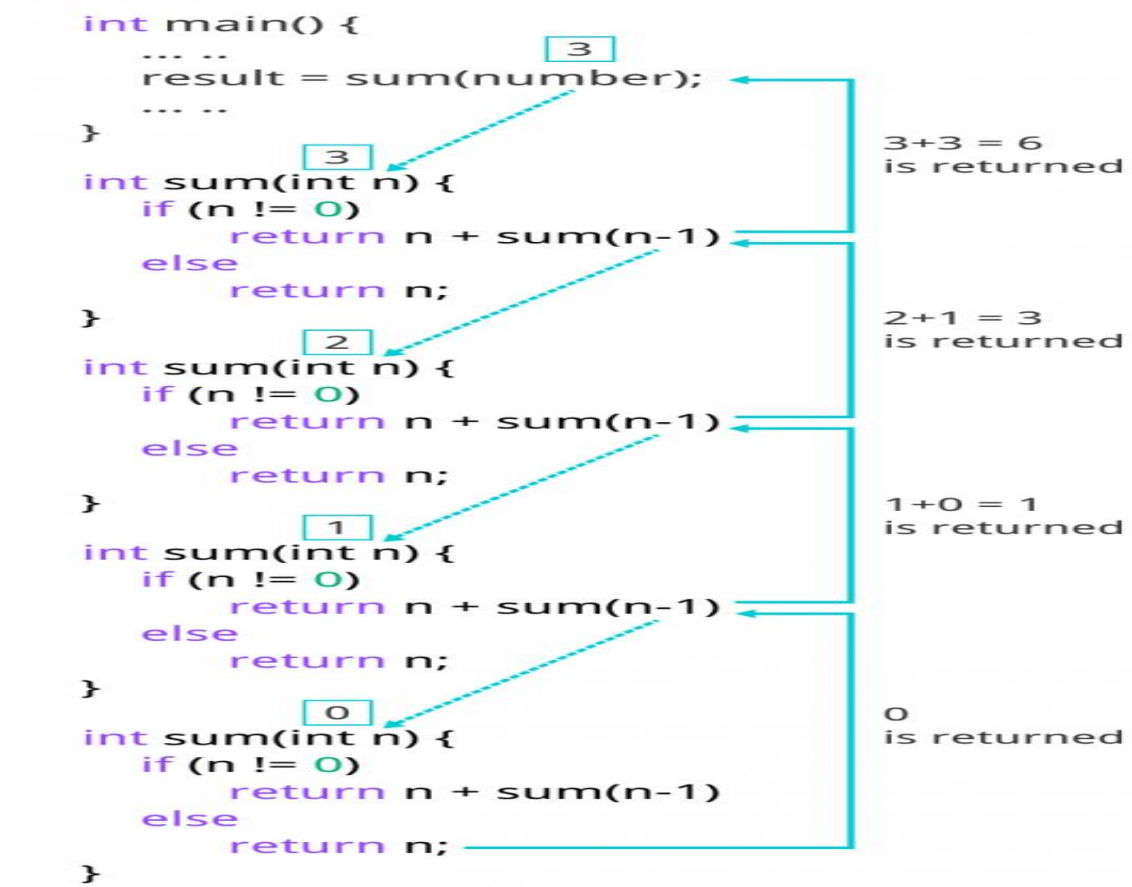
Example: Sum of Natural Numbers Using Recursion

```
#include <stdio.h>
 int sum(int n);
int main()
 {
int number, result;
printf("Enter a positive integer: ");
scanf("%d", &number);
result = sum(number);
printf("sum = %d", result);
return 0;
}
 int sum(int n)
 { if (n != 0)
return n + sum(n-1);
else
return n;
}
```

Output

Enter a positive integer:3 sum = 6

Cont...


int main() {
... ..
result = sum(number);
... ..
}
3
int sum(int n) {
if (n != 0)
return n + sum(n-1)
else
return n;
}
3+3 = 6
is returned
2
int sum(int n) {
if (n != 0)
return n + sum(n-1)
else
return n;
}
2+1 = 3
is returned
1
int sum(int n) {
if (n != 0)
return n + sum(n-1)
else
return n;
}
1+0 = 1
is returned
0
int sum(int n) {
if (n != 0)
return n + sum(n-1)
else
return n;
}
0
is returned

Bibliography

Programming in ANSI C by E.Balagurusamy, McGrawHill

Let us C Yashavant Kanetkar, BPB Publication

https://www.programiz.com/c-programming/c-functions

https://www.tutorialspoint.com/cprogramming/c_functions.htm

JECRC Foundation

JAIPUR ENGINEERING COLLEGE AND RESEARCH CENTRE

Thank you!