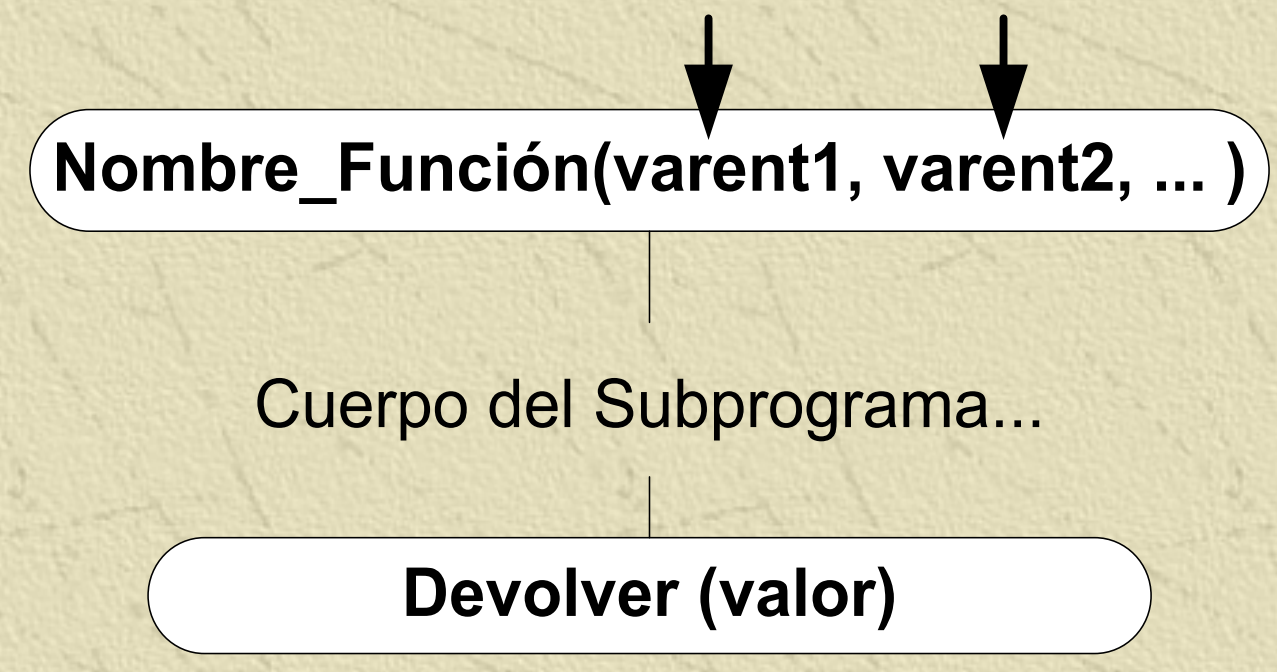
PROGRAMACIÓN MODULAR

Lic. Gladys Chuquimia

Funciones

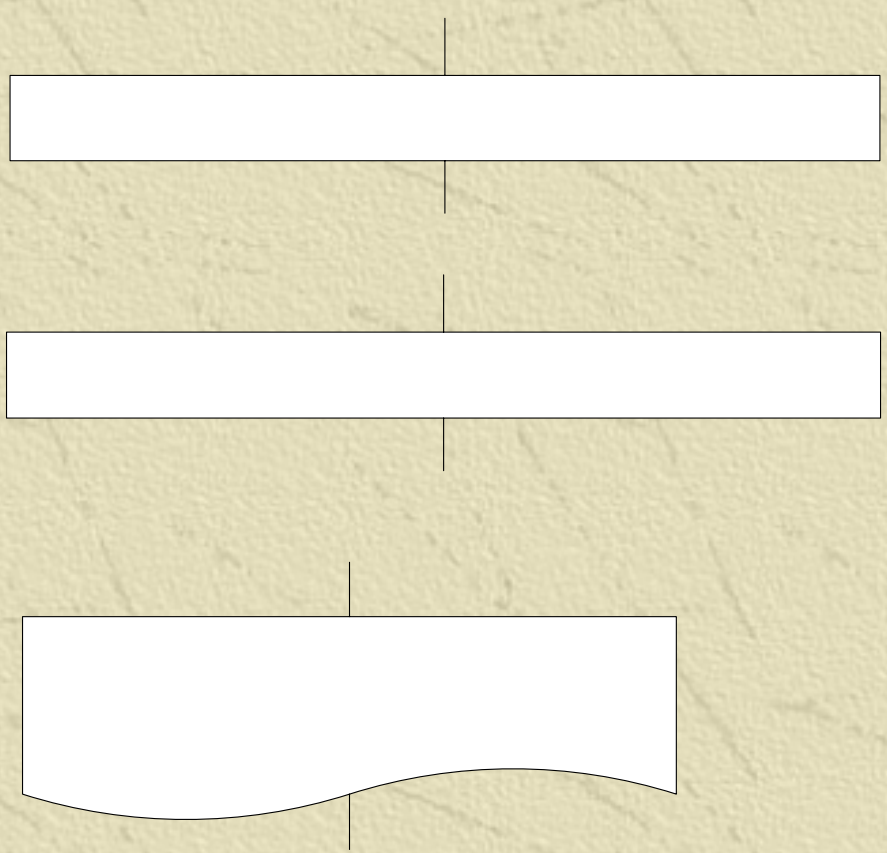
- Paso de parámetros por valor
- Devuelve siempre un solo valor.

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



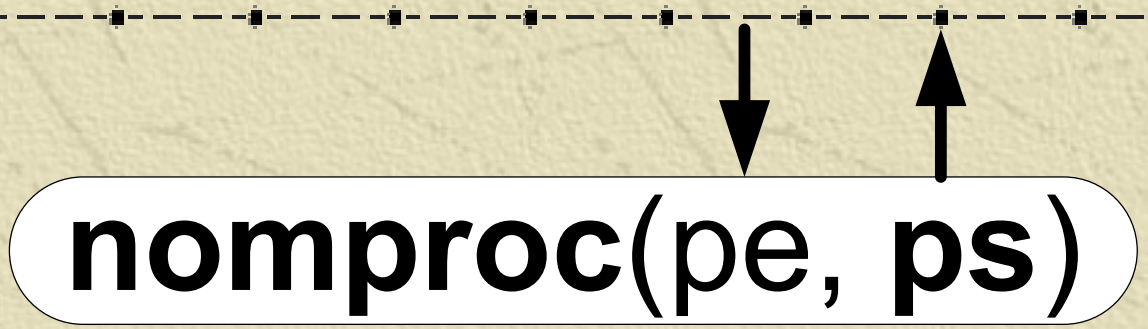
Declaraciones en Lenguaje C


```
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.
}
```

Procedimientos

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

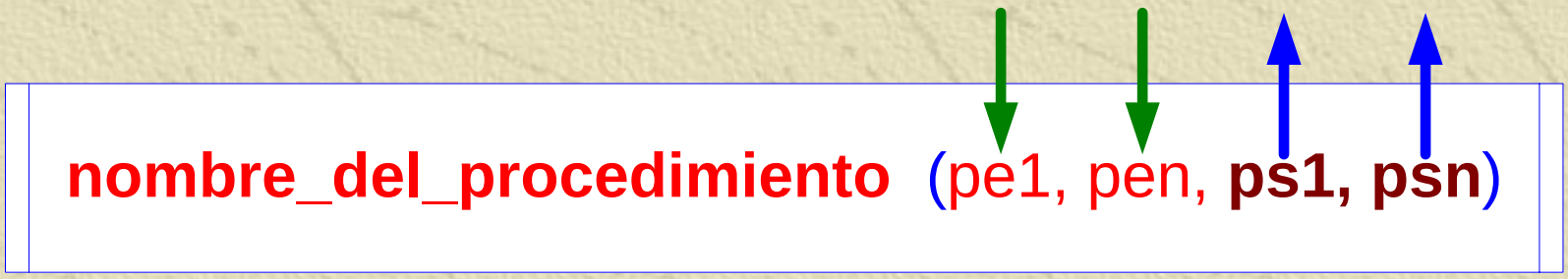
En lenguaje C se los declara así:




```
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:

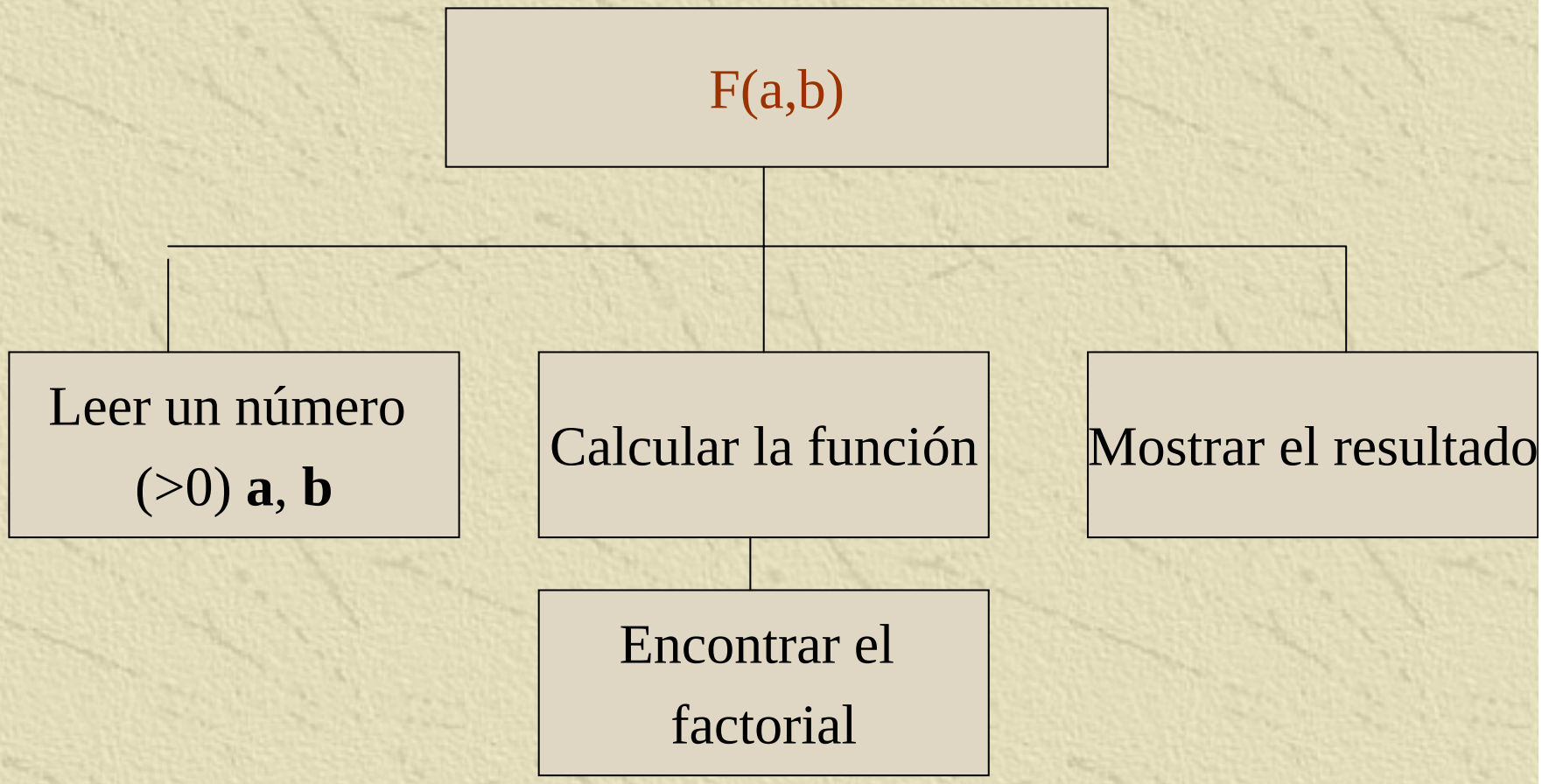
En el diagrama de flujo:



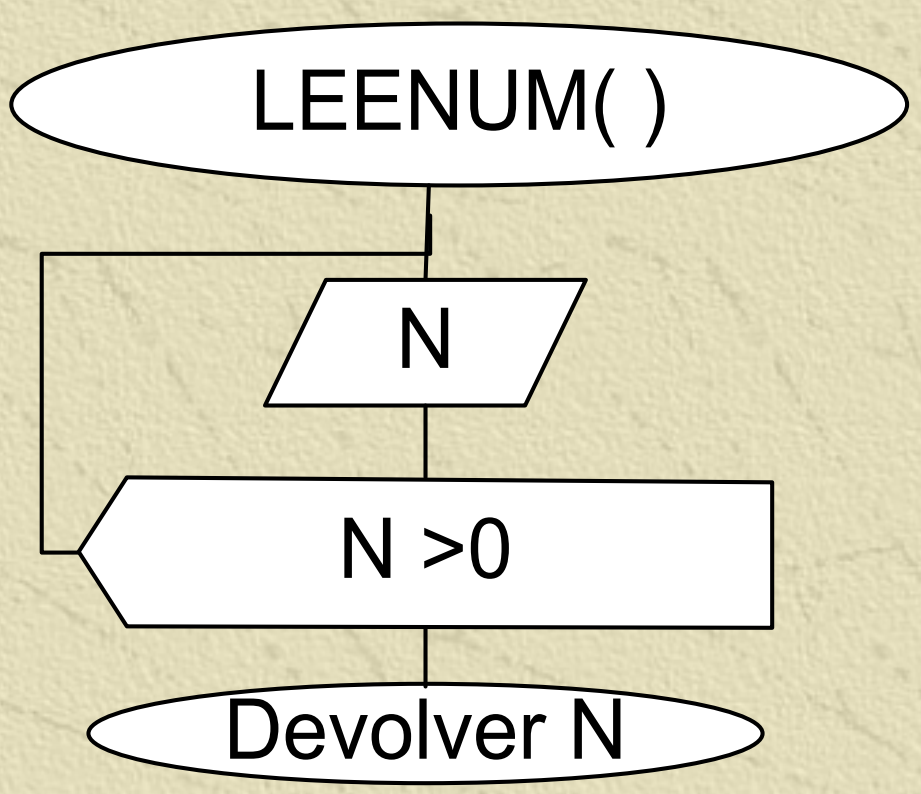
En lenguaje C:

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

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

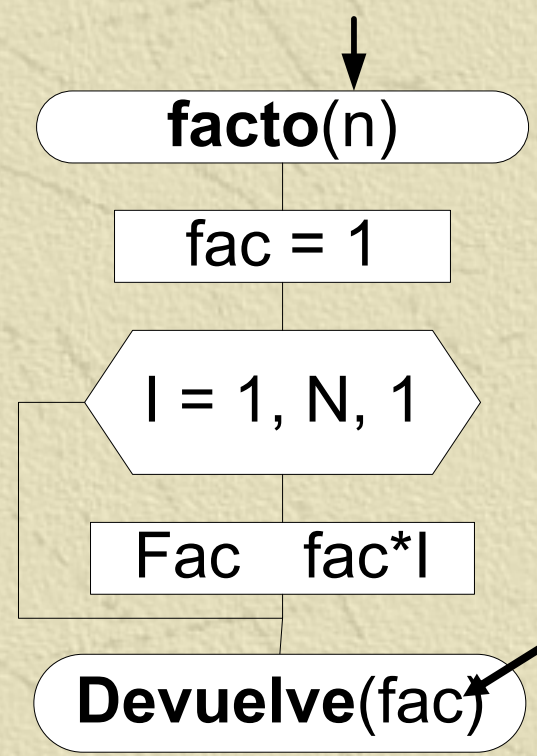


Codificando en Lenguaje C

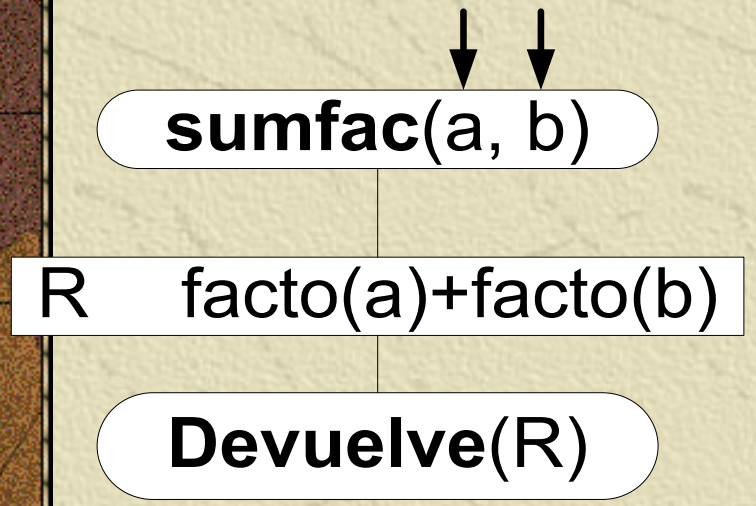



```
int leenum ( )
{
   int n;
   do{
        cin >> n;
   } while (n <= 0 );

   return n;
}
```

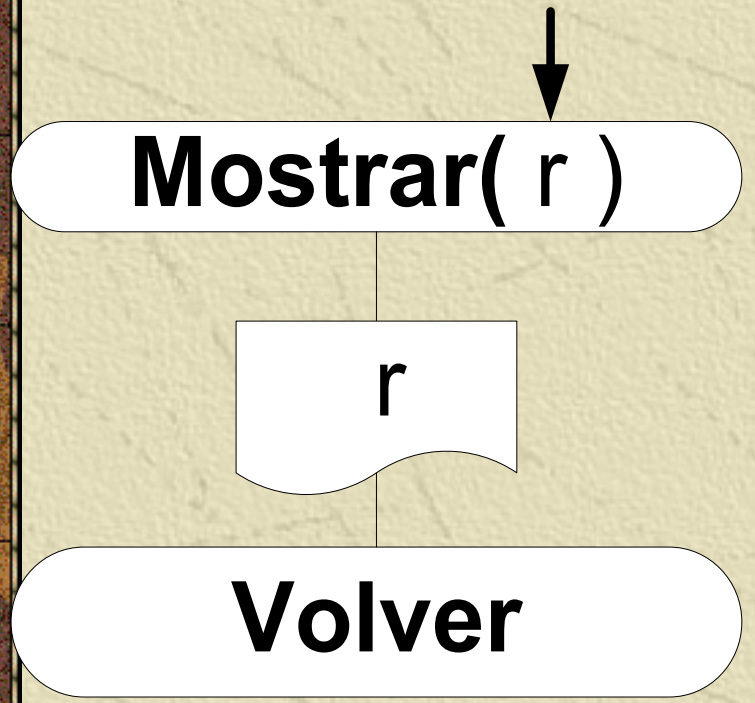
Codificando en Lenguaje C




```
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;
}
```

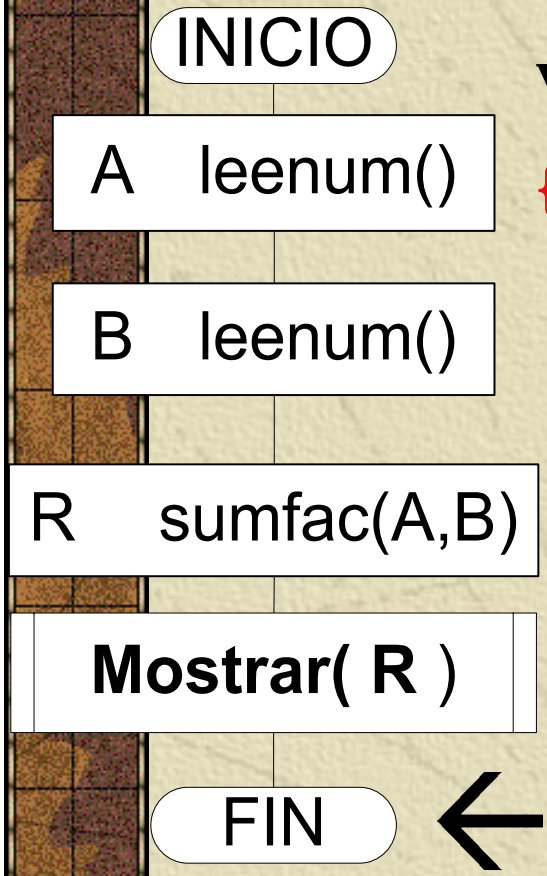



```
unsigned long int sumfac (int  a, int  b)
{  unsigned long int r;
   r = facto (a) + facto (b)
   return r;
}
```




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

Llamando al programa principal en lenguaje C




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

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;
 }
```

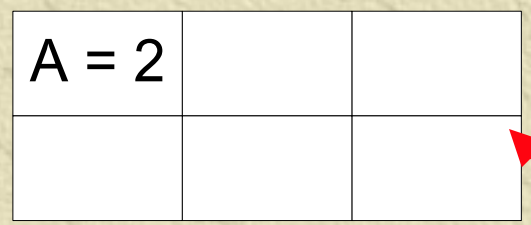
```
void mostrar (unsigned long int r)
{  cout<<"\nResultado " <<r;
}
```

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

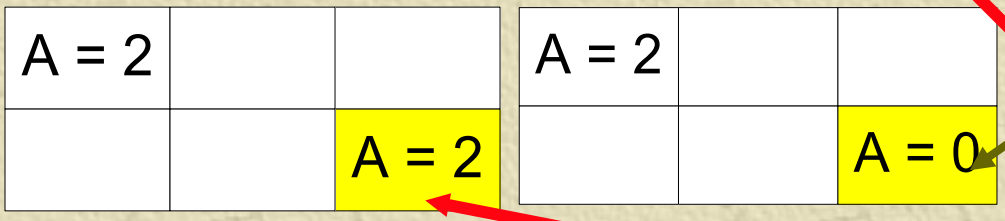
CTRL + F9

Paso de parámetros por Valor

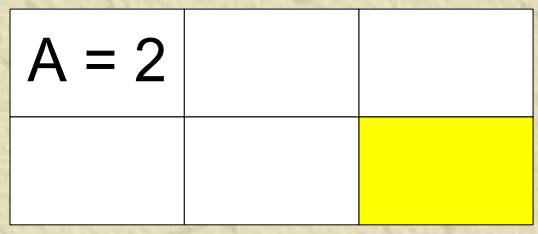
ANTES DE LA LLAMADA



EN LA LLAMADA



DESPUÉS DE LA LLAMADA

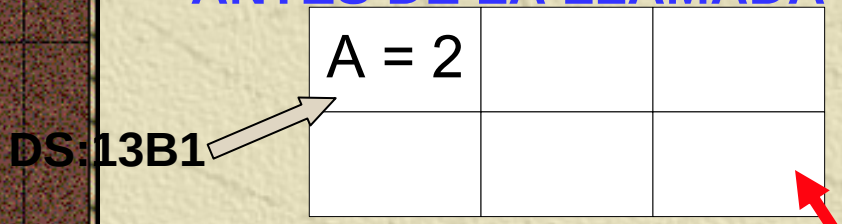


PARÁMETRO

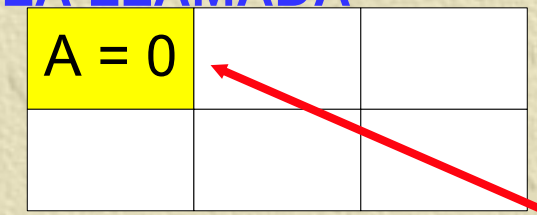
```
void Subrutina (int  A)
{  A= 0;}

void main( )
{  int A = 2;
   Subrutina(A);
   cout<<A;
}
```

Paso de parámetros por referencia

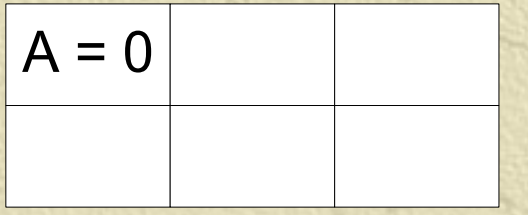
ANTES DE LA LLAMADA



EN LA LLAMADA



DESPUÉS DE LA LLAMADA

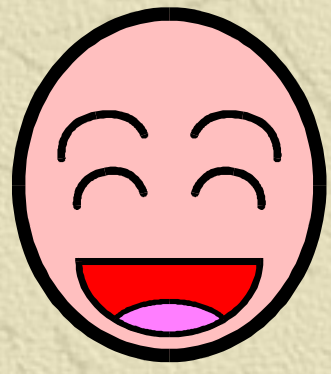


(TIPO_DE_DATO *IDENTIFICADOR,

PUNTEROS

PARÁMETRO

```
void Subrutina (int   *A)
{ *A= 0;}
void main()
{  int A = 2;
   Subrutina(&A);
   cout<<A;
}
```



Gracias por su atención.

Construye tus subprogramas!