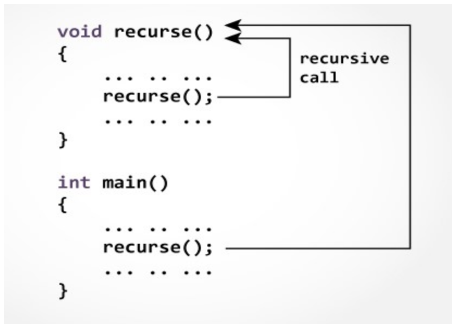
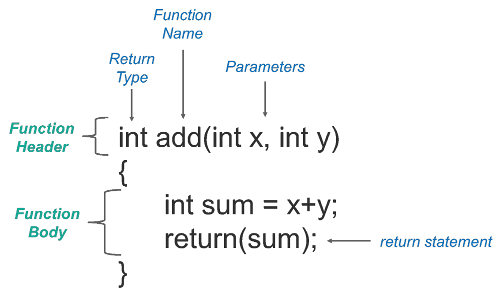
COLLEGE OF ENGINEERING & TECHNOLOGY

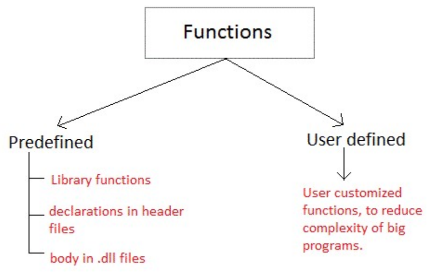
CHAPTER - 6
RECURSION



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

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



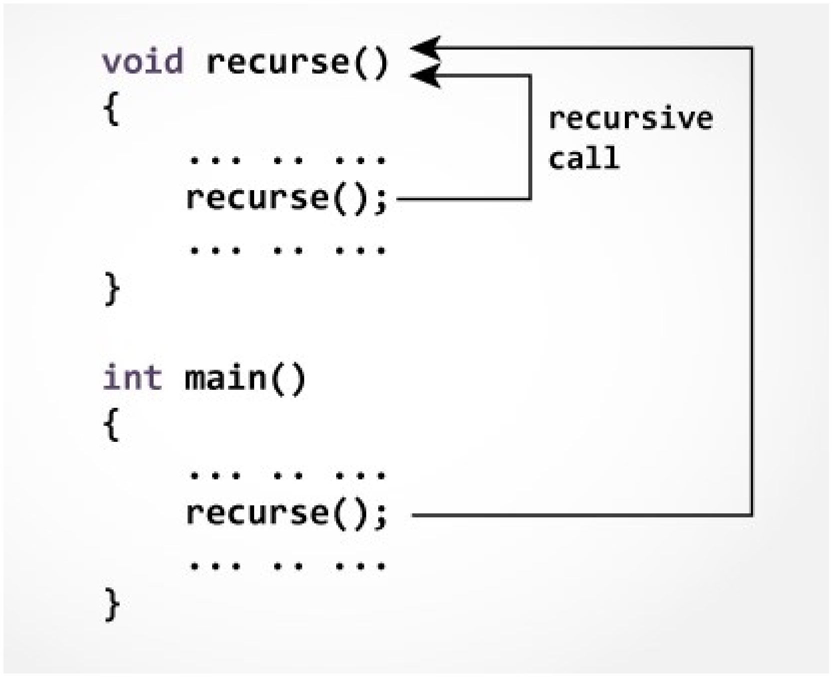
```
int add(int x, int y)
{
    int sum = x+y;
    return(sum);
}
```


Functions
Predefined
User defined
Library functions
declarations in header files
body in .dll files
User customized functions, to reduce complexity of big programs.

Subject : PPS Code : 3110003	**Prepared By:** Asst. Prof. Rupali Patel (CSE Department, ACET)	COLLEGE OF ENGINEERING & TECHNOLOGY

Recursion

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



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

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

Recursion (cont..)

Recursion is a method of solving problems that involves breaking a problem down into smaller and smaller sub-problems until you get to a small enough problem that it can be solved trivially.

Usually recursion involves a function calling itself.

While it may not seem like much on the surface, recursion allows us to write elegant solutions to problems that may otherwise be very difficult to program.

Examples of Recursion

calculate the factorial of a number using recursion.

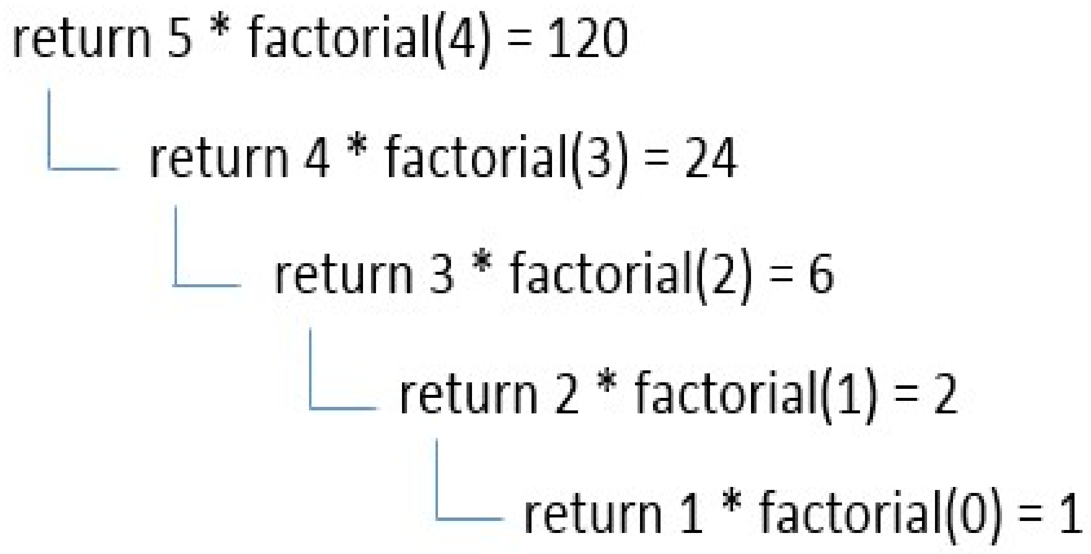
```
#include <stdio.h>
int fact(int);
void main()
{
        int n,f;

        printf("Enter the number
whose factorial you want to calculate?
");
        scanf("%d",&n);
        f = fact(n);  f=120;
        printf("factorial = %d",f);
}
```

```
int  fact(int n)
{
        if (n==0)
        {
                return 0;
        }
        else if ( n == 1)
        {
                return 1;
        }
        else
        {
                return n*fact(n-1);
        }
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Examples of Recursion (cont..)

Explanation of factorial using recursion.



```
return 5 * factorial(4) = 120
  └── return 4 * factorial(3) = 24
        └── return 3 * factorial(2) = 6
              └── return 2 * factorial(1) = 2
                    └── return 1 * factorial(0) = 1
```

Examples of Recursion (cont..)

find sum of all digits using recursion.

```
#include <stdio.h>
 int sumDigits(int num)
{
        static int sum=0;
        if(num>0)
        {
                sum+=(num%10);
                sumDigits(num/10);
        }
        else
        {
                return sum;
        }
}
```

```
int main()
{
        int number,sum;
        printf("Enter a positive 
integer number: ");
        scanf("%d",&number);
        sum=sumDigits(number);
sum=6;
        printf("Sum of all digits are: 
%d\n",sum);
        return 0;
}
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Examples of Recursion (cont..)

find the Fibonacci series using recursion.

```
#include <stdio.h>
int  fibonacci(int i)
{
            if(i == 0)
            {
                        return 0;
            }
            if(i == 1)
            {
                        return 1;
            }
            return fibonacci(i-1) +
                      fibonacci(i-2);
}
```

```
void  main()
{
  int i;
  for (i = 0; i < 10; i++)
    {
             printf("%d \n", fibonacci(i));

     }
 }
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Ackerman function

Ackerman function is one of the simplest and earliest-discovered examples of a total computable function that is not primitive recursive.

All primitive recursive functions are total and computable, but the Ackermann function illustrates that not all total computable functions are primitive recursive.

It's a function with two arguments each of which can be assigned any non-negative integer.

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1,A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

where m and n are non-negative integers

Ackerman function (cont..)

```
#include <stdio.h>
int ack(int m, int n)
{
   if (m == 0)
{
            return n+1;
   }
   else if((m > 0) && (n == 0))
{
 return ack(m-1,1);
}
   else if((m > 0) && (n > 0))
{
      return ack(m-1, ack(m, n-1));
   }
}
```

```
void main()
{
   Int a;
   a = ack(1, 2);        ack(m=1,n=2)
   printf("%d", A);
  }
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Thank you

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY