RECURSION / RECURSIVE FUNCTIONS

It is the process of calling a function itself.

Purpose:

Recursion allows the user to get results , without using loops. Due to this complexity of program is reduced.

Recursion reduce calling of function by the user.

By using recursion, we can control the function calling information or statements.

By using recursion, we can evaluate stack expressions.

Drawbacks:

They are slower than normal functions due to stack over lapping.

They can create stack over flow because of occupying more stack.

Recursion functions will create infinitive loops also.

Eg: 1

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

```
void  main()
{
printf("Welcome to C\n");
main();
}
```

Note: This program causes infinitive loops.

Eg 2: Controlling the above program

```
#include<stdio.h>
#include<conio.h>
int   a=1; /* global variable*/
void main()
{
printf("Welcome to C\n");
a++;
if(a<=3) main();
getch();
}
```

Eg: Finding factorial using recursion:

```
#include<stdio.h>
#include<conio.h>
long  fact(int  n)
{
if(n!=0) return  n * fact(n-1);
else  return 1;
}
void  main()
{
int  n;
clrscr();
printf(“Enter a no “); scanf(“%d”, &n);
printf(“%d  Factorial = %ld”, n, fact(n));
getch();
}
```

O/P: Enter a no 5

5 Factorial = 120

Finding power using recursion:

```
#include<stdio.h>
#include<conio.h>
long power(int b, int p)
{
if(p!=0) return b * power(b, p-1);
else return 1;
}
void main()
{
int b,p;
clrscr();
printf("Enter base, power values ");
scanf("%d %d",&b,&p);
printf("%d ^ %d = %ld", b, p, power(b,p));
getch();
```

```
}
```

Eg : Finding digital sum using recursion

```
#include<stdio.h>
#include<conio.h>
int s=0; /* global var*/
int  dsum(long n)
{
if(n!=0)
{
s=s+n%10;
dsum(n/10);
}
return s;
}
void main()
{
long  n;
```

```
clrscr();
printf("Enter a no");
scanf("%ld",&n);
printf("%ld digital sum = %d",n,dsum(n));
getch();
}
```

Output:

Enter a no: 123

123 digital sum = 6

COMMAND LINE ARGUMENTS

It is the process of sending arguments to the main() from command prompt.

It is useful, in designing the applications for dos/command based environment.

It allows to create new commands for CUI[Character User Interface] environments.

main() by default having 2 arguments.

1. argc
2. argv

argc is an integer, which counts the no of arguments entered at command prompt, including filename.

argv is a string array(pointer), which stores the arguments entered at command prompt.

The Default no of argument counter is 1.

Eg:

Finding n numbers sum using command line arguments:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void  main(int  argc,  char  *argv[])
```

```
{
int i, s=0;
clrscr();
for(i=1;i< argc ;i++)
s = s + atoi(argv[i]);
printf("Sum = %d", s);
getch();
}
```

Save file [eg: sum.c] , compile and ctrl+f9

Goto command prompt.

```
C:\tc> sum  10   20  30
Sum = 60
```

Eg:

Finding factorial using command line arguments

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void   main(int  argc,  char  *argv[])

{

Long int   f=1;

int  n=atoi(argv[1]);

clrscr();

while(n>=1) f=f*n--;

printf("Factorial = %ld", f);

getch();

}
```

Save file [Eg: fact.c],   ctrl+f9

C:\tc> fact 5

Factorial = 120

Eg:

```
/*Finding string length using command line argument*/
#include<stdio.h>
#include<conio.h>
void  main(int  argc,  char *argv[100])
{
int  i;
for(i=0;argv[1][i]!='\0';i++)
{
printf("%2c",argv[1][i]);
}
printf(" Length = %d", i);
```

}

Save the file [eg: len.c]

compile - alt+F9

run - ctrl+f9

goto command prompt.

c:\tc>len   kishore

k i s h o r e length = 7