

PROGRAMACIÓN MODULAR

Lic. Gladys Chuquimia

Funciones



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

En diagrama de flujo



```
graph TD; A[Nombre_Función(varent1, varent2, ...)] --> B[Cuerpo del Subprograma...]; B --> C[Devolver (valor)];
```

Nombre_Función(varent1, varent2, ...)

Cuerpo del Subprograma...

Devolver (valor)

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

A diagram illustrating a function call. A vertical line descends from the text 'Tome en cuenta' to a horizontal rectangular box. This box is connected by a vertical line to another horizontal rectangular box below it. This second box is then connected by a vertical line to a third horizontal rectangular box with a wavy bottom edge, representing the return value of the function.

Declaraciones en Lenguaje C

↓
nomfun(p)

Devuelve(val)

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.

}

Procedimientos

- ✱ Devuelven 0 o más valores.
- ✱ Utilizan el paso de parámetros por valor y por referencia.

En lenguaje C se los declara así:



nomproc(pe, ps)

void **nomproc** (**tipo** pentrada, **tipo** *psalida)

{ // Cuerpo del subprograma

//En todo el programa si hacemos algun cambio a *psalida

//siempre se debe colocar * antes del parámetro de salida.

}

Llamada a los procedimientos:

En el diagrama de flujo:

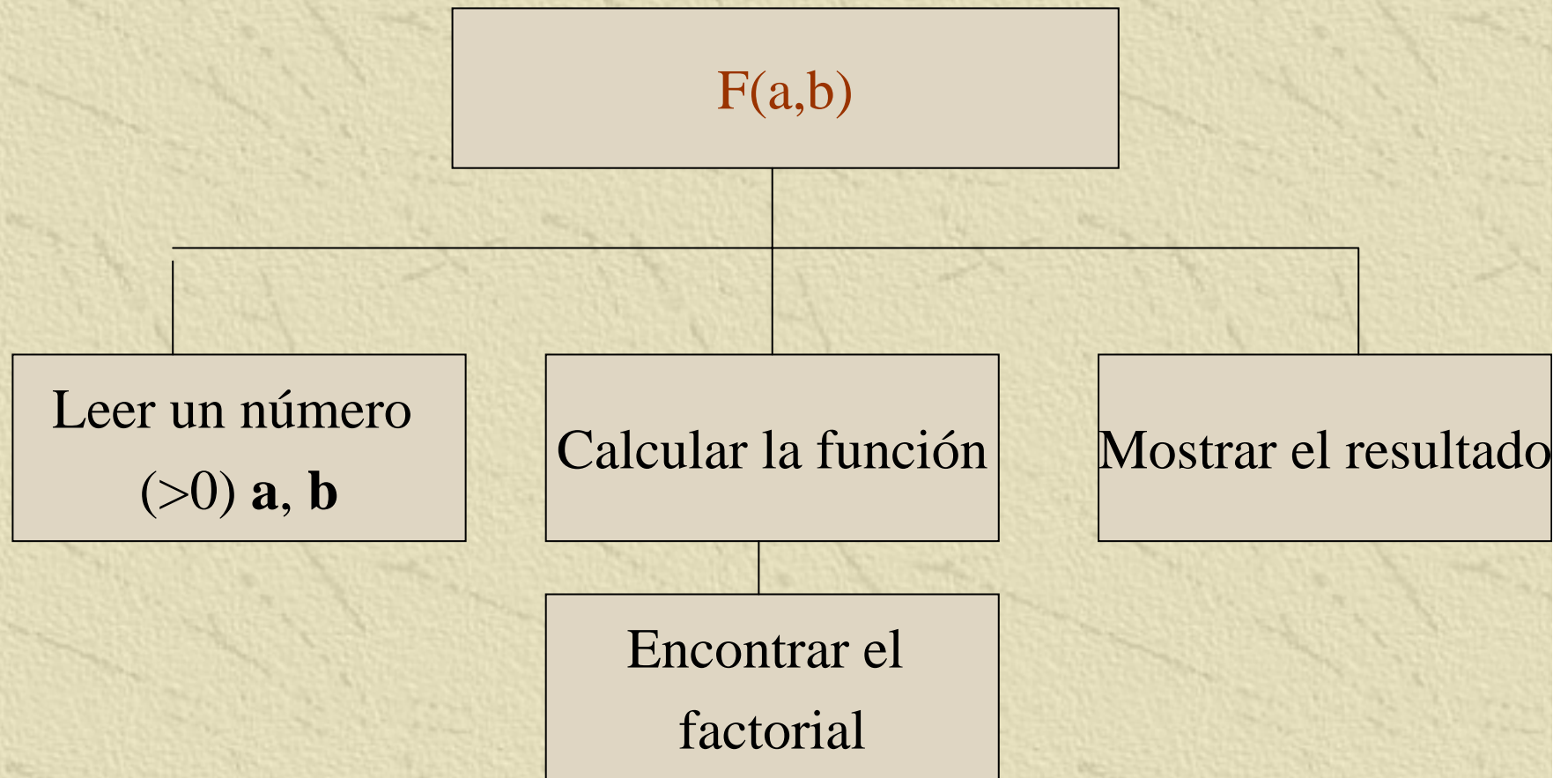


nombre_del_procedimiento (pe1, pen, ps1, psn)

En lenguaje C:

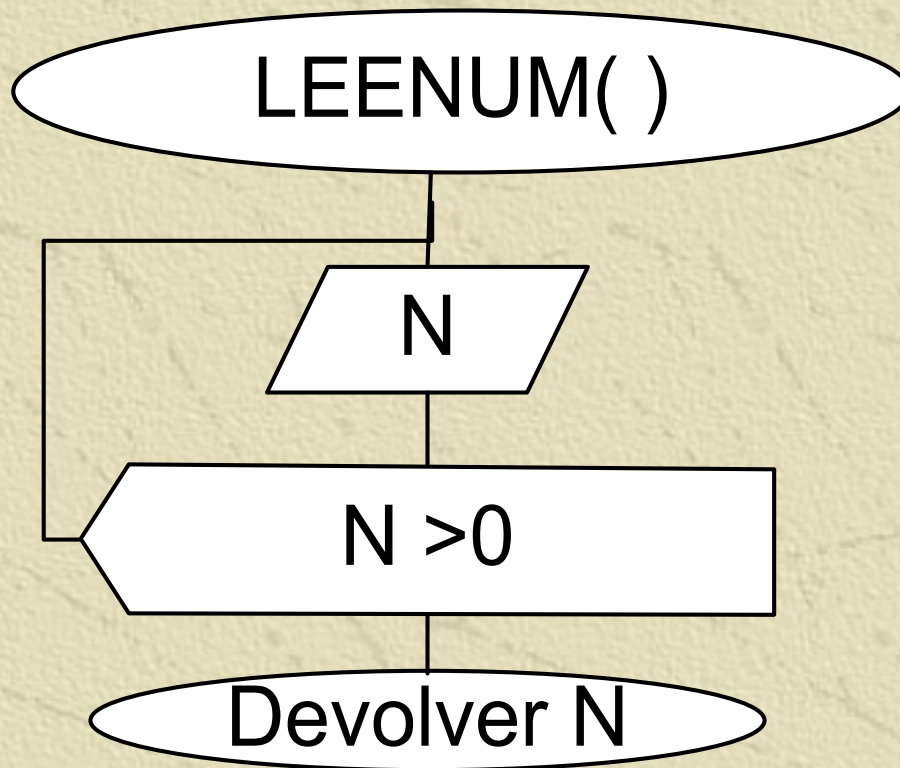
nombre_del_proc (varent1, varentN, &varsal1, & varsalN);

ANALISIS: Trabajando con Procesos bien definidos - $F(a, b) = a! + b!$



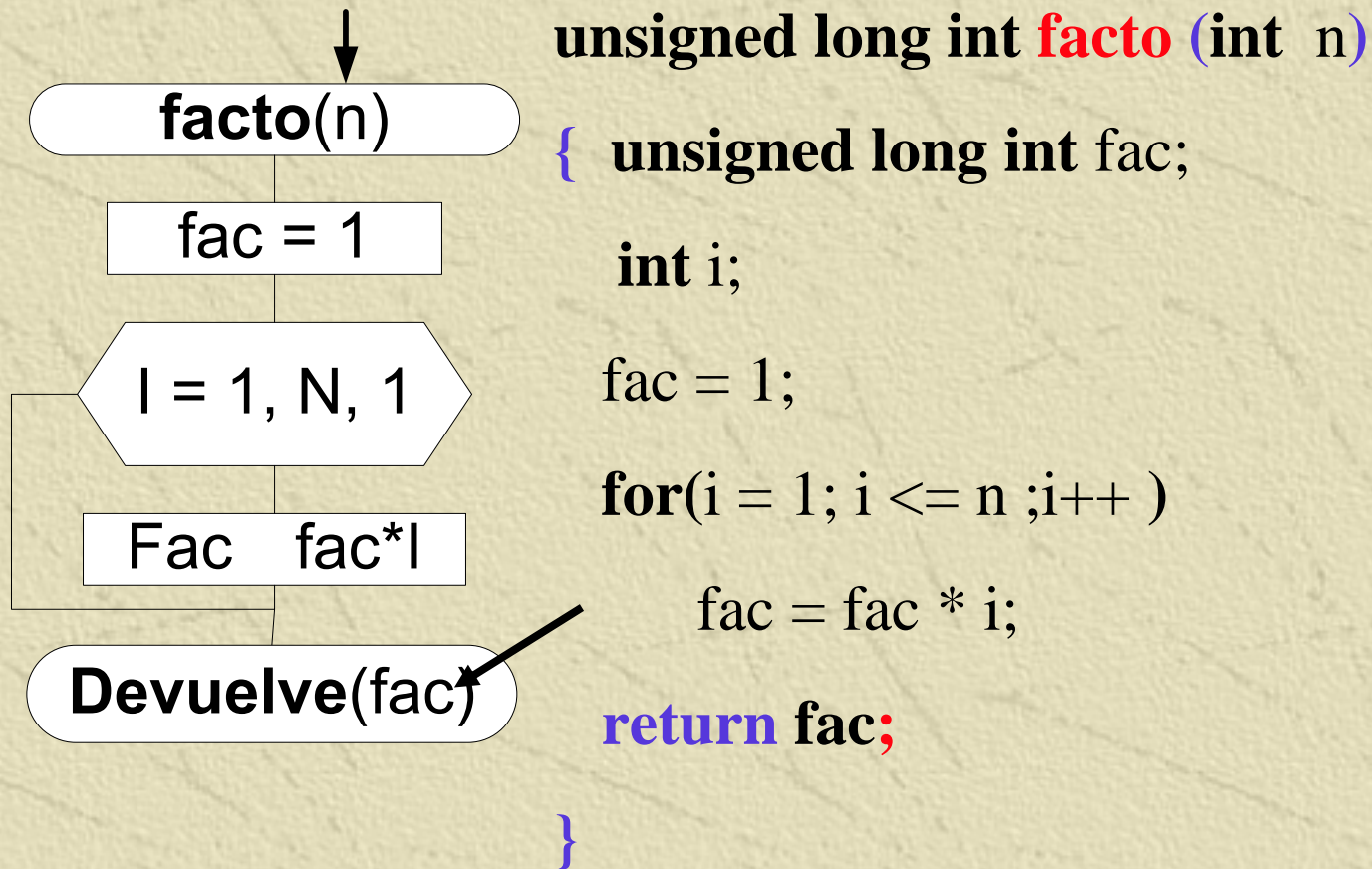
Elaborado por Lic. Gladys
Chuquimia

Codificando en Lenguaje C



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


Codificando en Lenguaje C



unsigned long int **sumfac** (int a, int b)

{ unsigned long int r;

r = **facto** (a) + **facto** (b)

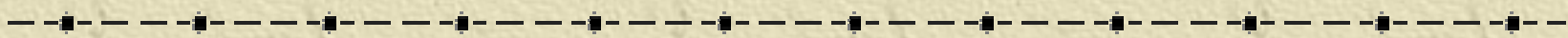
return r;

}

↓ ↓
sumfac(a, b)

R facto(a)+facto(b)

Devuelve(R)



Mostrar(r)

r

Volver

```
void mostrar (unsigned long int r)
{
    cout<<"\nResultado " <<r;
}
```

Llamando al programa principal en lenguaje C

INICIO

A leenum()

B leenum()

R sumfac(A,B)

Mostrar(R)

FIN

```
void main ( )
```

```
{ int a, b;
```

```
    unsigned long int r ;
```

```
    cout << "A = ";    a = leenum ( );
```

```
    cout << "B = ";    b = leenum ( );
```

```
    r = sumfac( a, b);
```

```
    mostrar(r);
```

```
}
```

Elaborado por Lic. Gladys
Chuquimia

Inclusión de cabeceras y Prototipos

```
#include <iostream.h>
```

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

```
//Prototipos
```

```
int leenum ( );
```

```
unsigned long int facto (int n);
```


```
unsigned long int sumfac(int a, int b);
```

```
void mostrar (unsigned long int r);
```

```
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;
```

```
}
```

```
unsigned long int sumfac (int a, int b)
```

```
{ unsigned long int r;
```

```
    r = facto (a) + facto (b)
```

```
    return r;
```

```
}
```



```
void mostrar (unsigned long int r)
{ cout<<"\nResultado " <<r;
}
```

```
void main ( )
{ int a, b;
  unsigned long int r ;
  cout <<"A = ";   a = leenum ( );
  cout <<"B = ";   b = leenum ( );
  r = sumfac( a, b);
  mostrar(r);
}
```

CTRL + F9

Paso de parámetros por Valor

ANTES DE LA LLAMADA

A = 2		

PARÁMETRO

void Subrutina (int **A**)

{ **A** = 0; }

EN LA LLAMADA

A = 2			A = 2		
		A = 2			A = 0

void main()

{ int A = 2;

Subrutina(A);

cout<<A;

}

DESPUÉS DE LA LLAMADA

A = 2		

Paso de parámetros por referencia

ANTES DE LA LLAMADA

DS:13B1

A = 2		

(TIPO_DE_DATO *IDENTIFICADOR,

PUNTEROS

PARÁMETRO

EN LA LLAMADA

A = 0		

DESPUÉS DE LA LLAMADA

A = 0		

```
void Subrutina (int *A)
```

```
{ *A= 0;}
```

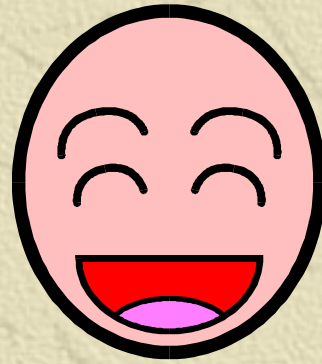
```
void main()
```

```
{ int A = 2;
```

```
  Subrutina(&A);
```

```
  cout<<A;
```

```
}
```



Gracias por su atención.

Construye tus subprogramas!