



# TEMA 6

## Estructuras de control

Grado en Ingeniería Eléctrica

Grado en Ingeniería Electrónica Industrial y Automática

**Escuela Politécnica de Ingeniería de Ferrol**

<https://www.udc.es/epef>

6.1.- Expresiones lógicas

6.2.- Instrucciones selectivas

6.3.- Instrucciones iterativas

6.4.- Instrucciones de salto

6.5.- Anexo: ejercicios resueltos

## 6.2.- Expresiones lógicas

- Se denomina expresión lógica a la que se evalúa como VERDADERA o FALSA
- Relaciona 2 o más expresiones mediante operadores relacionales y/o lógicos

**Operadores relaciones**

==	Igual
!=	Distinto
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

**Operadores lógicos**

&&	Y
	O
!	No

## 6.2.- Expresiones lógicas

- Operadores relacionales: comparan expresiones numéricas
- Operadores lógicos: se usan para realizar expresiones lógicas complejas a partir de otras más simples

`(a>b) && (c>b)`

`(numero != 0) || (letra == 'a')`

- En lenguaje C cualquier expresión numérica evaluada en un contexto lógico cuyo resultado sea:
  - Igual a 0                       $\rightarrow$             se considera **falsa**
  - Distinto de 0                 $\rightarrow$             se considera **verdadera**

## 6.2.- Expresiones lógicas

- Ejemplos:
  - La expresión **(a-5)** es equivalente a **(a!=5)**
  - La expresión **(opcion)** es equivalente a **(opcion!=0)**
- Por claridad es preferible usar los operadores relacionales y lógicos
- **Importante:** No confundir el operador de asignación = con el de comparación ==
  - Ejemplos:

<b>(a=0)</b>	→	Incorrecto
<b>(a==0)</b>	→	Correcto

## 6.3.- Instrucciones selectivas o condicionales

### ■ Sentencia condicional simple: **if**

- Sintaxis:

```
if (condición)  
    sentencia1;
```

```
if (condición)  
    sentencia1;  
else  
    sentencia2;
```

La parte **else** es opcional

- Ejemplos:

```
if (a!=0)  
    resultado = b/a;
```

```
if (a!=0)  
    resultado = b/a;  
else  
    printf("\nError!!");
```

## 6.3.- Instrucciones selectivas o condicionales

- Si son necesarias varias sentencias en la parte **if** o **else**:  
bloque de instrucciones o sentencias

### Sintaxis:

```
if (condición)
{
    Bloque de sentencias
}
else
{
    Bloque de sentencias
}
```

### Ejemplo:

```
if (numero>=0)
{
    contador = contador + 1;
    printf("\nNúmero correcto");
}
else
    printf("\nNúmero negativo");
```

## 6.3.- Instrucciones selectivas o condicionales

- Alcance de la sentencia if o else:
  - Si se usa un bloque de sentencias
    - Delimitado por las llaves: { }
  - Si no se usa un bloque de sentencias:
    - Terminan en el primer punto y coma
- Ejemplo:

```
if (lado!=0)
    Area = lado*lado;
    printf("\nArea: %f", Area);
```

← Fuera del alcance del if
- En el ejemplo: si se coloca un else después del printf → error



## 6.3.- Instrucciones selectivas o condicionales

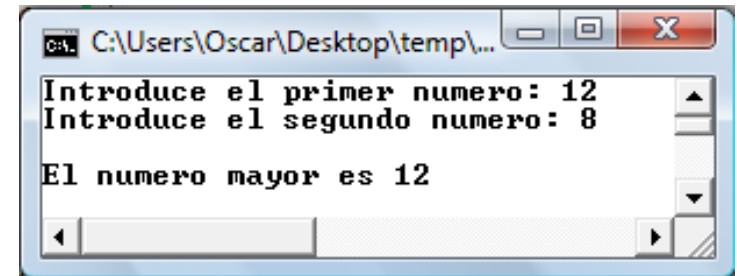
- Ejemplo: calcular el número mayor de dos dados

```
#include "stdio.h"

void main()
{
    int numero1, numero2;

    printf("Introduce el primer numero: ");
    scanf("%i", &numero1);
    printf("Introduce el segundo numero: ");
    scanf("%i", &numero2);

    if (numero2 > numero1)
        printf("\nEl numero mayor es %i\n", numero2);
    else
        printf("\nEl numero mayor es %i\n", numero1);
}
```

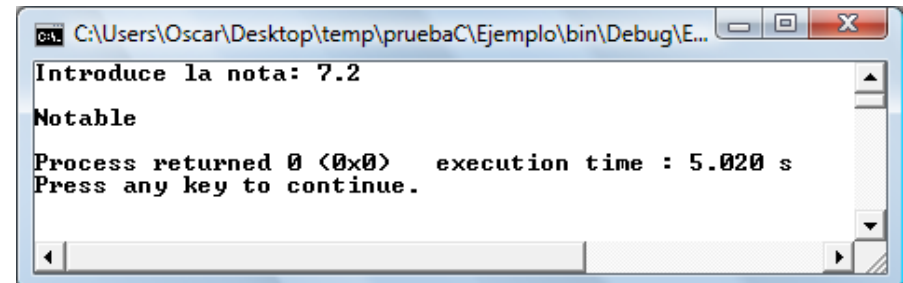


## 6.3.- Instrucciones selectivas o condicionales

- Sentencias **if** anidadas:
  - Cuando una sentencia if se ejecuta dentro de una de las ramas de otra sentencia if
  - El anidamiento puede ser en varios niveles
- Ejemplo:

```
#include "stdio.h"
void main()
{
    float nota;

    printf("Introduce la nota: ");
    scanf("%f", &nota);
    if (nota < 5)
        printf("\nSuspenso\n");
    else
        if (nota < 7)
            printf("\nAprobado\n");
        else
            if (nota < 9)
                printf("\nNotable\n");
            else
                printf("\nSobresaliente\n");
}
```



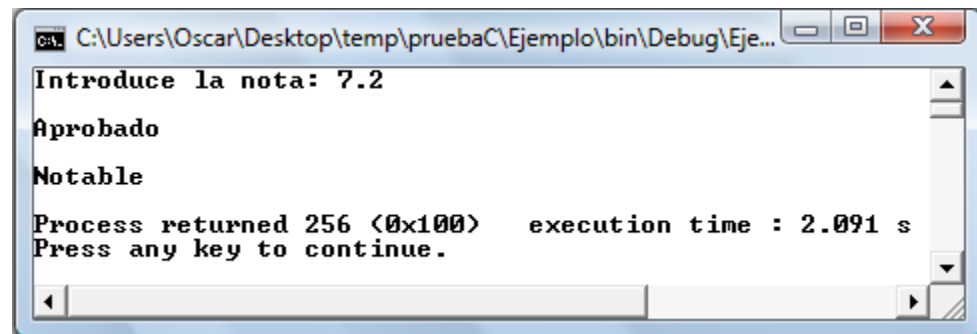
## 6.3.- Instrucciones selectivas o condicionales

- Ejemplo: ¿es válido este programa para mostrar la nota de un alumno?

```
#include "stdio.h"
void main()
{
    float nota;

    printf("Introduce la nota: ");
    scanf("%f", &nota);
    if (nota < 5)
        printf("\nSuspendido\n");
    if (nota >= 5)
        printf("\nAprobado\n");
    if (nota >= 7)
        printf("\nNotable\n");
    if (nota >= 9)
        printf("\nSobresaliente\n");
}
```

**PROGRAMA INCORRECTO**



## 6.3.- Instrucciones selectivas o condicionales

### ■ Bifurcación múltiple: sentencia **switch**

- Toma distintas acciones según el valor de una expresión de tipo *int* o *char*
- Sintaxis:

```
switch(expresión)  
{  
    case constante1: Bloque de sentencias1;  
    case constante2: Bloque de sentencias2;  
    .  
    .  
    case constanteN: Bloque de sentenciasN;  
    default: Bloque de sentencias_default;  
}
```

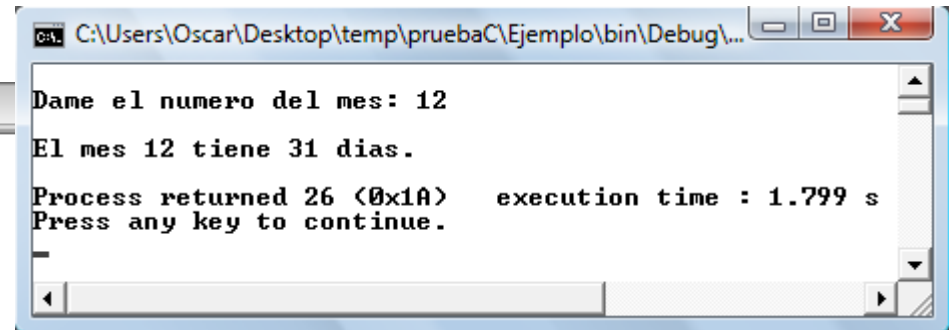
## 6.3.- Instrucciones selectivas o condicionales

- Se compara el valor con cada constante (por orden) y si coincide se ejecutan las sentencias asociadas
- **Importante:** Una vez que se cumple una sentencia *case* se ejecutan también las sentencias de todos los *case* que hay debajo
- Para cortar la ejecución en un *case* → ***break***
- La etiqueta ***default*** es opcional

## 6.3.- Instrucciones selectivas o condicionales

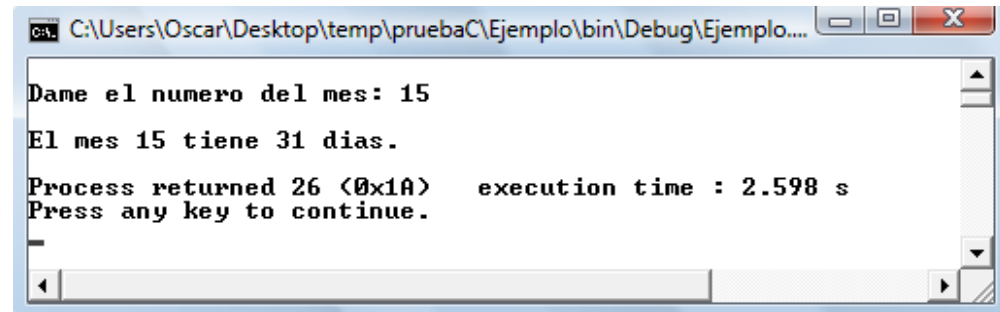
### ■ Ejemplo:

```
main.c x
1  #include "stdio.h"
2
3  void main()
4  {
5  ▶  int mes, dias;
6
7  printf("\nDame el numero del mes: ");
8  scanf("%d", &mes);
9  switch(mes)
10 {
11     case 2    : dias = 28;
12                break;
13     case 4    :
14     case 6    :
15     case 9    :
16     case 11   : dias = 30;
17                break;
18     default   : dias = 31;
19 }
20 printf("\nEl mes %d tiene %d dias.\n", mes, dias);
21 }
```



C:\Users\Oscar\Desktop\temp\pruebaC\Ejemplo\bin\Debug\...  
Dame el numero del mes: 12  
El mes 12 tiene 31 dias.  
Process returned 26 (0x1A) execution time : 1.799 s  
Press any key to continue.

**El resultado es incorrecto si el número de mes no es válido.  
¿Cómo se puede solucionar?**



C:\Users\Oscar\Desktop\temp\pruebaC\Ejemplo\bin\Debug\Ejemplo...  
Dame el numero del mes: 15  
El mes 15 tiene 31 dias.  
Process returned 26 (0x1A) execution time : 2.598 s  
Press any key to continue.

## 6.3.- Instrucciones selectivas o condicionales

### ■ Ejemplo:

```
main.c x
1  #include "stdio.h"
2  void main()
3  {
4      int mes, dias;
5
6      printf("\nDame el numero del mes: ");
7      scanf("%d", &mes);
8      if (mes > 0 && mes <= 12)
9      {
10         switch(mes)
11         {
12             case 2 : dias = 28;
13                     break;
14             case 4 :
15             case 6 :
16             case 9 :
17             case 11 : dias = 30;
18                       break;
19             default : dias = 31;
20         }
21         printf("\nEl mes %d tiene %d dias.\n", mes, dias);
22     }
23     else
24         printf("\nERROR, el mes numero %i no es valido.\n", mes);
25 }
```

C:\Users\Oscar\Desktop\temp\pruebaC\Ejemplo\bin\Debu...  
Dame el numero del mes: 15  
ERROR, el mes numero 15 no es valido.  
Process returned 39 (0x27) execution time : 1.521 s  
Press any key to continue.

## 6.3.- Instrucciones iterativas

- Sirven para ejecutar una o varias instrucciones múltiples veces
- También se les denomina **bucles**
- Tres tipos de bucles en lenguaje C:
  - **do while**
  - **while**
  - **for**



## 6.3.- Instrucciones iterativas

### ■ Bucle **do while**:

- Tiene la condición al final (con el while)
- Todas las instrucciones contenidas en el bucle se repiten (al menos una vez) mientras se cumpla la condición
- El bucle finaliza cuando deja de cumplirse la condición
- Sintaxis:

```
do  
    sentencia;  
while (condición);
```

```
do {  
    sentencia1;  
    ....  
    sentenciaN;  
} while (condición);
```

## 6.3.- Instrucciones iterativas

- Ejemplo:

```
main.c x
1  #include "stdio.h"
2
3  void main()
4  {
5      int numero, suma=0;
6
7      do
8      {
9          printf("\nDame un numero: ");
10         scanf("%d", &numero);
11         if (numero > 0)
12             suma = suma + numero;
13     } while(numero != 0);
14
15     printf("\nLa suma es: %d", suma);
16 }
```

```
C:\Users\Oscar\Desktop\temp\pruebaC\Ejemplo\bin\Debu...
Dame un numero: 5
Dame un numero: 12
Dame un numero: -2
Dame un numero: 8
Dame un numero: -2
Dame un numero: 0
La suma es: 25
Process returned 16 (0x10)   execution time : 14.624
Press any key to continue.
```

## 6.3.- Instrucciones iterativas

### ■ Bucle **while**:

- Tiene la condición al principio del bucle
- Las sentencias contenidas en él se ejecutan de 0 a M veces
- Sintaxis:

```
while (condición)  
    sentencia;
```

```
while (condición)  
{  
    sentencia1;  
    ....  
    sentenciaN;  
}
```

## 6.3.- Instrucciones iterativas

- Ejemplo: calcular el **primer** año bisiesto entre dos dados por teclado (un año es bisiesto si es múltiplo de 4, excepto los múltiplos de 100 que no lo son salvo que a su vez también lo sean de 400)

```
#include "stdio.h"
void main()
{
    int inicial, final, actual, aux, bisiesto=0;
```

```
    printf("Introduce el primer año: ");
    scanf("%i", &inicial);
    printf("Introduce el segundo año: ");
    scanf("%i", &final);
```

} Lectura de los dos datos por teclado

```
    if (inicial > final)
    {
        aux = inicial;
        inicial = final;
        final = aux;
```

} Los datos se intercambian si el primer dato introducido es mayor que el segundo. Para ello es necesaria una variable auxiliar (*aux*)

```
    actual = inicial;
    while (actual <= final && bisiesto == 0)
```

```
    {
        if (actual%4 == 0 && (actual%100!=0 || actual%400==0))
            bisiesto = 1;
        else
            actual++;
    }
```

} Determina si un determinado año (el que está actualmente en la variable *actual*) es bisiesto

```
    if (bisiesto == 1)
        printf("\nEl primer año bisiesto entre %i y %i es %i\n", inicial, final, actual);
    else
        printf("\nNo hay ningun año bisiesto entre %i y %i\n", inicial, final);
}
```

Se emplea para ir introduciendo en la variable *actual* todos los posibles años en el rango dado

## 6.3.- Instrucciones iterativas

### ■ Bucle **for**:

- Tiene la condición al principio
- Además incluye inicialización y actualización
- Sintaxis:

```
for (sentencia_inicio; condición; sentencia_actualización)
    sentencia;
```

```
for (sentencia_inicio; condición; sentencia_actualización)
{
    sentencia1;
    ...
    sentenciaN;
}
```

## 6.3.- Instrucciones iterativas

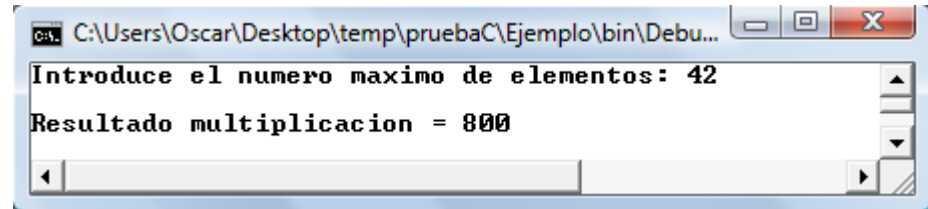
- Ejemplo: multiplicar todos los elementos pares hasta uno dado (introducido por teclado) que sean múltiplos de 4 y 10

```
#include "stdio.h"

void main()
{
    int maximo, cont;
    long int mult = 1;

    printf("Introduce el numero maximo de elementos: ");
    scanf("%i", &maximo);

    if (maximo > 0)
    {
        for (cont=2; cont<=maximo; cont=cont+2)
        {
            if (cont % 4 == 0 && cont % 10 == 0)
                mult = mult * cont;
            printf("\nResultado multiplicacion = %li\n", mult);
        }
    }
    else
        printf("\nError, el valor debe ser positivo.\n");
}
```



**Inicialización:**  
le asigna a la variable cont el valor 2

**Condición:** si se cumple entonces se repite el bucle sino se pasa a la siguiente instrucción

**Incremento:** en cada iteración del bucle la variable cont se incrementa en 2

## 6.3.- Instrucciones iterativas

- En el bucle **for** puede omitirse cualquiera de las sentencias (inicialización y actualización) o la condición:

```
for ( ; ; ) {  
    sentencia1;  
    sentencia2;  
}
```

```
for ( ; c<=10 ; c=c+1) {  
    sentencia1;  
    sentencia2;  
}
```

- Se pueden inicializar y actualizar más de una variable:
  - Separadas por comas

```
for (c=0, a=0.25, s=1 ; s<20 ; s=s+1, c=c*2)  
{  
    a++;  
    printf("\nValor de a: %.2f", a);  
}
```

## 6.3.- Instrucciones iterativas

- Todo bucle **for** es equivalente a un **while**:
  - Con la sentencia inicial antes del bucle
  - Con la sentencia de actualización al final

```
sentencia_inicio;  
while (condicion) {  
    sentencia1;  
    ...  
    sentenciaN;  
    sentencia_actualización;  
}
```



## 6.3.- Instrucciones iterativas

- Ejemplo:

```
for (suma=0, cont=1 ; cont<= 10; cont=cont+1)
{
    cuadrado = cont * cont;
    suma = suma + cuadrado;
}
```

**while** equivalente:

```
suma = 0;
cont = 1;
while (cont<=10) {
    cuadrado = cont * cont;
    suma = suma + cuadrado;
    cont = cont + 1;
}
```

## 6.3.- Instrucciones iterativas

- Bucles anidados:
  - Bucle dentro de otro
  - El bucle interno se ejecuta completamente para cada paso del bucle externo

```
for (a=0; a<10; a++)  
    for (b=1; b<=10; b++)  
        printf("\n¿Cuántas veces aparezco en pantalla?");
```

## 6.3.- Instrucciones iterativas

- Ejemplo: Tabla de multiplicar del 1 al 8

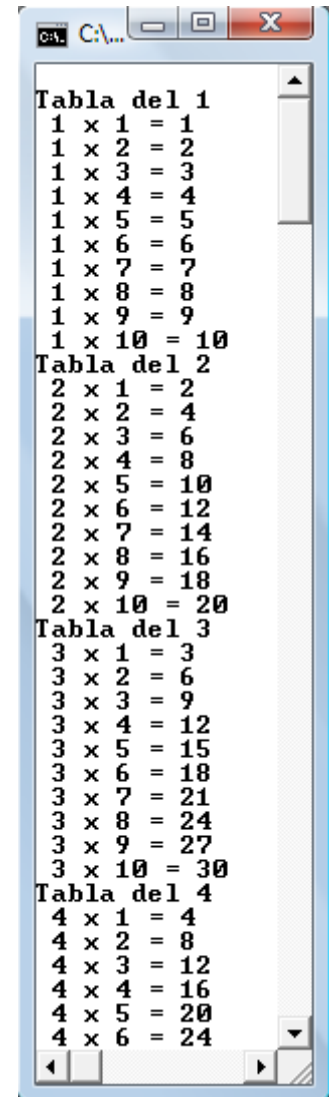
```
#include "stdio.h"

#define MAX 10

void main()
{
    int num, mult, resultado;

    for (num=1; num<=8; num++)
    {
        printf("\nTabla del %d", num);

        for (mult=1; mult<=MAX; mult=mult+1)
        {
            resultado = num*mult;
            printf("\n %d x %d = %d", num, mult, resultado);
        }
    }
}
```



```
Tabla del 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
Tabla del 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
Tabla del 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
Tabla del 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
```

## 6.3.- Instrucciones iterativas

- Ejemplo: muestra el primer divisor, distinto de 1, para todos los números entre 2 y uno dado por el usuario

```
#include <stdio.h>

int main()
{
    int maximo, num, div, primero;

    do{
        printf("Introduce el valor maximo (>1): ");
        scanf("%i", &maximo);
    } while(maximo<=1);

    for (num=2; num<=maximo; num++)
    {
        printf("\nPrimer divisor de %i distinto de 1:", num);
        for (div=2, primero=0; div<=num/2 && primero == 0; div++)
        {
            if (num%div==0)
            {
                printf(" %i", div);
                primero = 1;
            }
        }
        if (primero == 0)
            printf(" -");
    }
    return 0;
}
```

```
Introduce el valor maximo (>1): 18

Primer divisor de 2 distinto de 1: -
Primer divisor de 3 distinto de 1: -
Primer divisor de 4 distinto de 1: 2
Primer divisor de 5 distinto de 1: -
Primer divisor de 6 distinto de 1: 2
Primer divisor de 7 distinto de 1: -
Primer divisor de 8 distinto de 1: 2
Primer divisor de 9 distinto de 1: 3
Primer divisor de 10 distinto de 1: 2
Primer divisor de 11 distinto de 1: -
Primer divisor de 12 distinto de 1: 2
Primer divisor de 13 distinto de 1: -
Primer divisor de 14 distinto de 1: 2
Primer divisor de 15 distinto de 1: 3
Primer divisor de 16 distinto de 1: 2
Primer divisor de 17 distinto de 1: -
Primer divisor de 18 distinto de 1: 2
```

## 6.4.- Instrucciones de salto

- Alteran la ejecución normal de un bucle (**NO USAR**):
  - Sentencia **break**: permite salir de un bucle
  - Sentencia **continue**: inicia una nueva iteración del bucle ignorando las sentencias que la siguen
  - Se usan habitualmente en combinación con alguna sentencia condicional
  - Ejemplos:

```
for (suma=0, n=1; n<11; n++)  
{  
    scanf("%d", &numero);  
    if (numero==0)  
        break;  
    suma = suma + numero;  
}
```

```
for (suma=0, n=1; n<11; n++)  
{  
    scanf("%d", &numero);  
    if (numero<=0)  
        continue;  
    suma = suma + numero;  
}
```



# **ANEXO:**

## Ejercicios resueltos

## Ejercicio 1: enunciado

- Realizar un programa que muestre en pantalla los  $n$  primeros términos de una progresión geométrica que comience en 1. El programa deberá solicitar dos datos enteros al usuario: la razón de la progresión y el número de términos a mostrar. La razón podrá ser positiva o negativa pero no se considera válido un valor igual a 0. En cuanto al número de términos ha de ser mayor que cero. Si alguno de estos valores no cumple las condiciones mencionadas se le pedirá de nuevo al usuario hasta que introduzca uno válido.
- Ejemplo de funcionamiento:

Introduce la razón de la progresión geométrica: 0

Error, la razón no puede ser cero.

Introduce la razón de la progresión geométrica: 5

Introduce el numero de términos de la progresión: 10

La progresión de razón 5 es: 1 5 25 125 625 3125 15625 78125 390625 1953125

# Ejercicio 1: una posible solución

```
#include <stdio.h>
int main()
{
    int terminos, razon;
    long progresion=1;

    do{
        printf("Introduce la razon de la progresion geometrica: ");
        scanf("%i",&razon);
        if (razon==0)
            printf("Error, la razon no puede ser cero.\n");
    } while(razon==0);

    do{
        printf("Introduce el numero de terminos de la progresion: ");
        scanf("%i",&terminos);
        if (terminos<=0)
            printf("Error, el numero de terminos debe ser mayor que cero.\n");
    } while(terminos<=0);

    printf("La progresion de razon %i es: ", razon);
    while (terminos>0)
    {
        printf("%li ", progresion);
        progresion = progresion * razon;
        terminos = terminos - 1;
    }
    return 0;
}
```

**Inicialización:** se inicializa la variable a 1 porque es el primer término de la progresión (punto de partida)

Lectura de la razón de la progresión geométrica

Lectura del número de términos de la progresión

Se imprime el término actual de la progresión

El bucle se repite si se cumple la condición, en este caso si el número de términos sigue siendo mayor que 0

Se calcula el nuevo elemento de la progresión

Se decrementa el número de términos que quedan por mostrar



## Ejercicio 2: enunciado

- Realizar un programa que muestre en pantalla el último múltiplo de 5 de entre dos números dados por el usuario. Para ello, el programa deberá pedir dos datos, el primero de ellos positivo y el segundo mayor que el primero. Si alguno de ellos no cumple la condición indicada se pedirá de nuevo al usuario hasta que introduzca uno válido.

- Ejemplo 1 de funcionamiento:

```
Introduce el primer dato (positivo): 2  
Introduce el segundo dato (mayor que el primero): 14  
El ultimo múltiplo de 5 entre 2 y 14 es el 10.
```

- Ejemplo 2 de funcionamiento:

```
Introduce el primer dato (positivo): 3  
Introduce el segundo dato (mayor que el primero): 4  
No hay ningún múltiplo de 5 entre 3 y 4.
```

## Ejercicio 2: una posible solución

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

```
int main()
```

```
{
```

```
    int primero, ultimo, numero, multiplo;
```

```
    do
```

```
    {
```

```
        printf("\nIntroduce el primer dato (positivo): ");
```

```
        scanf("%i", &primero);
```

```
    } while (primero<=0);
```

```
    do
```

```
    {
```

```
        printf("\nIntroduce el segundo dato (mayor que el primero): ");
```

```
        scanf("%i", &ultimo);
```

```
    } while (ultimo<=primero);
```

La variable múltiplo se usa para guardar el número buscado (el último múltiplo de 5). Inicialmente se pone a cero para indicar que aún se ha encontrado.

```
    for (multiplo=0, numero=ultimo; numero>=primero && multiplo==0; numero--)
```

```
        if (numero%5 == 0)
```

```
            multiplo = numero;
```

El bucle se repite mientras el número actual es aún mayor o igual que el primero y si la variable múltiplo es todavía igual a cero (que indica que aún no se ha encontrado el valor buscado). Deben cumplirse ambas para que sigamos buscando el número.

```
    if (multiplo == 0)
```

```
        printf("\nNo hay ningun multiplo de 5 entre %i y %i.", primero, ultimo);
```

```
    else
```

```
        printf("\nEl ultimo multiplo de 5 entre %i y %i es el %i.", primero, ultimo, multiplo);
```

```
    return 0;
```

```
}
```

## Ejercicio 3: enunciado

- Realizar un programa en C que lea un número entero positivo  $n$  por teclado, que no sea mayor que 100, y que escriba tres columnas por pantalla:
  - La primera columna muestra los números desde 1 hasta  $n$  de uno en uno.
  - La segunda columna muestra los números desde  $2n$  hasta 2 de dos en dos.
  - La tercera columna contiene un símbolo (+) si el dato de la primera columna es mayor que el de la segunda y un (-) si sucede lo contrario.
- Si el número introducido no cumple las condiciones mencionadas se le pedirá de nuevo al usuario hasta que se obtenga uno válido.
- Ejemplo de funcionamiento:

Dime un numero positivo menor que 100: 6

Resultado:

1	12	(-)
2	10	(-)
3	8	(-)
4	6	(-)
5	4	(+)
6	2	(+)

## Ejercicio 3: una posible solución

```
#include <stdio.h>
#define MAX 100

int main()
{
    int num, col1, col2;

    do
    {
        printf("Dime un numero positivo menor que %i: ", MAX);
        scanf("%i", &num);
    }
    while (num<0 || num>MAX);

    printf("Resultado:\n");

    for (col1=1, col2=2*num; col1 <= num; col1++, col2=col2-2)
    {
        if (col1>col2)
            printf(" %i %3i (+)\n", col1, col2);
        else
            printf(" %i %3i (-)\n", col1, col2);
    }
}
```

Se inicializan dos variables. La primera (col1) permitirá ir desde 1 a el número indicado por el usuario (num). La segunda (col2) para recorrer desde 2num hasta 2.

Se incrementa en una unidad la variable col1 y se decrementa en dos unidades la variable col2 después de cada paso del bucle.

Si la variable col1 es menor que el número indicado por el usuario (almacenado en la variable num) se repite el bucle.

El 3 situado entre el % y la letra i indica que se desea imprimir al menos 3 dígitos por pantalla. Si el número tiene menos de 3 dígitos se rellenan con espacios a su izquierda. Con ello se consigue el efecto de que los números en esa columna queden alineados a la derecha (como aparece en la segunda columna del ejemplo del enunciado)

## Ejercicio 4: enunciado

- Realizar un programa en C que muestre en pantalla una X, formada por asteriscos, del tamaño que indique el usuario por teclado (mayor de 2).
- Ejemplos de funcionamiento:

**Dime el tamaño de la X: 10**

A 10x10 grid of asterisks (\*) forming a diamond shape. The pattern is symmetric about both the horizontal and vertical axes. The top row has 1 asterisk at column 5. The second row has 2 asterisks at columns 4 and 6. The third row has 3 asterisks at columns 3, 5, and 7. The fourth row has 4 asterisks at columns 2, 4, 6, and 8. The fifth row has 5 asterisks at columns 1, 3, 5, 7, and 9. The sixth row has 4 asterisks at columns 2, 4, 6, and 8. The seventh row has 3 asterisks at columns 3, 5, and 7. The eighth row has 2 asterisks at columns 4 and 6. The ninth row has 1 asterisk at column 5. The tenth row has 1 asterisk at column 5.

Dime el tamaño de la X: 15

## Ejercicio 4: una posible solución

```
#include <stdio.h>

void main()
{
    int fil, col, tamaño;

    do
    {
        printf("Dime el tamaño de la X: ");
        scanf("%i", &tamaño);
        if (tamaño < 3)
            printf("Error: el valor debe ser mayor que 2.\n");
    }
    while (tamaño < 3);

    for (fil = 1; fil <= tamaño; fil++)
    {
        for (col = 1; col <= tamaño; col++)
        {
            if ((fil == col) || (fil == (tamaño - col + 1)))
                printf("*");
            else
                printf(" ");
        }
        printf("\n");
    }
}
```

El bucle externo permite variar el valor de la variable `fil` desde 1, con incremento de uno, y mientras que esa variable sea menor que el número de filas totales (indicado por la variable `tamaño`). Esta variable indicará la fila en la que nos encontramos en pantalla.

El bucle interno realiza el mismo proceso que el primero pero con otra variable (`col`) que indicará la columna en la que nos encontramos en pantalla en este momento.

Si estamos en un elemento de la diagonal principal o de la secundaria de la X se imprime un asterisco en caso contrario se imprime un espacio. Para detectar ese caso se mira si el índice de la fila es igual al de la columna (estamos en un elemento de la diagonal principal) o bien si la fila es igual a `tamaño-col+1` (estamos en un elemento de la diagonal secundaria).