Functions

What is function?

- Piece of code to accomplish certain operation.
- It has a name for identification
- They are of two types
  - Predefined functions
  - User defined functions

FUNCTION SYNTAX

```
#include<stdio.h>
    void abc();  //function prototype declaration

void main()
            {
              abc(); //function call
                        printf("I am main line!");
              }

void abc() //function definition
            {
            Printf("I am function body \n");
            }
```

FUNCTION SYNTAX

```
#include<stdio.h>
    void abc();  //function prototype declaration
    void main()
            {
              abc();                    //function call
               printf("I am main line!");
              }
           void abc()      //function definition
                    {
                    Print("I am function body \n");
                    }
```

FUNCTION FOR ADDTION

```
#include<stdio.h>
#include<conio.h>
add(); //declaration
```

```
void main()
{
clrscr();
add(); //calling
getch();
}
```

```
add()  //definition
{
int a,b,c;
printf(“Enter two no. to add");
scanf("%d%d",&a,&b);
c=a+b;
printf("%d",c);
}
```

Benefits of function

- Modularization
- Easy to read
- Easy to debug
- Easy to modify
- Avoids rewriting of same code over and over
- Better memory utilization

FUNCTION FOR ADDTION

```
#include<stdio.h>
#include<conio.h>
add();
sub();
mul();
div();

void main()
{
clrscr();
add();
sub();
mul();
div();
getch();
}

add()
{
int a,b,c;
printf("Enter two no. for Addition");
scanf("%d%d",&a,&b);
c=a+b;
printf("Addition is : %d",c);
}
```

```
sub()
{
int a,b,c;
printf("Enter two no. for Substraction");
scanf("%d%d",&a,&b);
c=a-b;
printf("Substraction is : %d",c); }
```

```
mul()
{
int a,b,c;
printf("Enter two no for Multiplication");
scanf("%d%d",&a,&b);
c=a*b;
printf("Multiplication is : %d",c); }
```

```
div()
{
int a,b,c;
printf("Enter Two no for Division");
scanf("%d%d",&a,&b);
c=a/b;
printf("division is : %d",c);
}
```

Remember

- Program must have at least one function
- Function names must be unique
- Function is an operation, once defined can be used many times.
- One function in the program must be main
- Function consumes memory only when it is invoked and released from RAM as soon as it finishes its job.

FUNCTION CALL

Passing value between functions

- We can communicate between the calling and the called functions by passing the parameters.

There are two ways to pass parameters to a function:

1. **CALL BY VALUE:**
2. **CALL BY REFERENCE:**

Passing value between functions

CALL BY VALUE:

- You actually pass a copy of the variable to the function modifies the copy the original remains unaltered.

CALL BY REFERENCE:

- Call by reference mechanism is used when you want functions to do the changes in called parameters and reflect those changes back to the calling function .
- In this case addresses (pointer) of the variable are passed to function so that function can work directly over the addresses.

CALL BY VALUE

Example - Call by Value

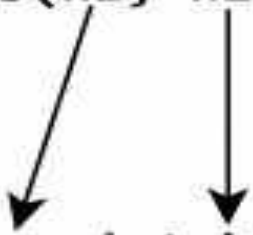
```
#include<stdio.h>
#include<conio.h>
int square(int x);
void main()    //main function
{
int a,b;
clrscr();
printf("enter any value:\n");
scanf("%d",&a);
b=square(a);    //calling by value
printf("square is %d “,b);
getch();   }
```

```
int square(int x)
{
int y;
y=x*x;
return(y);
}
```

PASSING ARGUMENTS TO FUNCTION

How to Pass arguments to a function ?

```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```



Example: User-defined function

```
#include <stdio.h>
int addNumbers(int a, int b); //function prototype
void main()
{
int n1,n2,sum;
printf("Enters two numbers: ");
scanf("%d %d",&n1,&n2);
sum = addNumbers(n1, n2); //function call
printf("sum = %d",sum);
getch();
}
```

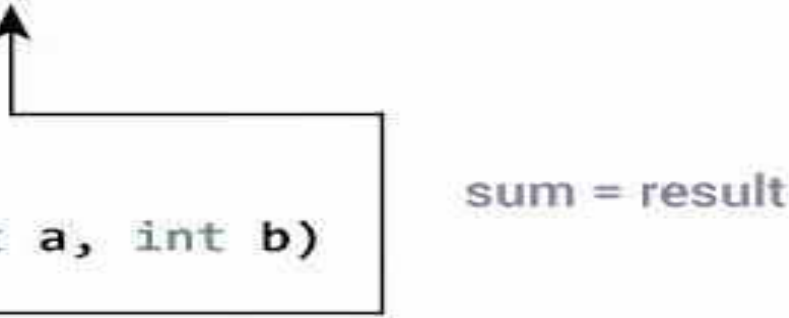
Example: User-defined function

```
int addNumbers(int a, int b)//function definition
{
int result;
result = a+b;
return result; // return statement
}
```

RETURN STATEMENT IN FUNCTION

Return statement of a function

```c
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    return result;
}
```



Example - Call by Value

```
#include <stdio.h>
void swap(int p1,int p2);
void main() //main function
{
int a=10, b=20;
printf("before swaping value of a=%d and b=%d\n",a,b);
swap(a,b); //calling by value
printf("\n\n values of a=%d and b=%d\n",a,b);
getch();
}
```

Example - Call by Value

```
void swap(int p1, int p2) //function method
{
int c;
c =p1;
p1=p2;
p2=c;
printf("\n after swapping value of a(p1)=%d and b(p2)=%d\n",p1,p2);
}
```

CALL BY REFERENCE

Example - Call by Reference

```
void swap(int *p1,int *p2);
void main() //main function
{
int a=10;
int b=20;
printf("\n before value of a:%d and value of b:%d\n",a,b);
swap(&a,&b); //calling by reference
printf("\n value of a(p1):%d and value of b(p2):%d \n",a,b);
getch();
}
```

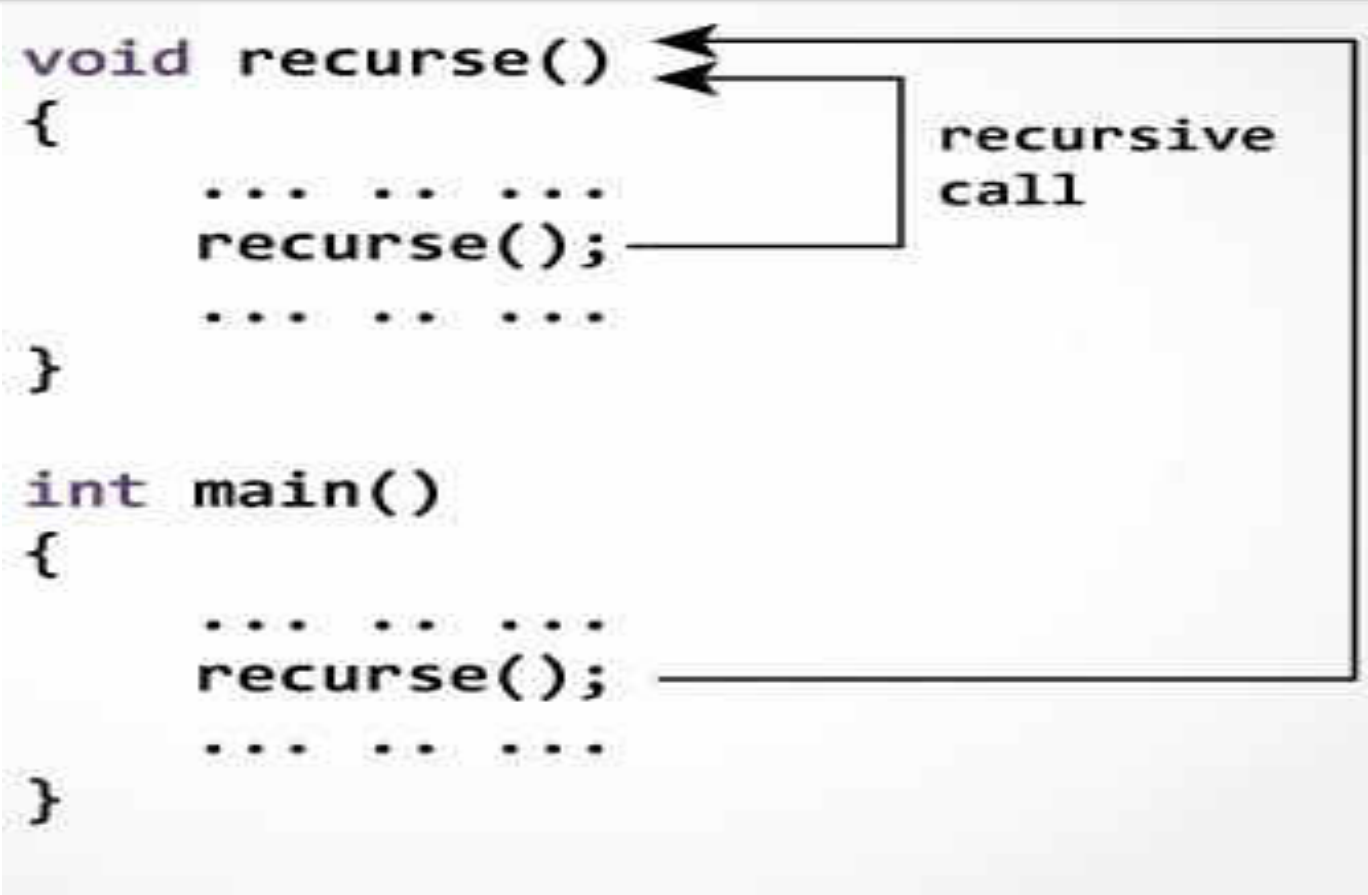
Example - Call by Reference

```
void swap(int *p1,int *p2) //function/method
{
int c;
c = *p1;
*p1 = *p2;
*p2 = c;
printf("after swaping value of a(p1):%d and value of
    b(p2):%d\n",*p1,*p2);
}
```

C Recursion

- A function that calls itself is known as a recursive function. And, this technique is known as recursion.
- The recursion continues until some condition is met to prevent it.
- To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call, and other doesn't.

How does Recursion work ?



Factorial

```
#include <stdio.h>
factorial(int i)
{
        if(i <= 1)
        {
        return 1;
        }
 return i * factorial(i--);
}
```

```
void main()
{
 int i = 5;
printf("Factorial of %d is %d\n", i,
factorial(i));
getch();
}
```

use loops instead of recursion. recursion is usually much slower.

Feature of recursion:

- *there should be at least one if statement used to Terminate recursion.*

It does not contain any looping statement.

Advantage of recursion:

- *It is easy to use*
- *It represent compact programming structures.*
- *It can be applied to calculate factorial of a a,*
- *Fibonacci series.*

Disadvantage of recursion:

- *It is slower than that of looping statement because each time function is called.*

GETC() & PUTC() FUNCTION

Getc( ) & putc( ) function

```
#include<stdio.h>
void main ()
 {
   char c;
   printf("Enter character: ");
   c = getc(stdin);
   printf("Character entered: ");
   putc(c, stdout);
      getch();
}
```

Getc( ) & putc( ) function

```
#include<stdio.h>
#include<conio.h>
void main()
{
char a[]="hello";
clrscr();
int i= 0;
while(a[i])
putc(a[i++],stdout);
getch();
}
```

THANKS