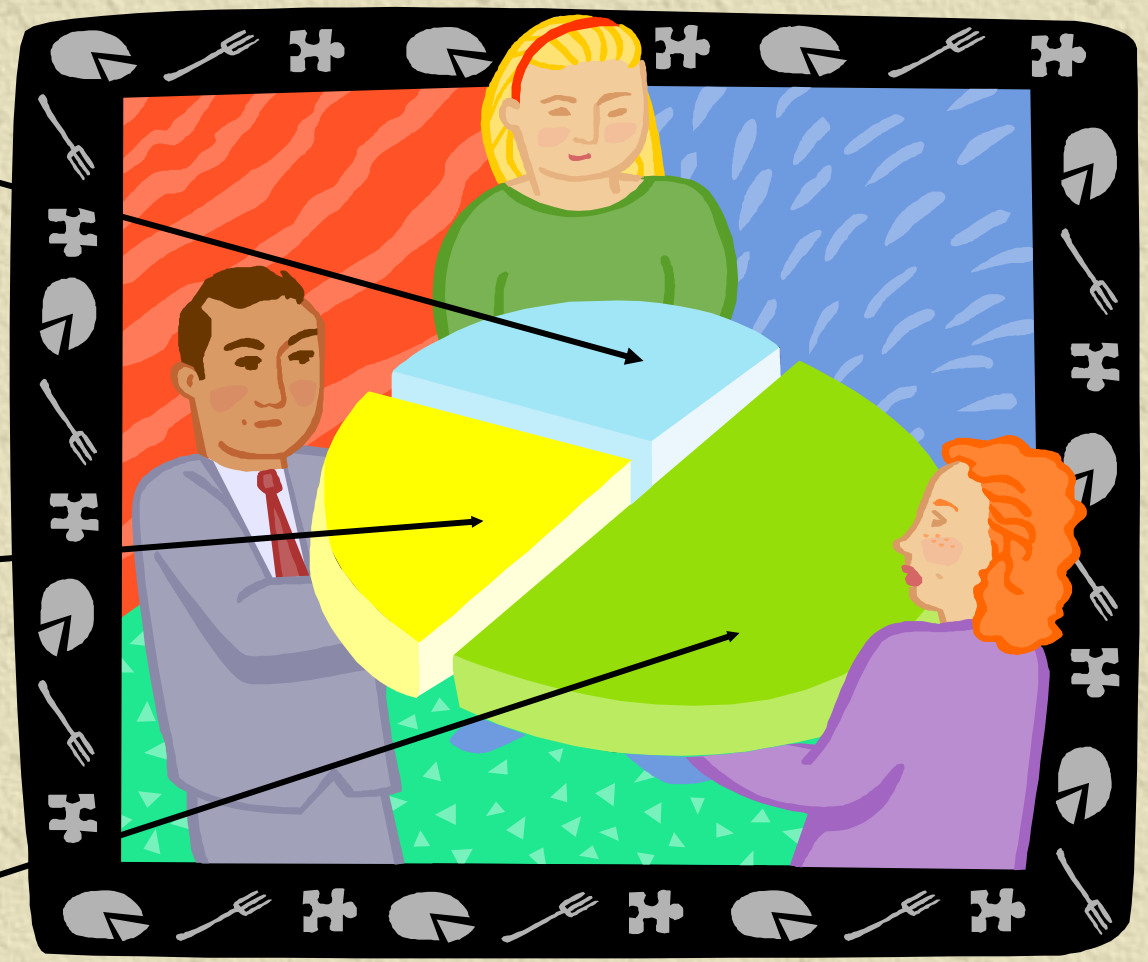
PROGRAMACIÓN MODULAR

Lic. Gladys Chuquimia

¿En qué consiste la programación modular?

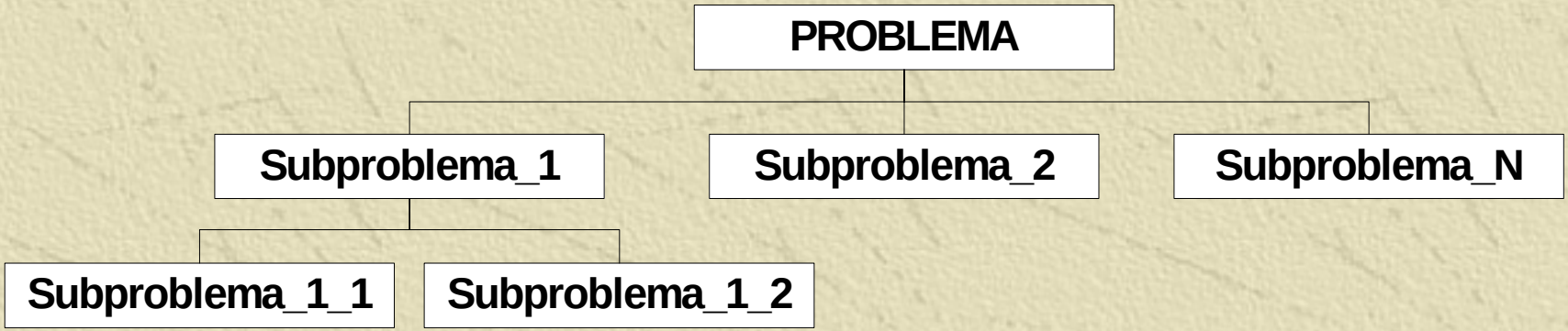
MODULOS

Subproblemas

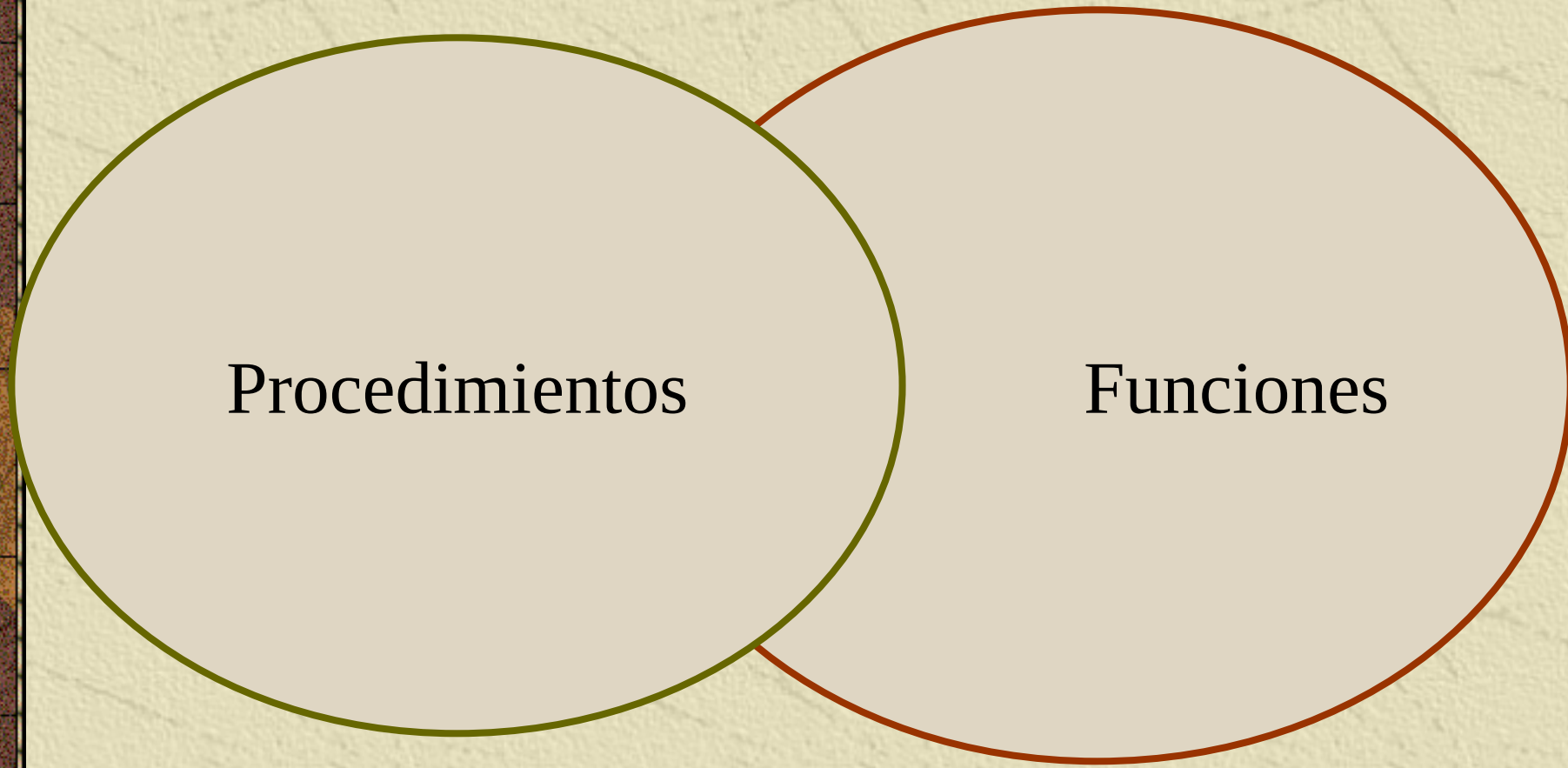


Características

- Permite resolver problemas dividiéndolo en subproblemas más sencillos de resolver mediante el ***diseño descendente***.
- Los módulos deben tener alta cohesión y mínimo acoplamiento.



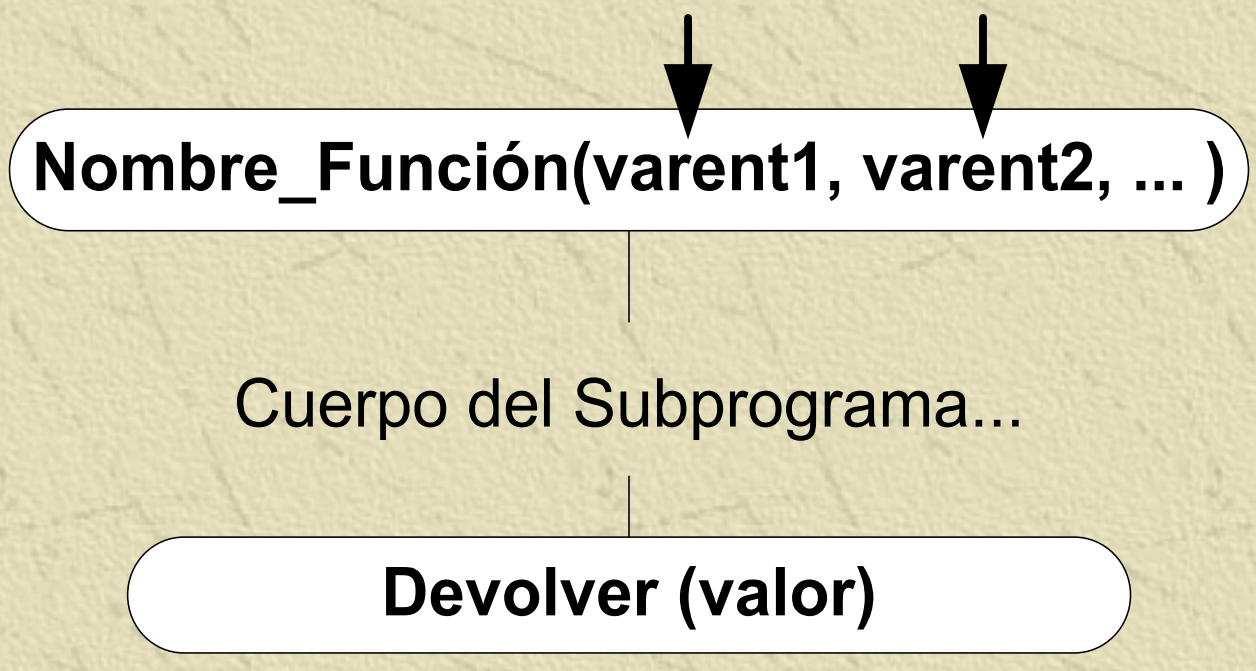
Tipos de Módulos



Funciones

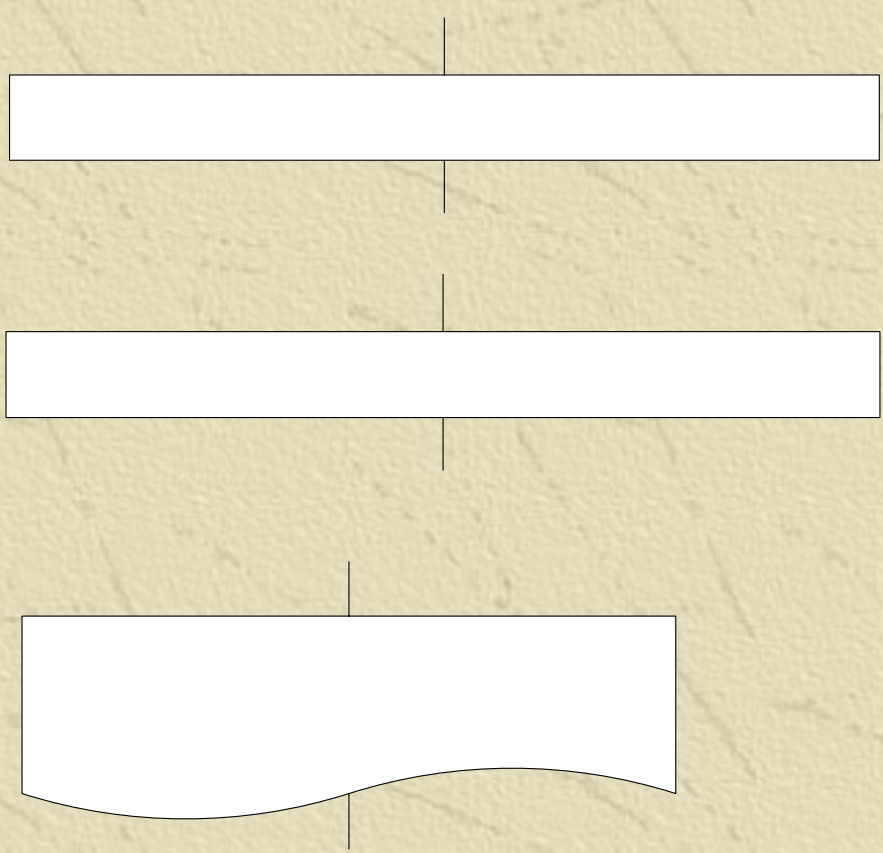
- Devuelve siempre un solo valor.
- Paso de parámetros por valor

En diagrama de flujo



Invocando una función

- Tome en cuenta que siempre devuelve un valor, por tanto:
  - Asignar el valor a una variable.
  - Realizar operaciones con el valor.
  - Imprimir el valor



Declaraciones en Lenguaje C

Tipo_dato_devuelto nombre_funcion (tipo v1, tipo vn)

{ //Cuerpo del subprograma

return val; //Devuelve valor que debe pertenecer al mismo

// tipo de dato devuelto.

}

Elaboración de un programa sin emplear la programación modular

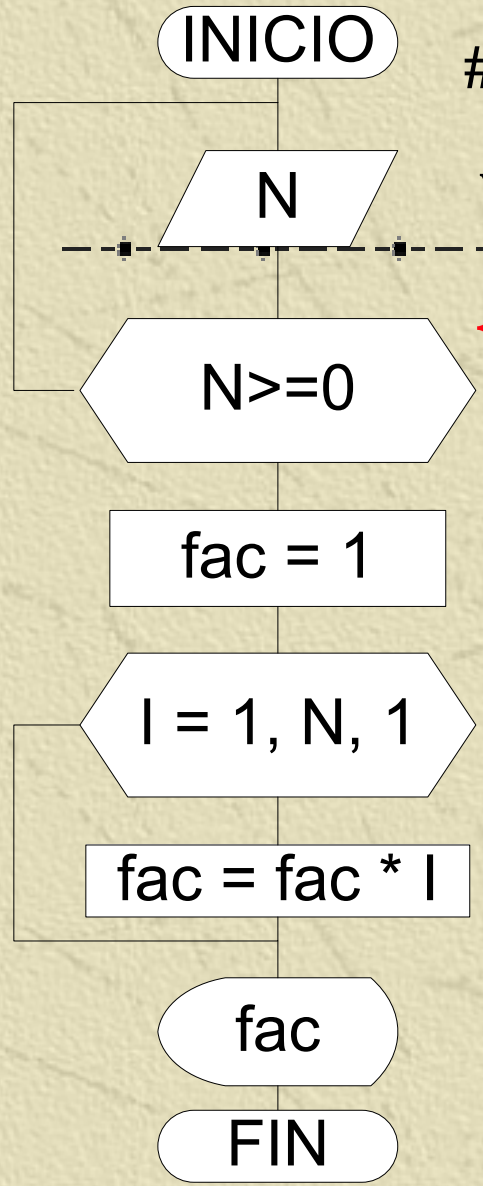
Diseñar un programa para encontrar el factorial de un número entero positivo.

Datos de Entrada:

N, número entero positivo

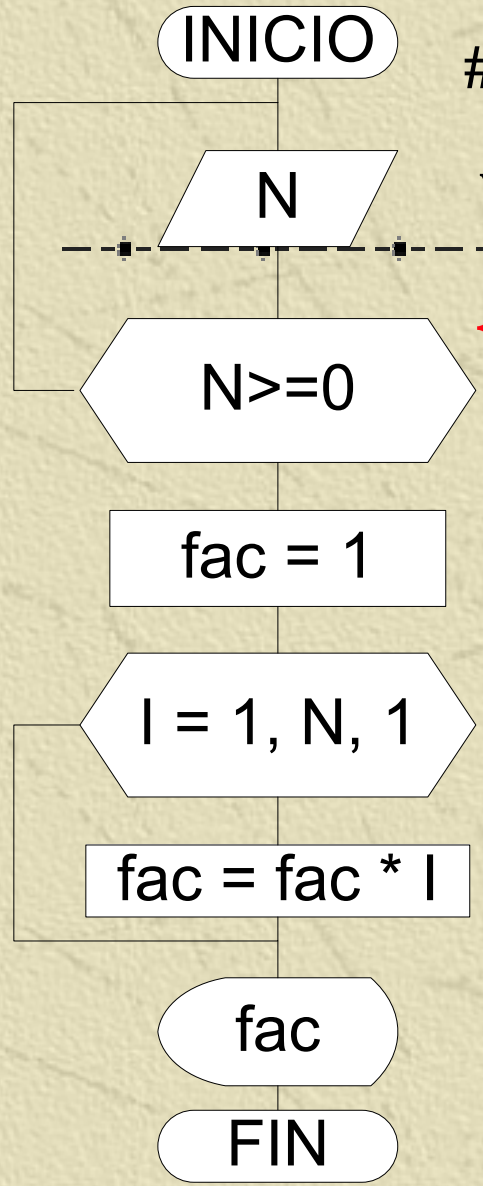
Salida:

El **factorial** de N




```
#include “stdio.h”
#include “conio.h”
void main()
{   int   n,  fac, i;    clrscr();
do{
 printf(“Digite un número entero positivo”);
 scanf(“%d”,&n);
} while (n <0);
  fac = 1;
 for(i=1; i<=n;i++)
 {     fac =  fac * i;  }
 printf(“El factorial  es: %d”, fac); getch();
}
```

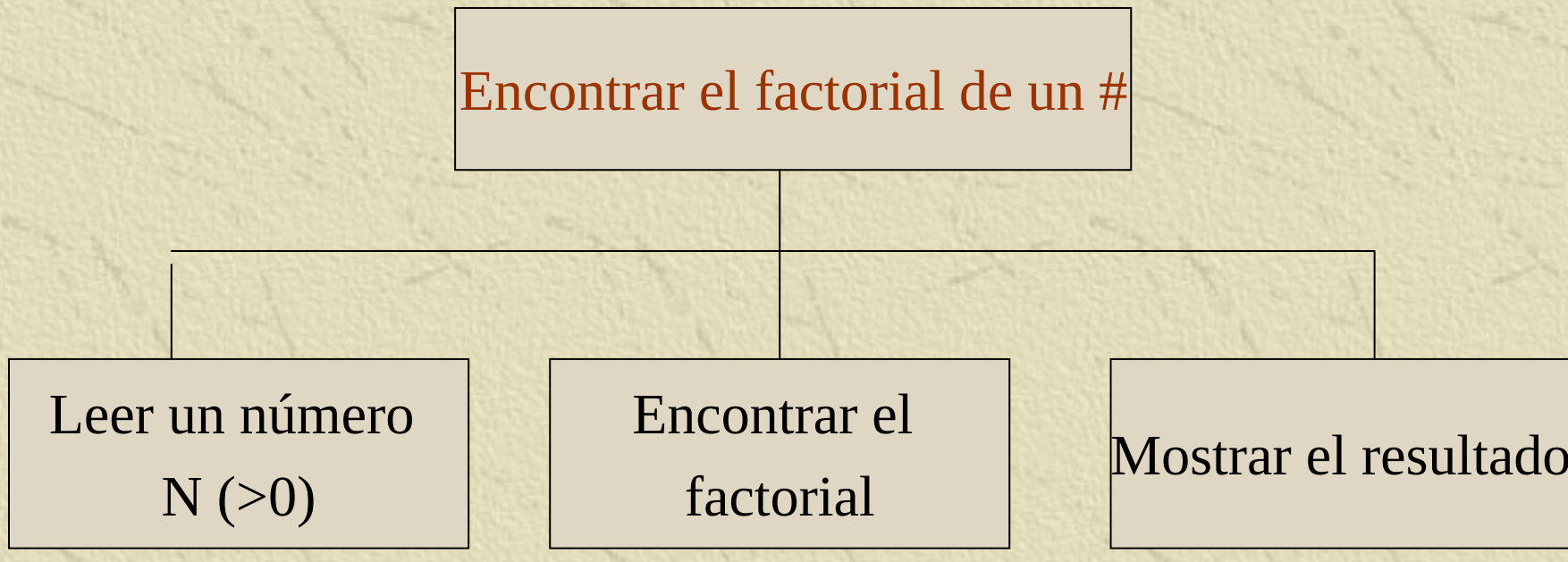
Elaborado por Lic. Gladys Chuquimia



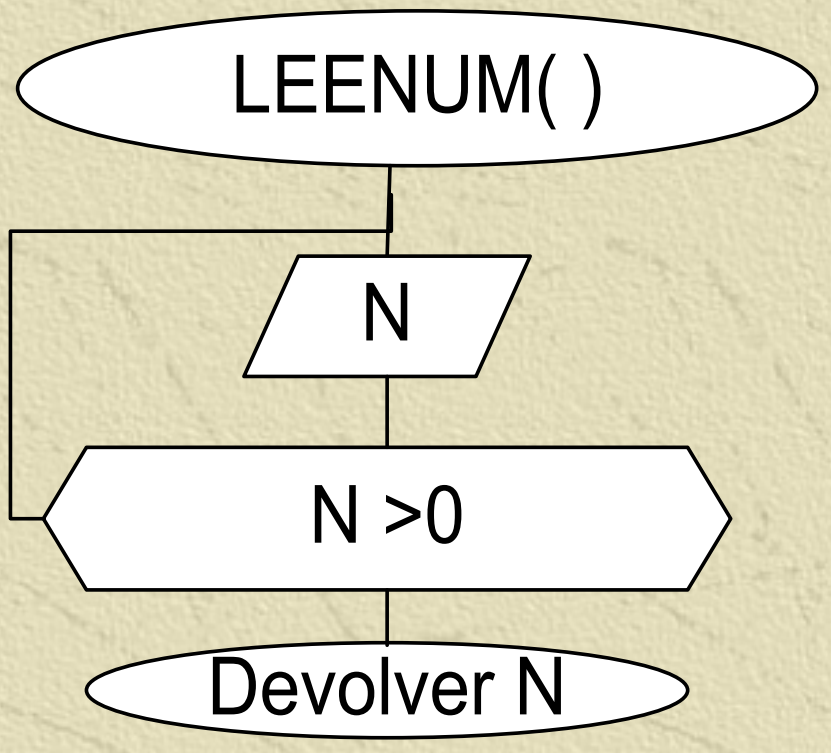

```
#include "iostream.h"
#include "conio.h"
void main()
{   int   n,  fac, i;    clrscr();
do{
cout <<"Digite un número entero positivo";
 cin >> n;
} while (n <0);
   fac = 1;
  for(i=1; i<=n; i++)
  {     fac =  fac * i;  }
 cout <<"El factorial  es:" << fac; getch();
}
```

Elaborado por Lic. Gladys Chuquimia

ANALISIS: Trabajando con Procesos bien definidos



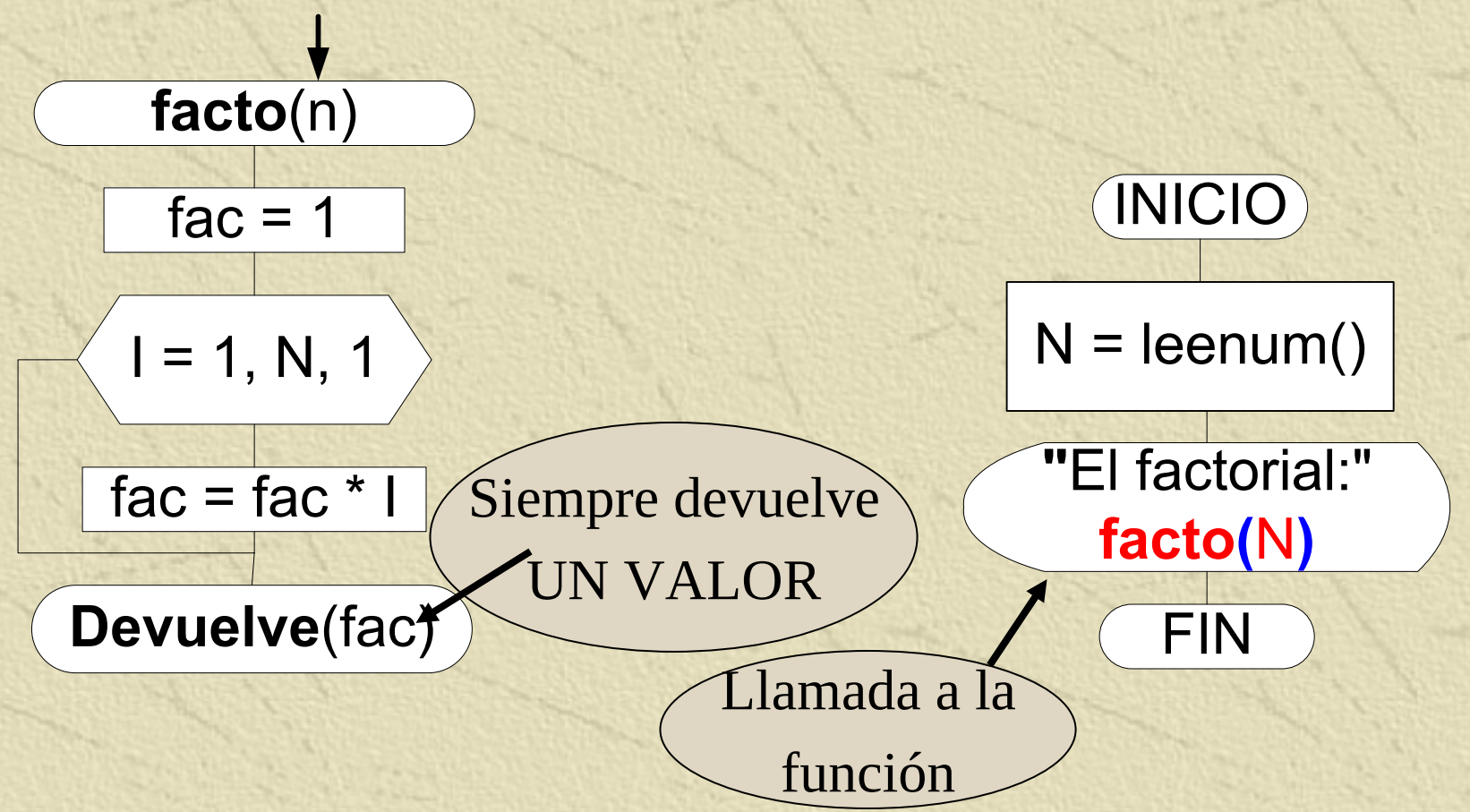
Codificando en Lenguaje C



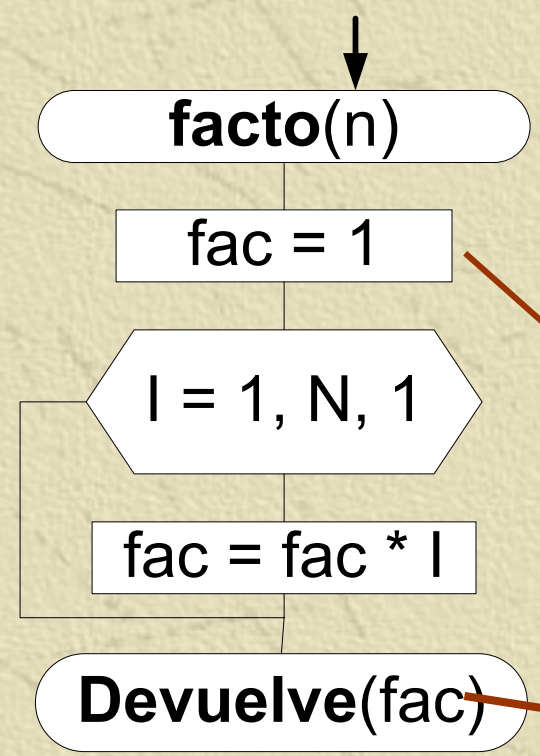

```
int leenum ( )
{
   int n;
   do{
        cin >> n;
   } while (n <= 0 );

   return n;
}
```

Diseñando la función factorial

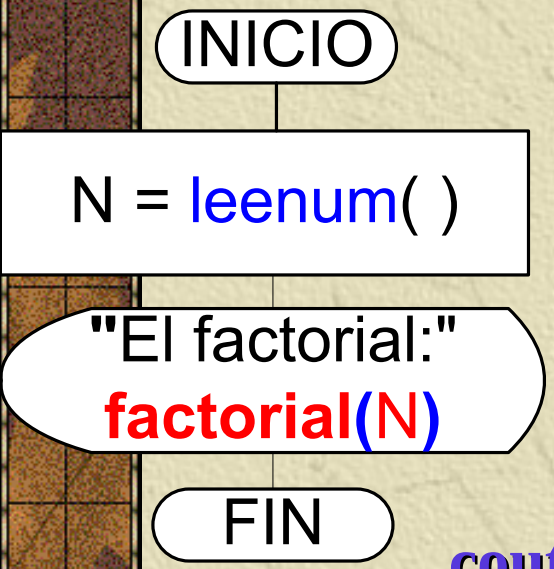


Codificando en Lenguaje C




```
unsigned long int facto (int  n)
{  unsigned long int fac;
   int i;
   fac = 1;
   for(i = 1; i <= n ;i++ )
        fac = fac * i;
   return fac;
}
```

Llamando al programa principal en lenguaje C




```
void main ( )
{  int N;
    cout <<“Digite un número entero positivo”;
   N = leenum ( );  //Llamada a la función leenum
cout << “El factorial de: ”<<N<<“ es:”<<factorial(N);
}
```

Inclusión de cabeceras y Prototipos

```
#include <iostream.h>
#include <conio.h>
//Prototipos
int leenum ( );
unsigned long int facto (int  n);
//Cuerpo de los módulos
int leenum ( )
{....
 }  //Todos los cuerpos de módulos
void main ( )  //Todo el programa principal
```

El programa: facto.cpp

```
#include <iostream.h>
#include <conio.h>
//Prototipos
int leenum ( );
unsigned long int facto (int  n);
//Cuerpo de los módulos
int leenum ( )
{  int n;
   do{
        cin >> n;
```

```
} while (n <= 0 );
   return n;
}
unsigned long int facto (int  n)
{  unsigned long int fac;
    int i;
   fac = 1;
   for(i = 1; i <= n ;i++ )
         fac = fac * i;
   return fac;
}
void main ( )
{  int N;
    cout <<"Digite un número entero positivo";
   N = leenum ( );
```

```
cout << "El factorial de: "<<N<<" es:"<<factorial(N);
getch();
}
```

Ejercicio

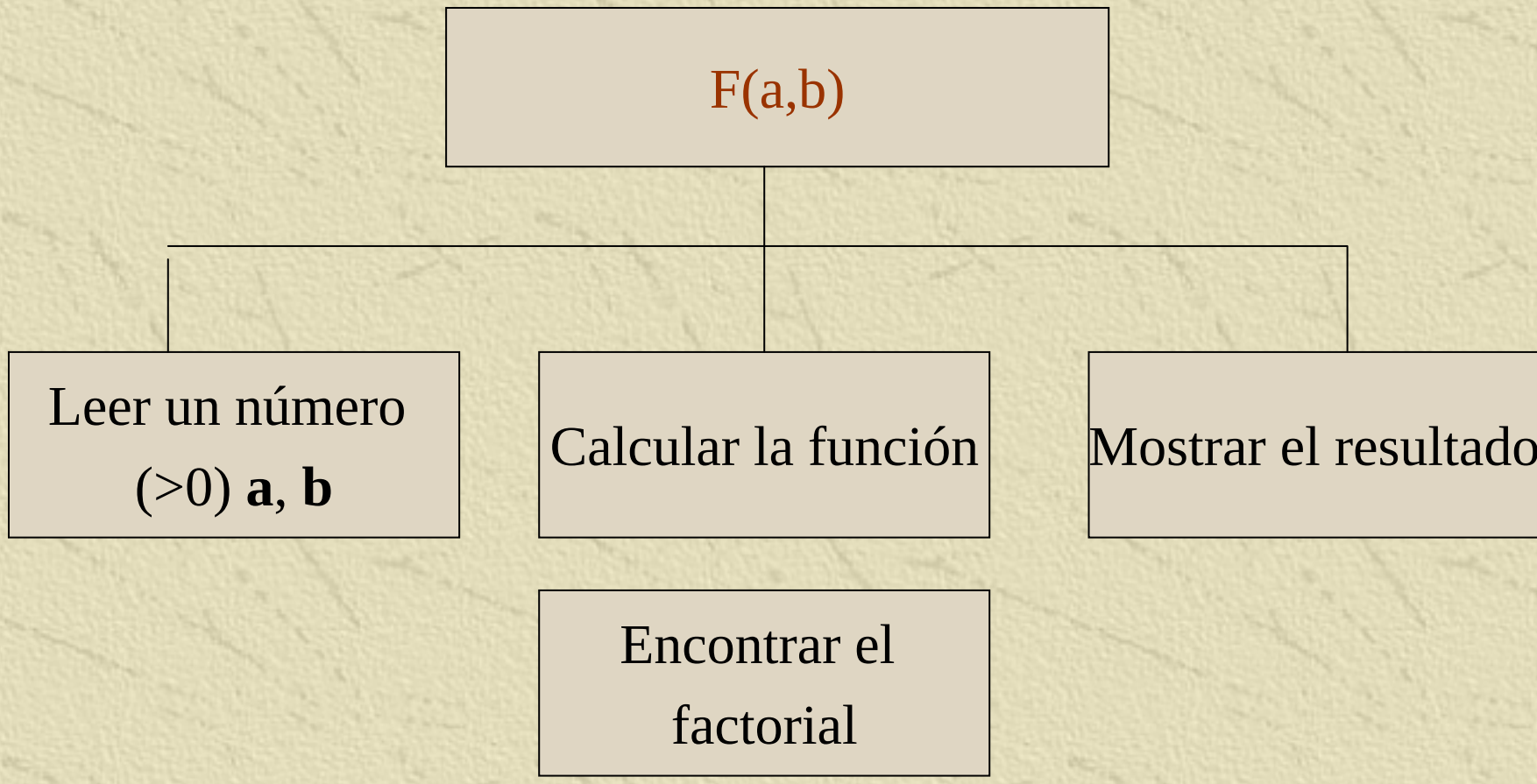
- Evaluar la siguiente función:

$$F(a, b) = a! + b!$$

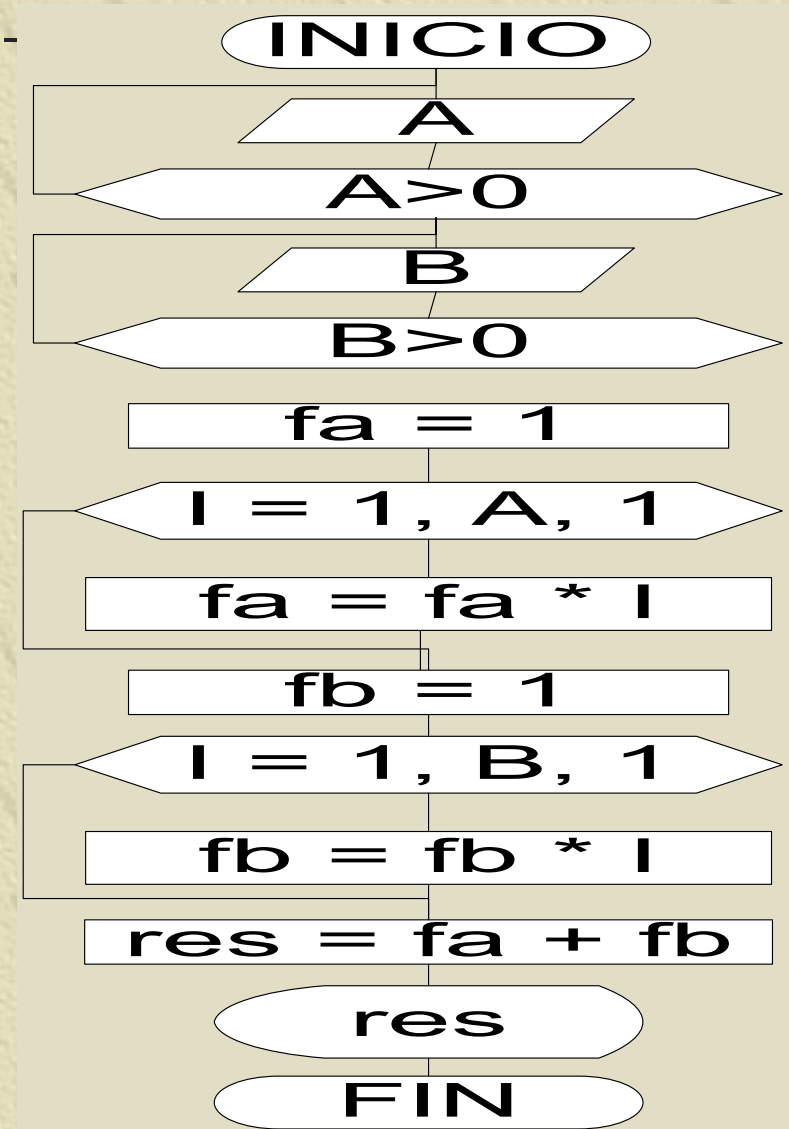
Ejemplo:

Si a = 3 y b = 5 → Salida es 126 (3! + 5!)

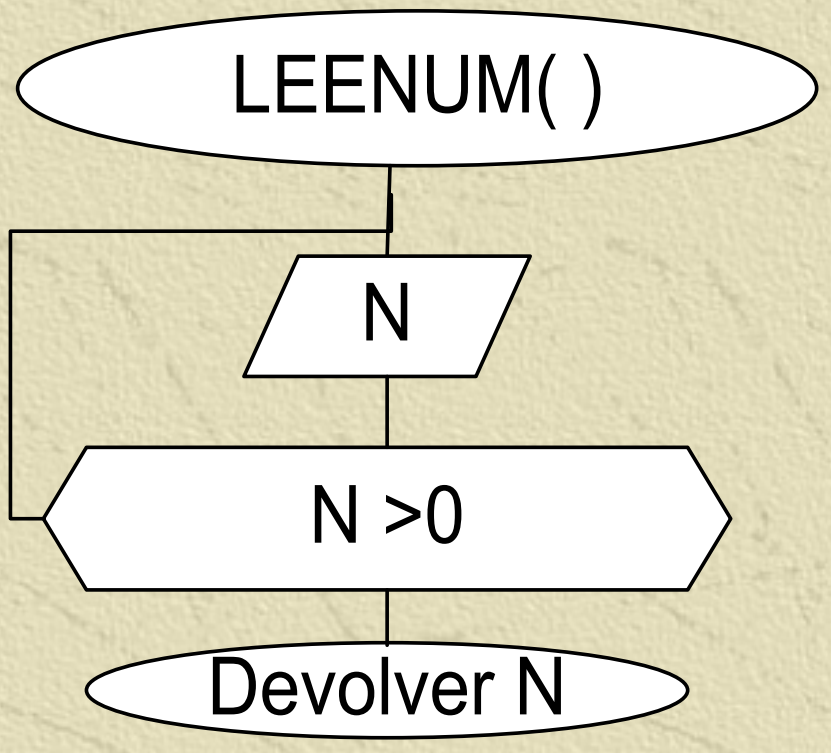
ANALISIS: Trabajando con Procesos bien definidos



De manera tradicional sería:



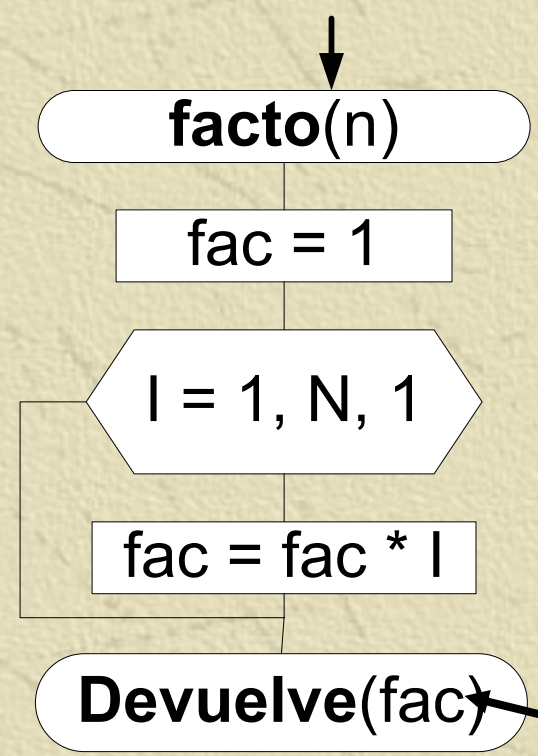
Con módulos todo se simplifica:




```
int leenum ( )
{
   int n;
   do{
        cin >> n;
   } while (n <= 0 );

   return n;
}
```

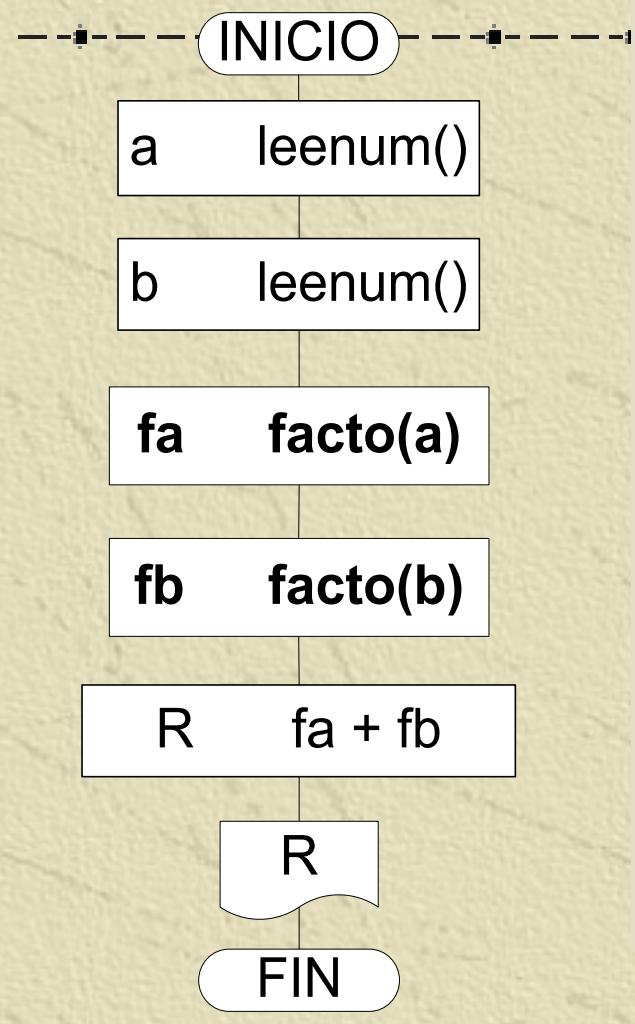
Contamos con la función factorial



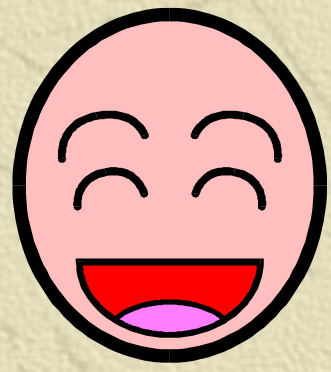

```
unsigned long int facto (int  n)
{  unsigned long int fac;
    int i;
   fac = 1;
   for(i = 1; i <= n ;i++ )
        fac = fac * i;
   return fac;
}
```

Siempre devuelve UN VALOR

El programa principal, es:




```
void main ( )
{  int a, b;
   unsigned long int fa, fb, r;
    cout <<"A : ";    a = leenum ( );
    cout <<"B : ";    b = leenum ( );
    fa = facto (a );
    fb = facto (b );
    r = fa + fb ;
    cout <<"Resultado: "<<r;
  getch();
}
```



Gracias por su atención.

Construye tus subprogramas!

Elaborado por Lic. Gladys Chuquimia