9 Recursion

"Pride will destroy a person."

A function that calls itself is an important feature of C and such functions are called recursive functions. The term "recursion" was derived from the Latin word *recursus*, which means, "to run back". "Recursive" thinking may be tough for beginners. In this chapter, I have presented some interesting recursive programs. Few programs are my original work, others are improved version of existing recursive programs.

Note

As recursive programs use "memory stack", it reduces execution speed. And it may cause "stack overflow" which would in turn crash your system. If you compile your program with "Test stack overflow" option, you can avoid this problem. For this, choose OPTIONS >COMPILER >ENTRY/EXIT CODE > Test stack overflow.

9.1 Factorial

This is the most famous program on recursion. Many versions of this program are available. All programs differ only in checking conditions. I prefer to write like the following one.

```
long Factorial( int n ) /* returns factorial */
{
   if ( n>0 )
      return( n * Factorial(n-1) );
     else
      return( 1 );
} /*--Factorial( )------*/
```

9.2 Fibonacci

The following program returns the n^{th} Fibonacci number. Fibonacci series is : 1, 1, 2, 3, 5, 8, 13, 21…

```
int Fibonacci( int n )  /* returns nth Fibonacci number */
{
   if ( n==1 || n==2 )
      return( 1 );
    else
      return( Fibonacci(n-1) + Fibonacci(n-2) );
} /*--Fibonacci( )-------*/
```

9.3 GCD

Here is the program to find the Greatest Common Divisior (GCD) of two numbers a & b.

```
int GCD( int a, int b ) /* returns GCD of a, b */
{
   if ( a>=b && a%b==0 )
      return( b );
    else if ( a<b )
      return( GCD( b, a ) );
    else
      return( GCD( b, a%b ) );
} /*--GCD( )---------*/
```

9.4 Power

I haven't yet come across user defined `power` function, which could handle negative n (say, 4.5^{-5}). Here is the program I tried…it could handle negative n too!

```
double Power( double x, int n )     /* returns x power n */
{
   if ( n==0 )
      return( 1 );
     else if ( n>0 )
      return( x * Power( x, n-1 ) );
     else
      return( (1/x) * Power( x, n+1 ) );
} /*--Power( )-----------*/
```

9.5 Reverse Printing

This is a wonderful program to understand the behavior of recursion.

```
void ReverseChar( void )      /* prints characters in reverse */
{
   char ch;
   if ( (ch=getchar( ))!='\n' )
      ReverseChar( );
   putchar( ch );
} /*--ReverseChar( )---------*/
```

9.6 Decimal to binary conversion

The following recursive function gets decimal value as input and prints binary value. It prints each bit value (0 or 1) one by one.

```
void ToBin( int n )     /* prints decimal in binary */
{
   if (n>1)
      ToBin( n/2 );
   printf( "%d", n%2 );
} /*--ToBin( )---------*/
```

9.7 Decimal to hexadecimal conversion

```
void ToHex( int n )     /* prints decimal in hexadecimal */
{
   char *htab[ ] = { "0", "1", "2", "3", "4", "5", "6", "7", "8",
                     "9", "A", "B", "C", "D", "E", "F" };
   if (n>15)
      ToHex( n/16 );
   printf( "%s", htab[n%16] );
} /*--ToHex( )-----------*/
```

9.8 Printing a decimal in words

The following recursive function gets a decimal number as argument and prints it in words. For example, `12340` will be printed as `One Two Three Four Zero`.

```
void InWord( int n )    /* prints decimal in words */
{
   char *wtab[ ] = { "Zero", "One", "Two", "Three", "Four",
                     "Five", "Six", "Seven", "Eight", "Nine" };
   if (n>9)
      InWord( n/10 );
   printf( "%s ", wtab[n%10] );
} /*--InWord( )----------*/
```