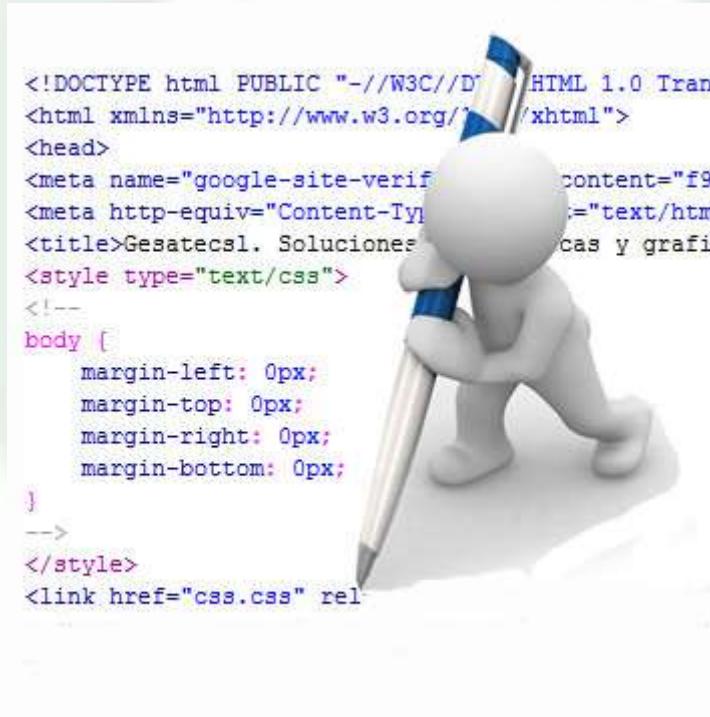


# Elementos de Programación



# Programación estructurada

## Estructura Secuencial

# Programación Estructurada

A finales de los años 1960 surgió una nueva forma de programar que no solamente daba lugar a programas fiables y eficientes, sino que además estaban escritos de manera que facilitaba su comprensión posterior. El *Teorema del Programa Estructurado*, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

- Secuencia
- Instrucción de decisión o condicional.
- Iteración (bucle de instrucciones) con condición conocida o desconocida.

Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, éstas pueden ser construidas mediante las tres básicas.

# Programación Estructurada

## Conceptos Básicos

### ➤ **Variable:**

Se entiende por **variable** a la posición de memoria que contiene datos, recibe un nombre único dado por el programador y que mantiene los datos asignados hasta que un nuevo valor se le asigne o hasta que el programa termine. Lo opuesto de **variable** es **constante** y en este caso se trata de un valor o conjunto de caracteres que permanecen invariables durante la ejecución del programa.

### ➤ **Tipo de dato de Variable (los más importantes):**

**int → entero**

**float → real**

**char → carácter**

# Programación Estructurada

## Conceptos Básicos

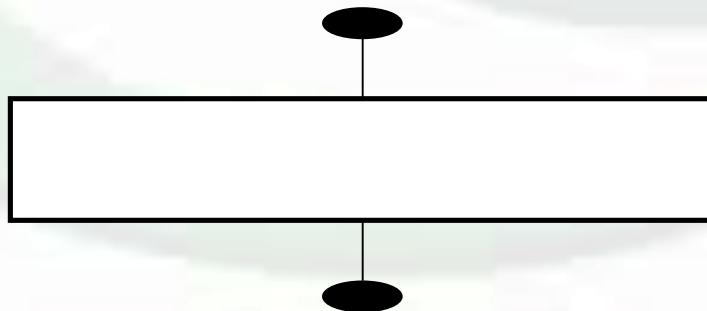
- **Constante:**

Son aquellos valores que no se alteran durante la ejecución del programa. Se diferencian dos tipos de constantes:

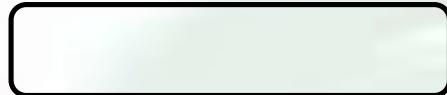
- ✓ **Numéricas:** 56, -5, 8.90E12, -1.25, 3.1415
- ✓ **No numéricas:** ‘\$', “mts”, “grados”, “34-35”

# Estructura Secuencial

Es una sucesión de instrucciones básicas o estructuras que se ejecutan en el mismo orden en que fueron escritas. Cada estructura debe tener un único punto de ENTRADA y un único punto de SALIDA. Podemos decir que todo programa estará compuesto por una estructura secuencial. Una estructura de programa es secuencial si se ejecutan una tras otra a modo de secuencia, es decir que una instrucción no se ejecuta hasta que finaliza la anterior.



# Estructura Secuencial



→ **Comienzo del programa**



→ **Asignación u operación**



→ **Ingreso de datos por teclado**



→ **Salida de información a pantalla de monitor**



→ **Conector**



→ **Fin del programa**

# Estructura Secuencial

## Operadores Aritméticos

Operador	Operación	Observaciones
+	Suma	Operador n-ario
-	Resta	Operador binario
*	Producto	Operador n-ario
/	Cociente	Operador binario (*)
=	Asignación	Operador binario (**)
%	Resto de división entre enteros	Operador binario (***)

(\*) Si los operandos son ambos enteros, el cociente será entero. Con que uno de ellos sea real, el cociente será real.

(\*\*) La asignación SIEMPRE es de **derecha a izquierda** ←

(\*\*\*) Sólo se puede utilizar si los operandos son enteros.

# Estructura Secuencial

## Operadores Aritméticos especiales

Operador	Operación	Observaciones
<code>++</code>	Incrementador de unidad	Operador unario (*)
<code>--</code>	Decrementador de unidad	Operador unario (**)
<code>+=</code>	Acumulador	Operador binario (***)
<code>-=</code>	Restador	Operador binario (***)

(\*) Operador que tiene acción sobre una única variable.

Cont++ es igual a decir  $\text{Cont} = \text{Cont} + 1$

(\*\*) Operador que tiene acción sobre una única variable.

Cont-- es igual a decir  $\text{Cont} = \text{Cont} - 1$

(\*\*\*) Operador que tiene acción sobre dos variables o una variable y una constante

Acum += nro es igual a decir  $\text{Acum} = \text{Acum} + \text{nro}$

Tot -= nro es igual a decir  $\text{Tot} = \text{Tot} - \text{nro}$

# Estructura Secuencial - Ejemplo

Confeccionar un programa que ingrese un valor correspondiente al radio de un círculo y determine e informe la superficie del mismo.

## Datos de Ingreso:

- El valor del radio del círculo (no se conocen unidades de medida).
- Prerrequisito: El valor del radio es un número mayor que cero.

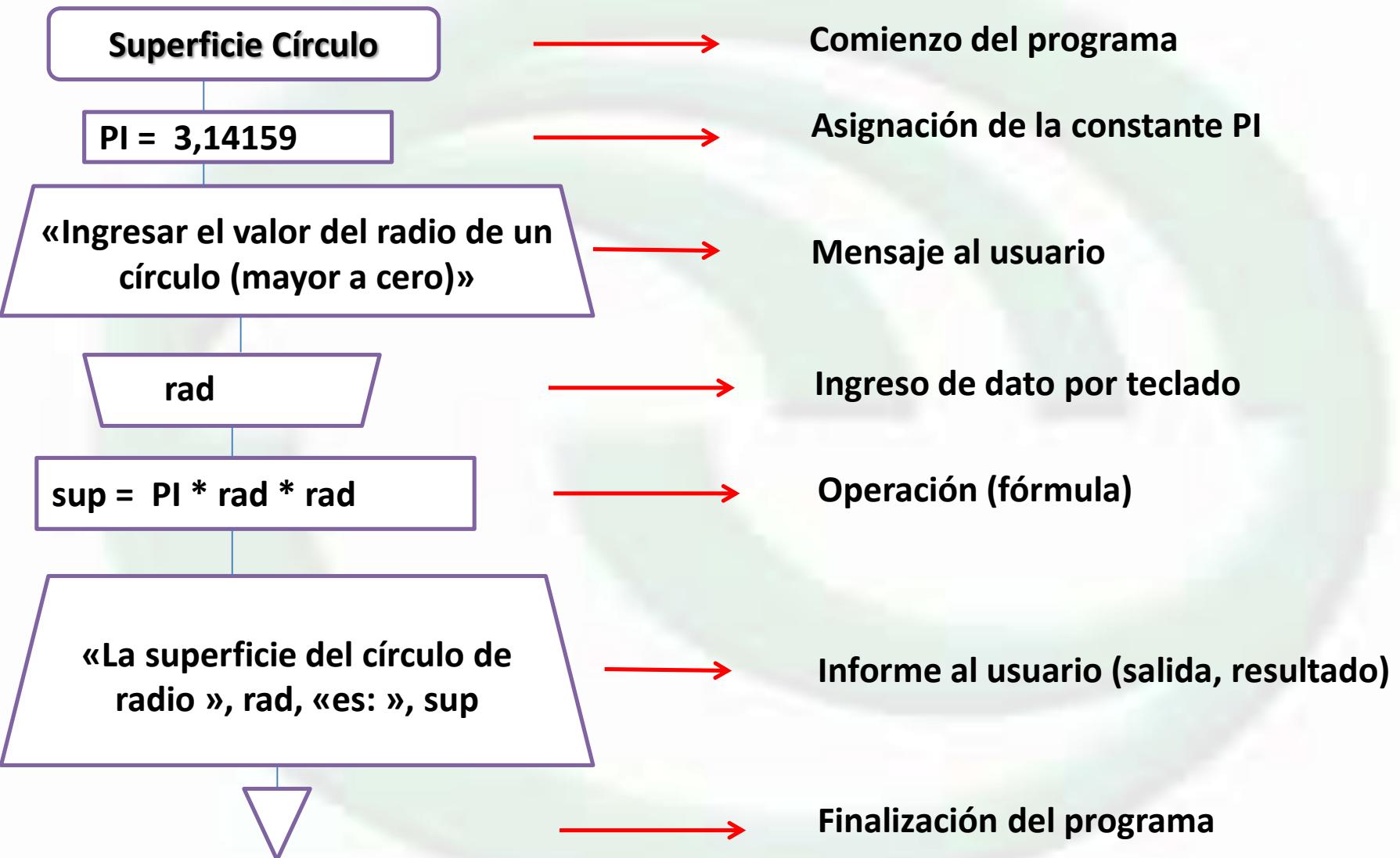
## Proceso:

- Se ingresa el valor del radio del círculo (mayor que cero).
- Se realiza la operación de cálculo de la superficie del círculo.
- Se informa la superficie del círculo.

## Información de Salida:

- El valor de la superficie del círculo (sin unidades de medida).

# Estructura Secuencial - Ejemplo



# Estructura Secuencial - Ejemplo

```
"C:\Users\User\Documents\Universidades\UNLaM\Elementos de Programaci³n\Programas en C\S..." X
Ingresar el valor del radio de un circulo - Mayor a cero 2.59
La superficie del circulo de radio 2.59 es: 21.07
```



Preguntas????



# Elementos de Programación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta name="google-site-verification" content="f90...>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">  
<title>Gesatecsl. Soluciones de software para empresas y graficos</title>  
<style type="text/css">  
<!--  
body {  
    margin-left: 0px;  
    margin-top: 0px;  
    margin-right: 0px;  
    margin-bottom: 0px;  
}  
-->  
</style>  
<link href="css.css" rel="stylesheet" type="text/css" />
```



Codificando  
en lenguaje  
"C"

# Herramientas y/o entornos de programación en C

- Borland C++
- Codeblocks
- Dev C++
- Visual Studio (.net)
- Text Editor / Movil C (Android)

(Editor)

Código fuente .c .cpp →

(Compilador-traductor)

Código objeto .o .obj →

(Enlazador)

Código ejecutable .exe

# Bloques Declarativos

- **General**
  - Directivas al procesador
    - #include< biblioteca>
    - #define var valor
  - Declaración de variables Globales
  - Declaración de estructuras
  - Prototipo de funciones

- **Principal**

```
Int main()
{
    • Declaración de variables locales
    • Sentencias
}
```

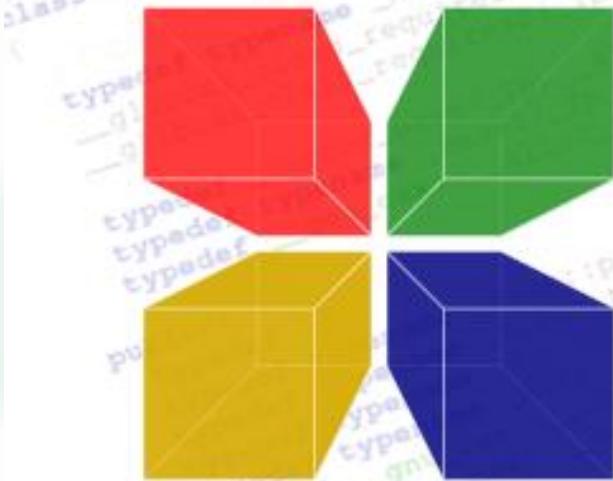
- **Funciones**

# Archivos fuente (bibliotecas)

- stdio.h Funciones E/S a pantalla o archivos
  - printf() scanf() fopen() fclose() feof() etc.
- string.h Funciones con cadenas
  - strcmp() strcpy() strlen() strstr() etc.
- conio.h (No pertenece a ANSI ) Funciones de pantalla
  - clrscr() –solo Borland c++
- windows.h (codeblock-dev) Suplantar conio.h
  - system("cls");
- Otras en apuntes

# Variables

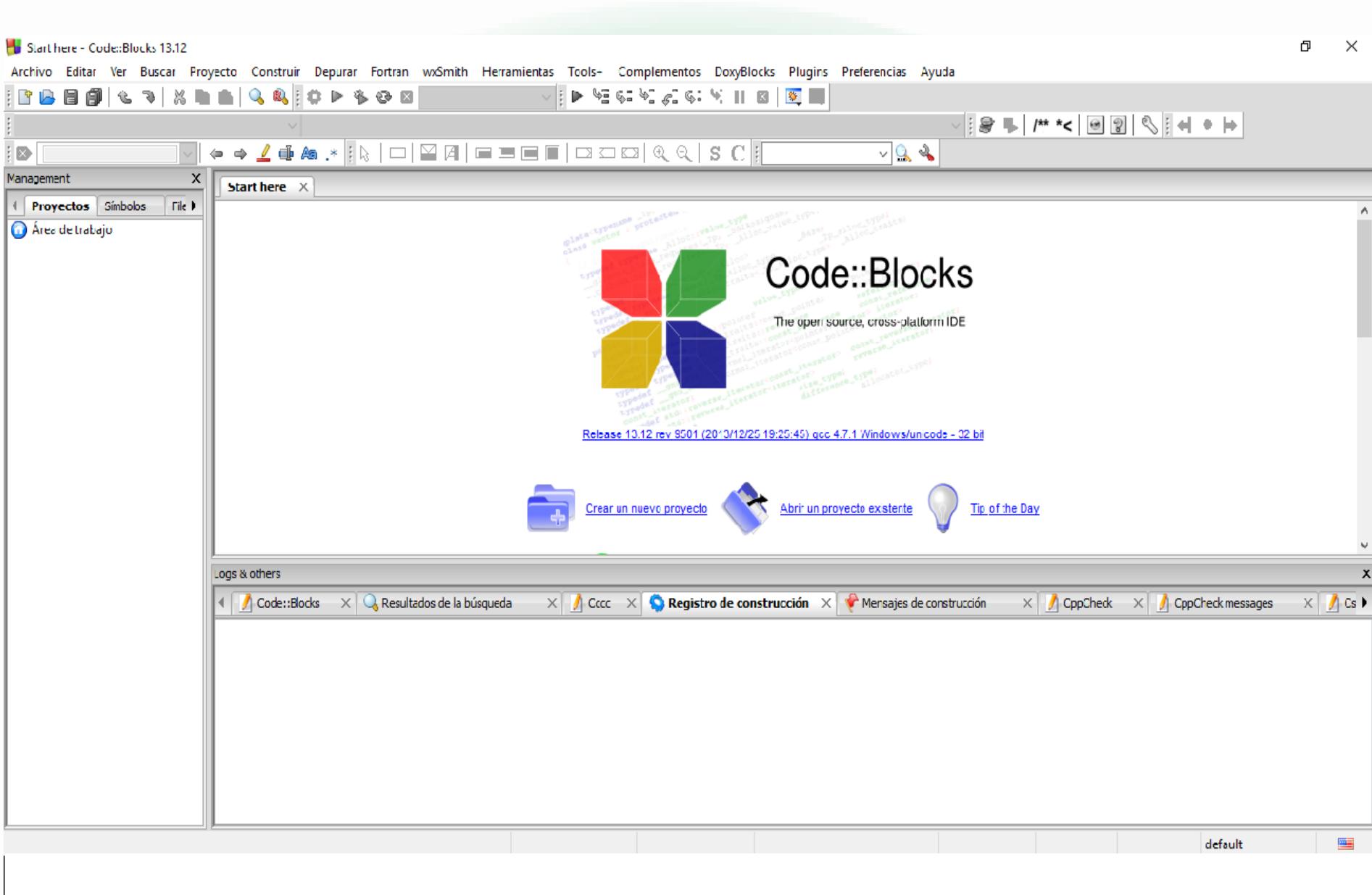
- Tipo (El tamaño depende de tecnología)
  - int
  - float
  - double
  - char
- Modificador de tipo:
  - signed
  - unsigned
  - long
  - short
- Ámbito de incumbencia



# Code::Blocks

The open source, cross-platform IDE

**13.12**



Archivo Editar Ver Buscar Proyecto Construir Depurar Fortran wxSmith Herramientas Tools+

New  
Open... Ctrl-O  
Open with hex editor  
Abrir área de trabajo por defecto  
Proyectos recientes  
Recent files  
Import project

Empty file Ctrl-Shift-N  
Class...  
Project...  
Build target...  
File...  
Custom...  
From template...  
Nassi Shneiderman diagram

Start here \*Untitled1 Instruction Ejercicio3.c \*Untitled2

1

# printf()

## (print formatted)

*Escribe datos en el flujo de salida estándar (stdout)*

- printf("Cadena",argumentos)
- cadena:
  - Leyenda
  - Secuencia de escape.
    - \n        \t        \" \"        \r
  - Especificaciones de formatos → argumentos
    - %d entero
    - %f float %.2f float con decimales limitados
    - %c carácter
    - %s array de char
    - %d para char muestra el ASCII correspondiente
- Lista de argumentos

# `scanf()`

(*scan-format*, analizar con formato)

***Lee datos del flujo de entrada estándar (stdin)***

- `scanf("formatos", argumentos)`
- Formatos
  - Idem `printf()`
- argumentos
  - `&identificador` (`&` como dirección de...)
  - Ausencia `&` no detecta error
  - Leer notas adicionales en apunte teórico.

# if()

- if(Una o + Preguntas lógicas)  
{ /\*indicador de comienzo de bloque\*/  
    Sentencia 1;  
    Sentencia 2;  
}  
else  
{ sentencias....;}

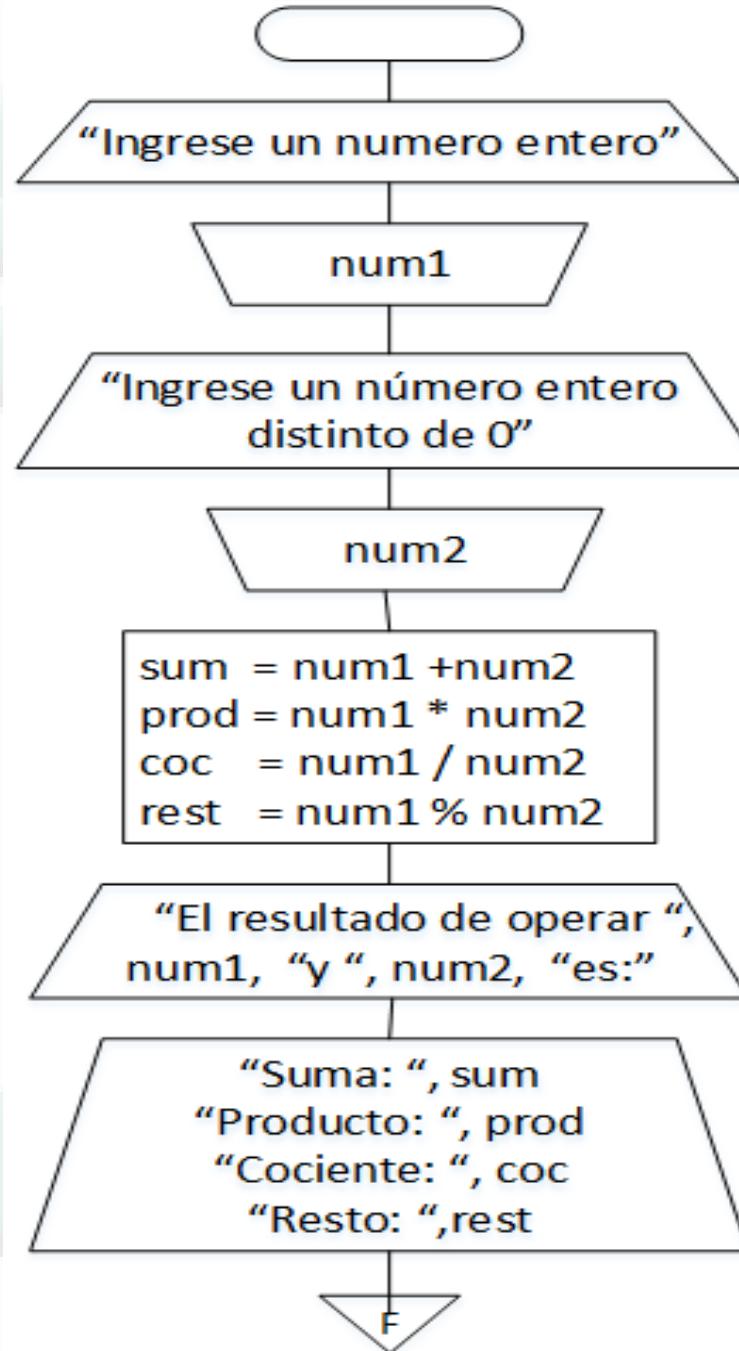
# switch()

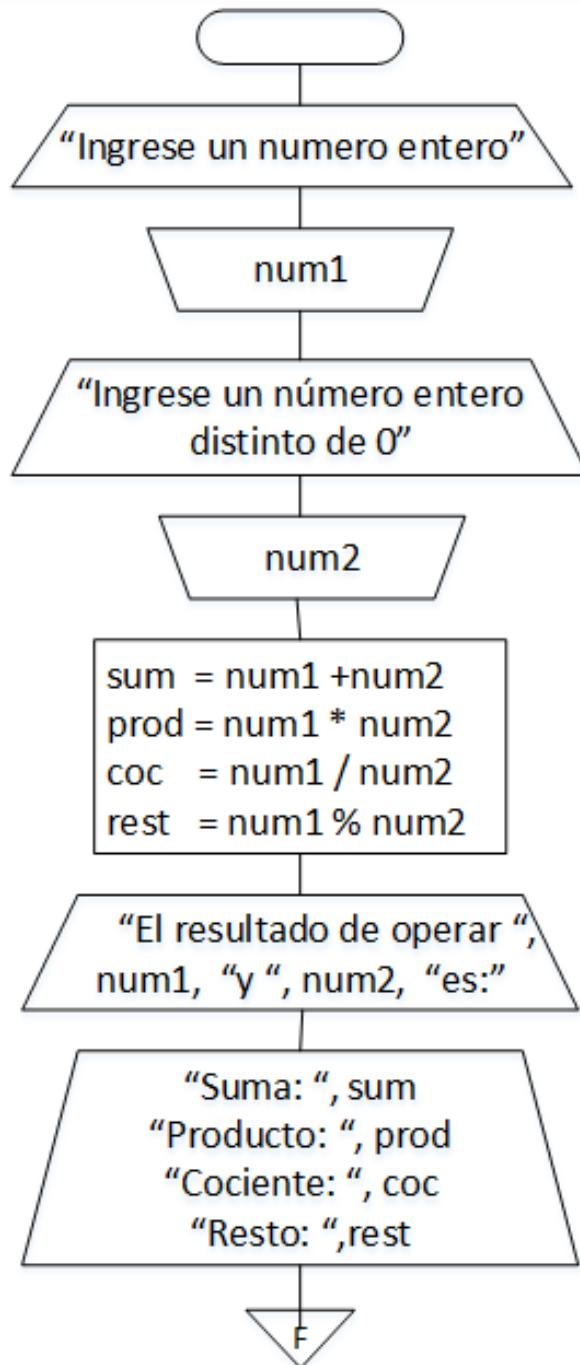
```
switch( variable selectora)
{
    case “opción”: sentencias;
                    break;
    case “opción”: sentencias;
    case...
    default: sentencias;
}
```

# Unidad 3

## Ejercicio 1

Opción 1





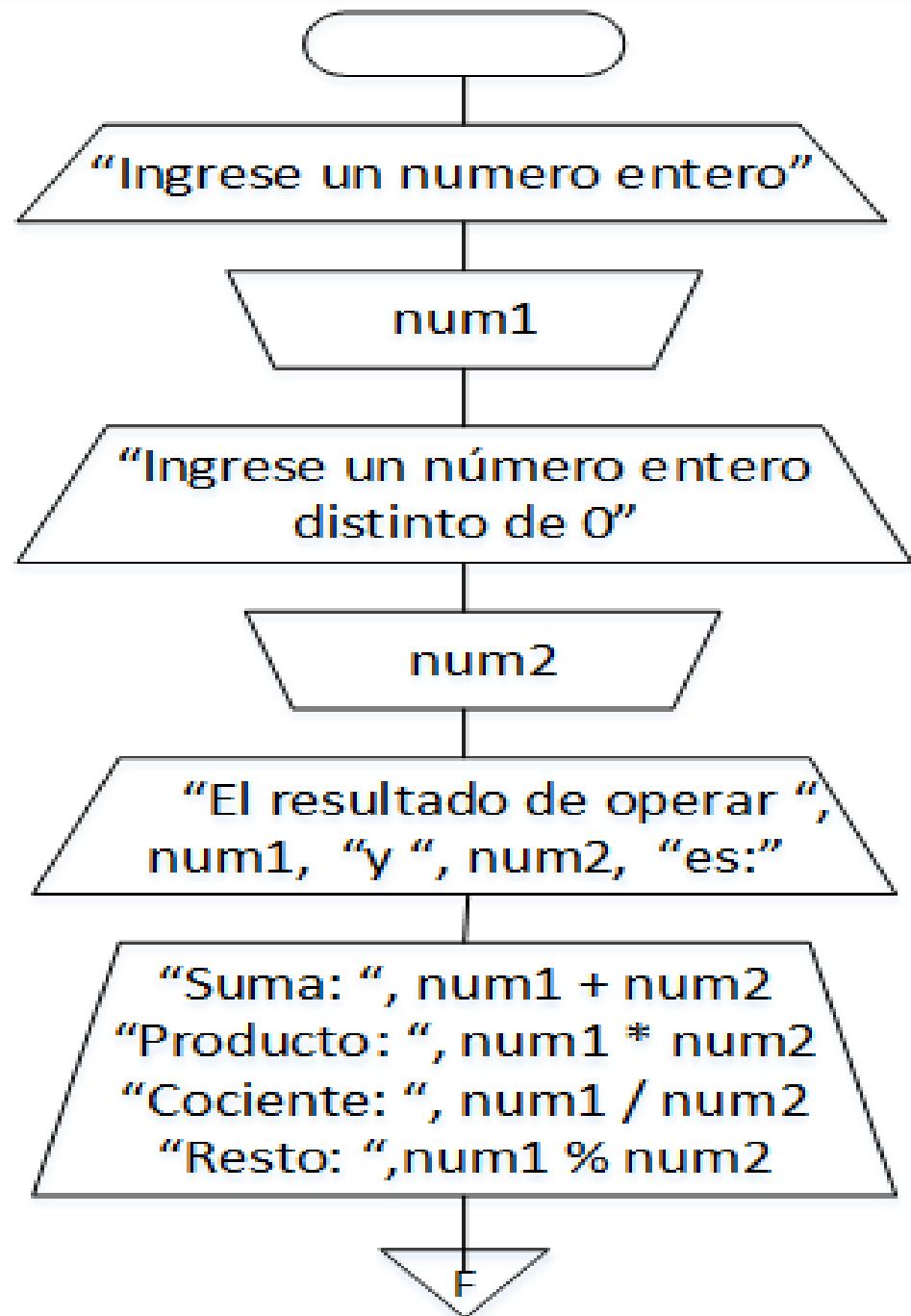
```

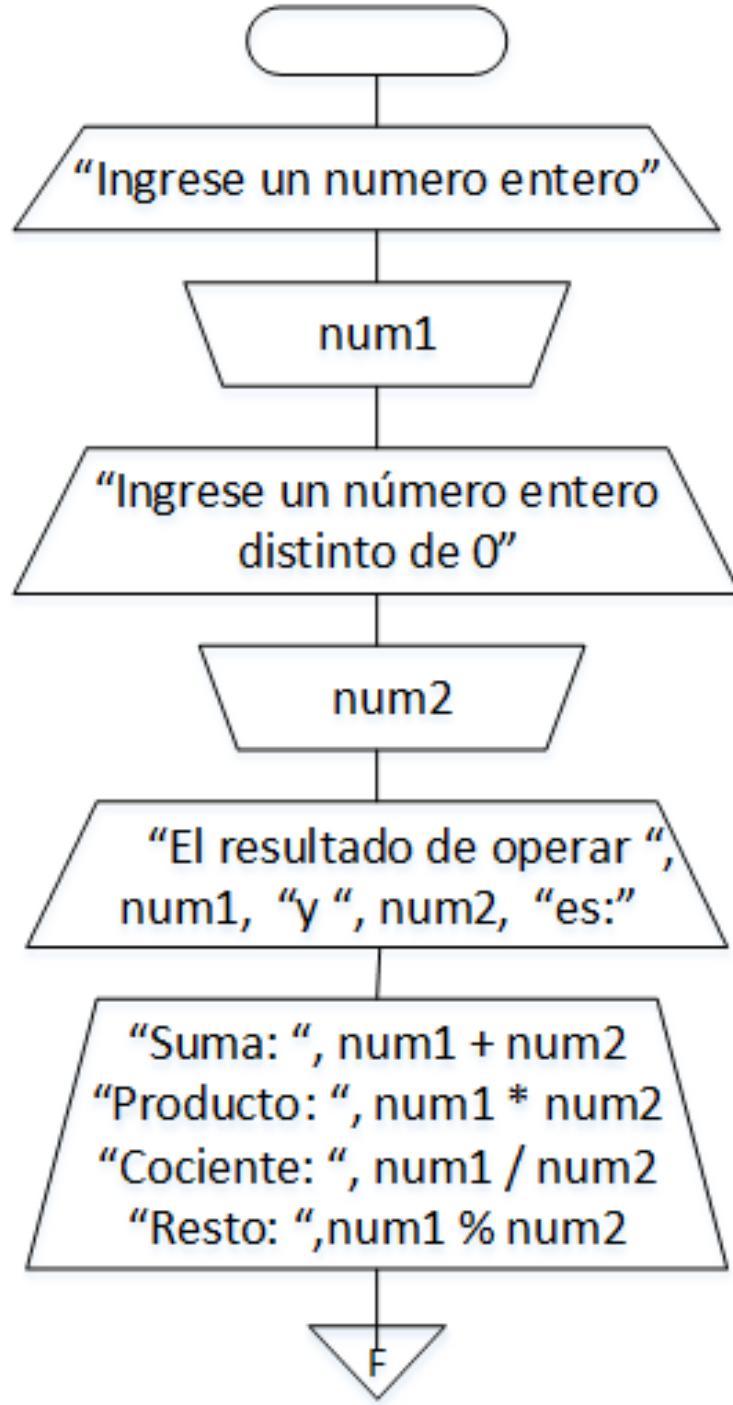
/*Aclaraciones, comentarios y notas*/
#include<stdio.h>
int main()
{
    int num1, num2, sum, prod, coc, res;
    printf("Ingresar un número entero: ");
    scanf("%d", &num1);
    printf("\n Ingresar un número entero distinto de 0: ");
    scanf("%d", &num2);
    sum=num1+num2;
    prod=num1*num2;
    coc=num1/num2;
    res=num1%num2;
    printf("\n El resultado de operar %d y %d es:", num1, num2);
    printf("\n\n Suma: %d",sum);
    printf("\n Producto: %d",prod);
    printf("\n Cociente: %d",coc);
    printf("\n Resto: %d",res);
}
  
```

# Unidad 3

## Ejercicio 1

Opción 2





```
#include<stdio.h>

Int main()
{
    int num1, num2;

    printf("Ingrese un numero entero: ");
    scanf("%d", &num1);

    printf("\n Ingrese un número entero distinto de 0: ");
    scanf("%d", &num2);

    printf("\n El resultado de operar %d y %d es:", num1,
           num2);

    printf("\n\n Suma: %d", num1+num2);
    printf("\n Producto: %d", num1*num2);
    printf("\n Cociente: %d", num1/num2);
    printf("\n Resto: %d", num1%num2);
}
```

# Ejercicio

- Ingresar 3 valores reales que correspondan a longitudes e informar
  1. Si forman o no triángulo.
  2. Si es equilátero, isósceles o escaleno.
  3. Si es triángulo rectángulo.

```
//Comienzo e ingreso de datos
#include<stdio.h>
Int main()
{
    float l1, l2, l3;
    printf("\n Ingrese 3 longitudes con valores mayores a 0");
    printf("\n Lado 1: ");
    scanf("%f", &l1);
    printf("\n Lado 2: ");
    scanf("%f", &l2);
    printf("\n Lado 3: ");
    scanf("%f", &l3);
```

```
if (l1>0 && l2>0 && l3>0)
{  if (l1+l2>l3 && l2+l3>l1 && l1+l3>l2)
{
    if (l1==l2 && l1==l3) printf("\nEl triangulo es equilatero");
    else
    {  if (l1==l2 || l2==l3 || l1==l3)
        printf("\nEl triangulo es isosceles");
        else printf("\nEl triangulo es escaleno"); }
    if (l1*l1+l2*l2==l3*l3 || l1*l1+l3*l3==l2*l2 || l3*l3+l2*l2==l1*l1)
        printf("\n\nEl triangulo es rectangulo\n\n");
    else printf("\n\nEl triangulo NO es rectangulo\n\n");
}
else printf("\n Los lados NO forman triangulo\n\n");
}
else printf("\n Ingrese valores validos\n\n");
}
```

**/\*Ingrese un número entero que represente un mes del año e indique a qué mes corresponde\*/**

```
#include<stdio.h>
int main()
{ int mes;
printf("\nIngrese un numero entero entre 1 y 12: ");
scanf("%d",&mes);
switch (mes)
{ case 1: printf("\nEl mes %d corresponde a enero", mes); break;
  case 2: printf("\nEl mes %d corresponde a febrero", mes); break;
  case 3: printf("\nEl mes %d corresponde a marzo", mes); break;
  case 4: printf("\nEl mes %d corresponde a abril", mes); break;
  case 5: printf("\nEl mes %d corresponde a mayo", mes); break;
  case 6: printf("\nEl mes %d corresponde a junio", mes); break;
  case 7: printf("\nEl mes %d corresponde a julio", mes); break;
  case 8: printf("\nEl mes %d corresponde a agosto", mes); break;
  case 9: printf("\nEl mes %d corresponde a septiembre", mes); break;
  case 10: printf("\nEl mes %d corresponde a octubre", mes); break;
  case 11: printf("\nEl mes %d corresponde a noviembre", mes); break;
  case 12: printf("\nEl mes %d corresponde a diciembre", mes); break;
  default: printf("\nValor fuera de rango");
}
printf("\n\n");}
```

```
/*Ingrese un número entero que represente un mes del año*/
#include<stdio.h>
int main()
{int mes;
printf("\nIngrese un numero entero entre 1 y 12:");
scanf("%d",&mes);
switch (mes)
{
case 1:
case 2:
case 3:
case 4:
case 5:
case 6: printf("\nEl mes %d corresponde al primer semestre", mes); break;
case 7: case 8: case 9: case 10: case 11: case 12:
printf("\nEl mes %d corresponde al segundo semestre", mes);break;
default: printf("\nValor fuera de rango");
}
printf("\n\n");}
```



**¿CONSULTAS?**

MUCHAS GRACIAS  
POR SU ATENCIÓN.

MANOS A LA OBRA y...

SUERTE





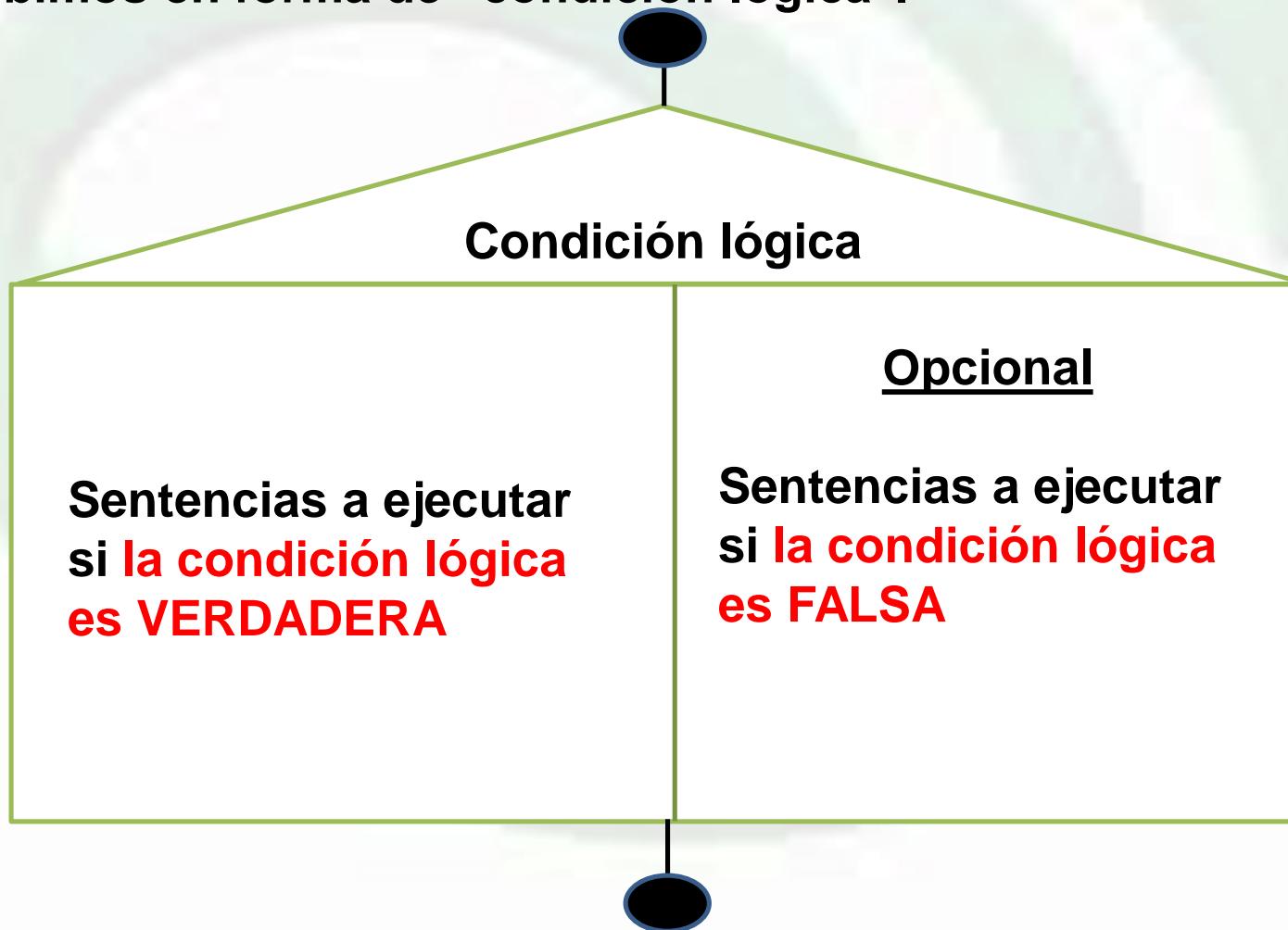
# Elementos de Programación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN" "http://www.w3.org/TR/1999/REC-html401-19991224/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones de software para la creación de casas y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>
<script type="text/javascript" src="js.js">
</head>
<body>
<div id="header">
<h1>Gesatecsl. Soluciones de software para la creación de casas y gráficos</h1>
<h2>Software para la creación de casas y gráficos</h2>
<h3>Software para la creación de casas y gráficos</h3>
<h4>Software para la creación de casas y gráficos</h4>
<h5>Software para la creación de casas y gráficos</h5>
<h6>Software para la creación de casas y gráficos</h6>
</div>
<div id="content">
<h1>Gesatecsl. Soluciones de software para la creación de casas y gráficos</h1>
<h2>Software para la creación de casas y gráficos</h2>
<h3>Software para la creación de casas y gráficos</h3>
<h4>Software para la creación de casas y gráficos</h4>
<h5>Software para la creación de casas y gráficos</h5>
<h6>Software para la creación de casas y gráficos</h6>
</div>
<div id="footer">
<h1>Gesatecsl. Soluciones de software para la creación de casas y gráficos</h1>
<h2>Software para la creación de casas y gráficos</h2>
<h3>Software para la creación de casas y gráficos</h3>
<h4>Software para la creación de casas y gráficos</h4>
<h5>Software para la creación de casas y gráficos</h5>
<h6>Software para la creación de casas y gráficos</h6>
</div>
</body>
</html>
```

Estructura  
Selección  
Simple, compuesta  
y múltiple

# Estructura de Selección Simple

Esta estructura nos permite la selección entre dos posibles cursos de acción, en base a la verdad o falsedad de una expresión que escribimos en forma de "condición lógica".



# Estructura de Selección Simple

## Operadores Relacionales

Operador	Operación
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
==	Igualdad (igual que)
!=	Desigualdad (distinto de)

# Estructura de Selección Simple

## ➤ Ejemplo:

Confeccionar un programa que ingrese dos valores numéricos y determine e informe al mayor de ellos si son distintos, o un mensaje que diga 'IGUALES 'en caso de serlo.

### Datos de Ingreso:

- Dos valores numéricos (no se conocen unidades de medida ni tipo).

### Proceso:

- Se ingresan los valores numéricos.
- Se averigua si son distintos y cuál es el mayor.

### Información de Salida:

- Si los valores son distintos, se informa el mayor.
- Si los valores son iguales, se informa esta situación.

# Estructura de Selección Simple

Valores

“Ingrese dos valores numéricos”

Num1, Num2

Num1 != Num2

Num1 > Num2

“El mayor es: ”, Num1

“El mayor es: ”, Num2

Num1, “y”,  
Num2,  
“son iguales”

# Estructura de Selección Compuesta

Operador	Operación
&&	Y (AND) Conjunción Lógica
	O (OR) Disyunción Lógica
!	NO (NOT) Negación Lógica

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

A	B	A    B
F	F	F
F	V	V
V	F	V
V	V	V

A	! A
F	V
V	F

# Estructura de Selección Compuesta

Ingresar tres valores reales correspondiente a los lados de un triángulo e informar si es equilátero, isósceles o escaleno.

## Triángulos

“Ingrese tres valores reales correspondiente a un triángulo”

L1, L2, L3

$L1 == L2 \ \&& \ L2 == L3$

“Triángulo Equilátero”

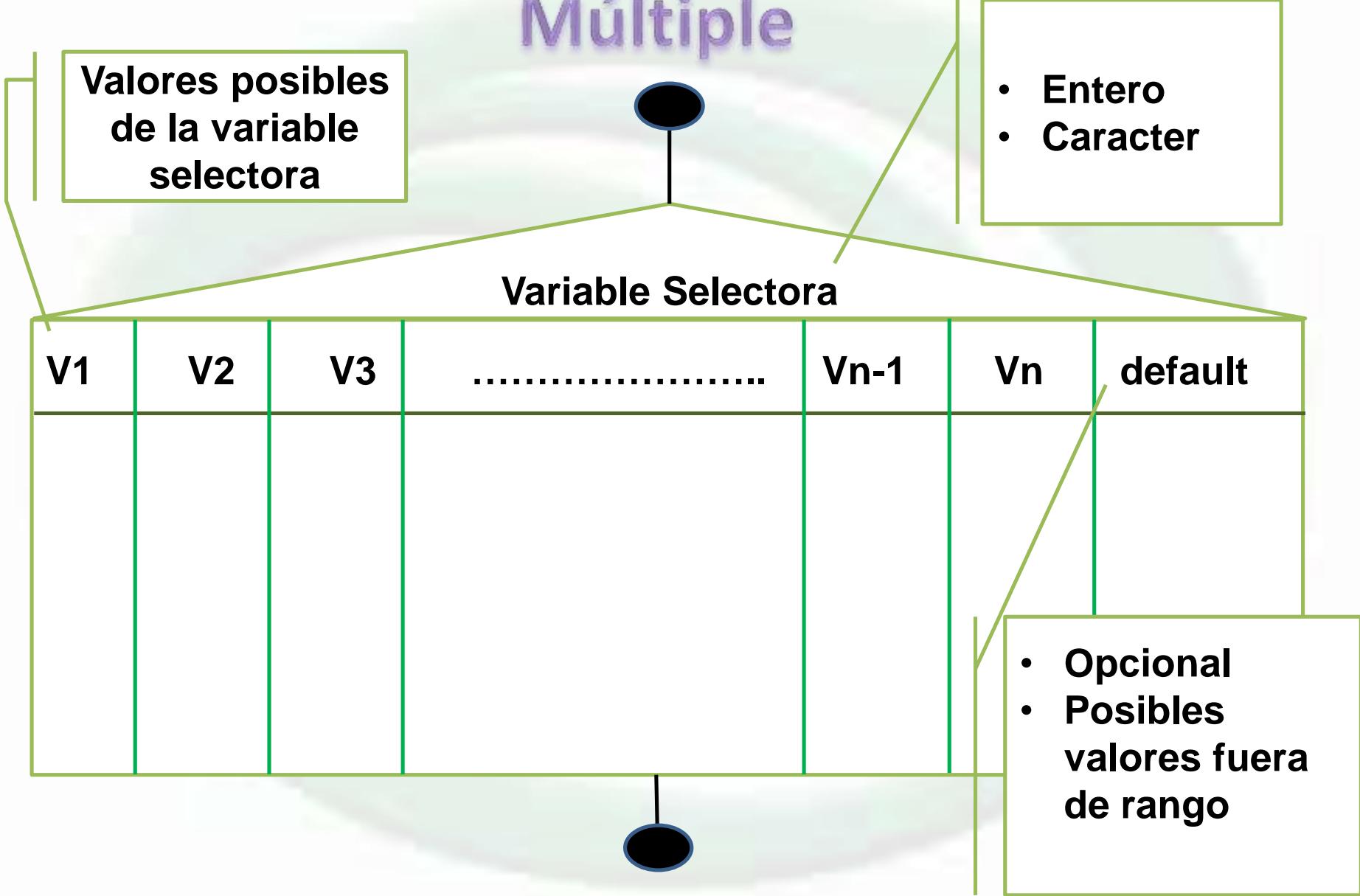
$L1 == L2 \ || \ L1 == L3 \ || \ L2 == L3$

“Triángulo Isósceles”

“Triángulo Escaleno”

# Estructura de Selección

## Múltiple



# Estructura de Selección Múltiple

Ingresar un valor numérico correspondiente a un mes e informar el nombre de dicho mes.

Mes

“Ingrese un valor entero correspondiente a un mes”

mes

mes

1

2

.....

12

default

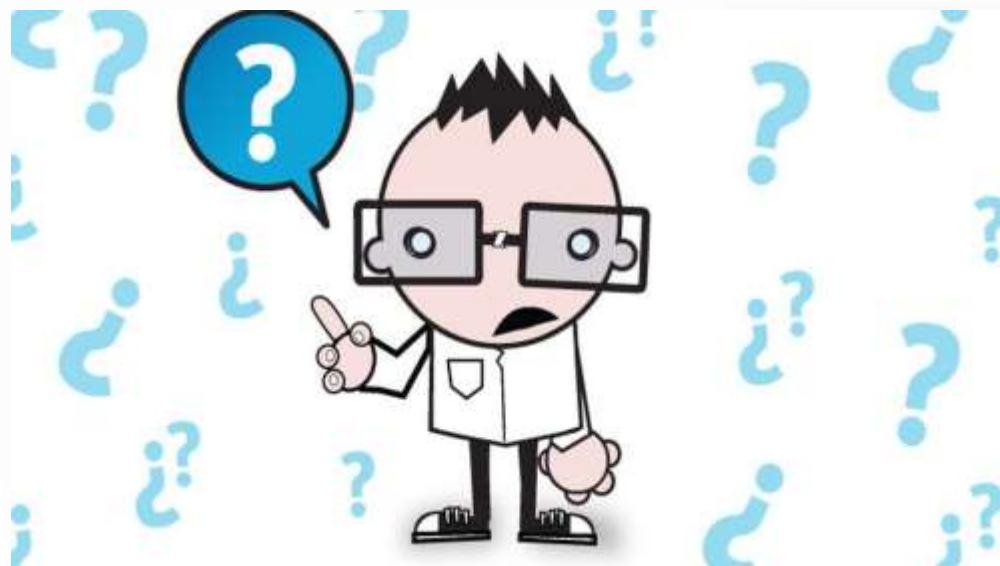
“Ene”

“Feb”

“Dic”

“El  
valor no  
es un  
mes”





# Preguntas????



©PAWS

# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f91..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsol. Soluciones de software y servicios informáticos y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>

```



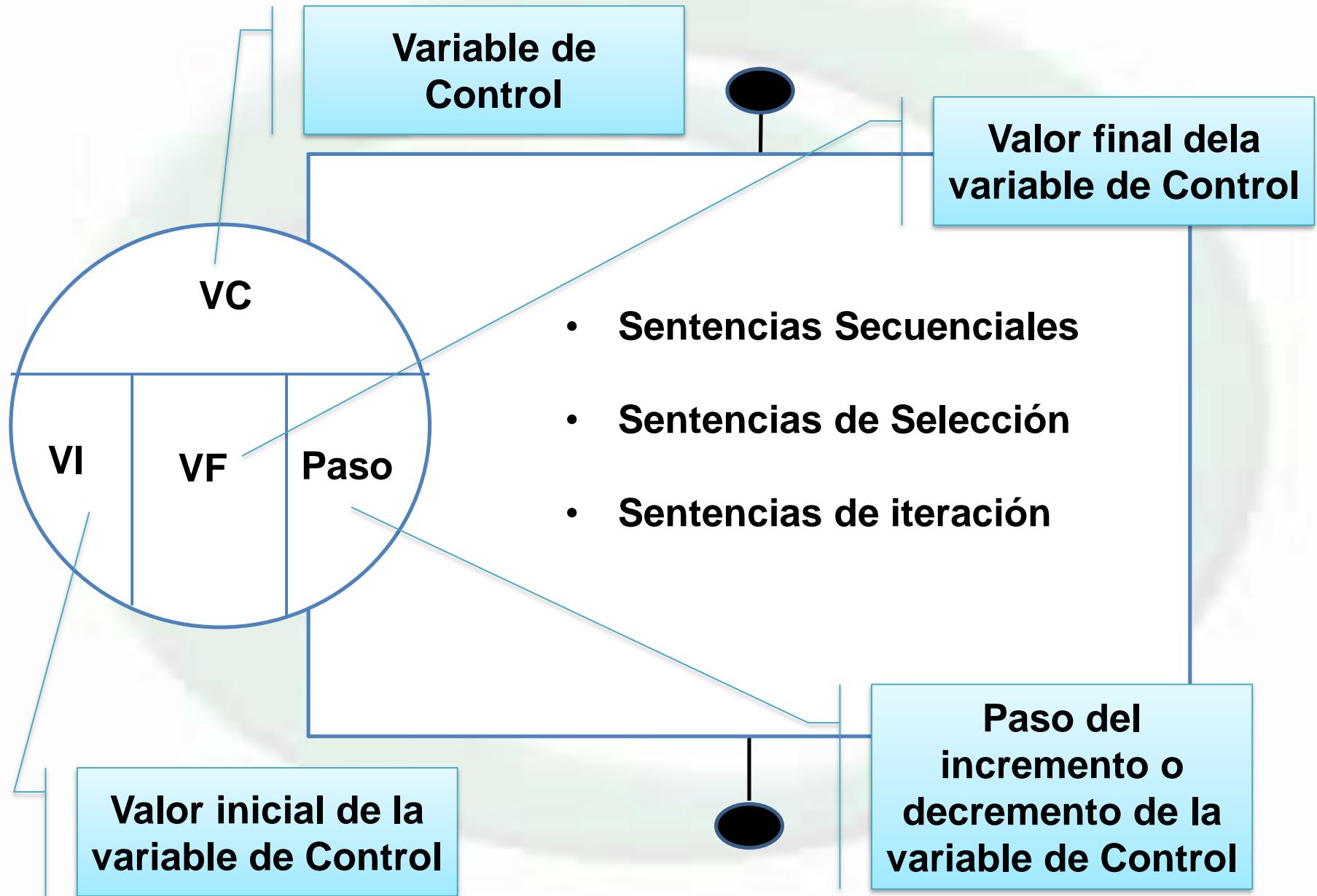
# Estructura Iteración Definida (for)

# Estructuras de Iteración

En los programas vistos hasta ahora cada instrucción, se ejecutaba "una sola vez", en el mismo orden que aparecía en el programa. La ejecución repetida de un grupo de sentencias ofrece una gran posibilidad en los programas que tienen un interés práctico. Existen dos tipos de bucles bien diferentes:

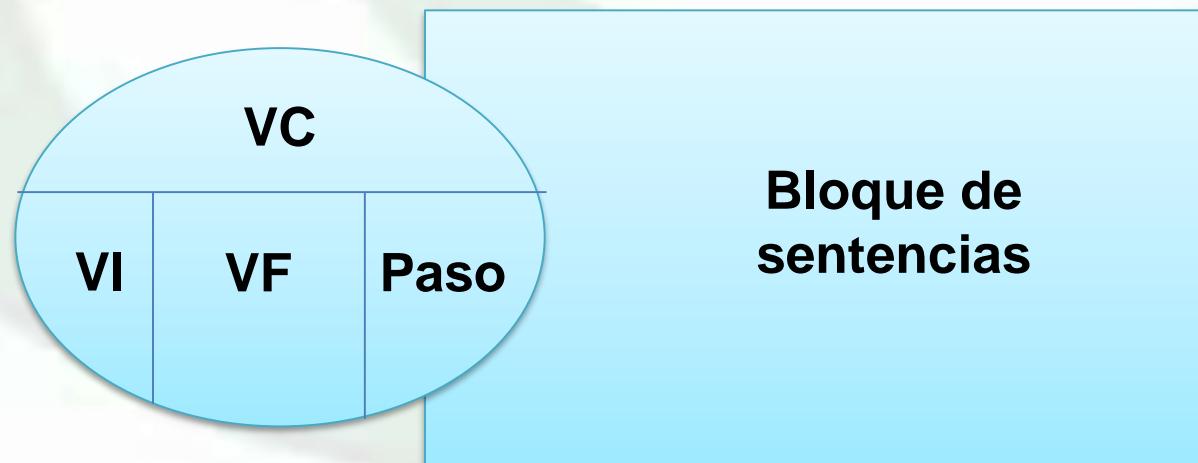
- **Iteración Definida.** Se conoce de antemano la cantidad "exacta" de veces que debo repetir ese proceso o grupo de sentencias.
- **Iteración Indefinida.** NO se conoce de antemano la cantidad de iteraciones a efectuar, o sea que será un ciclo "condicionado" al cumplimiento de una cierta condición.

# Estructuras de Iteración Definida (for)



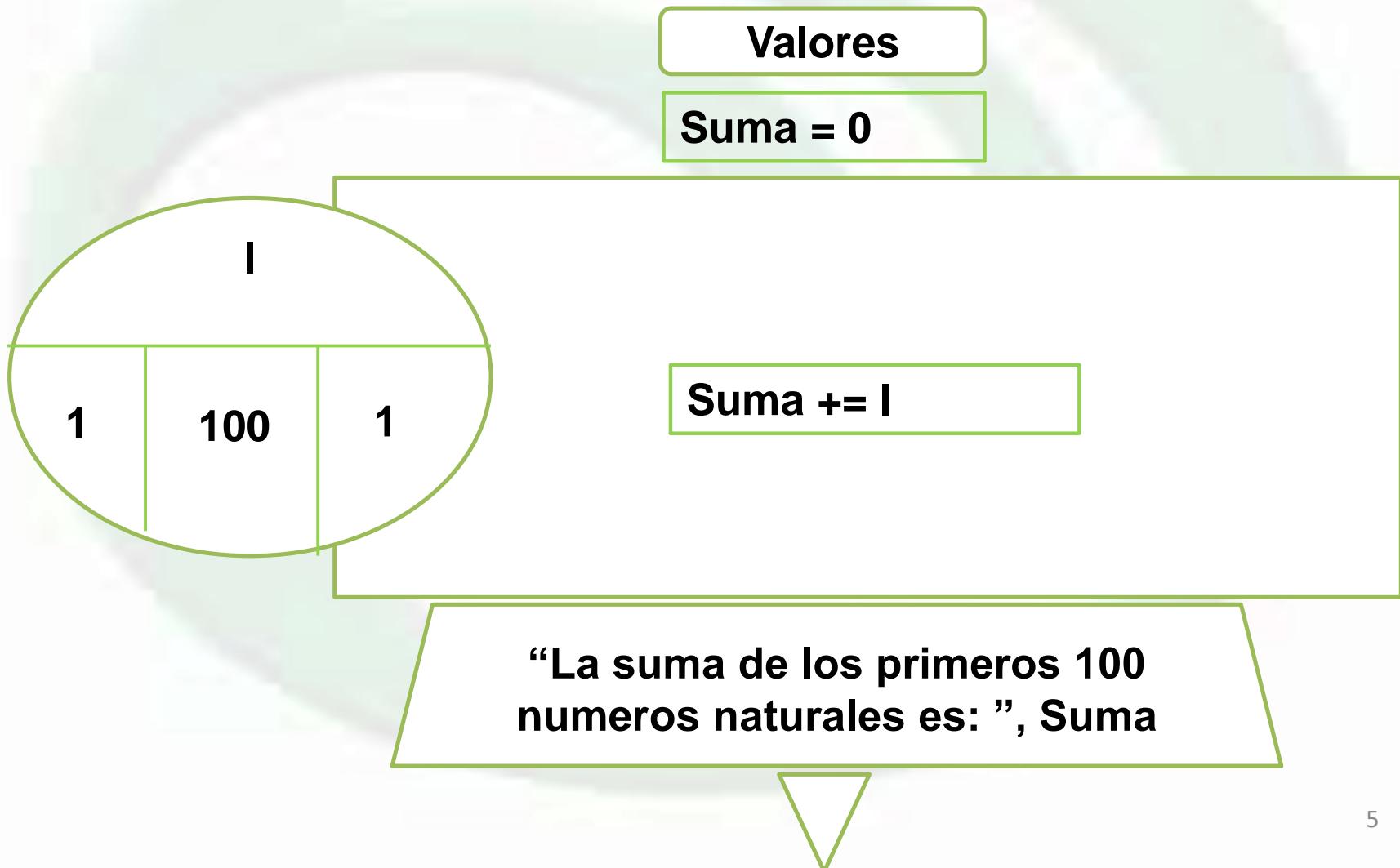
# Estructuras de Iteración Definida (for)

```
for (valor inicial VC ; Rango de repetición; Paso )  
{  
    Bloque de sentencias ;  
}
```



# Estructuras de Iteración Definida (for)

Informar la suma de los primeros cien números naturales.



# Estructuras de Iteración Definida (for)

```
# include <stdio.h>

int main ()
{
    int Suma, I;

    Suma = 0;

    for (I = 1; I <= 100; I++)
        Suma += I;

    printf ("\nLa suma de los primeros 100 numeros naturales
            es: %d", Suma);

    return 0;
}
```

# Estructuras de Iteración Definida (for)

## Observaciones:

- La prueba de la condición de final se efectúa siempre al inicio del bucle, o sea que si el valor inicial es mayor al final el ciclo no se ejecuta nunca, es un ciclo 0 - N
- El valor del índice puede ser modificado dentro del ciclo, pudiendo pasar cualquier cosa. Se pueden generar errores. Por eso, **NO DEBE HACERSE.**
- Finalizado el FOR, el valor del índice (variable de control) queda con el valor de la última ejecución realizada más el valor del incremento, o sea el valor con el cual no pudo ejecutar otra pasada.
- Tener la precaución de “**NO**” colocar un ; luego de los paréntesis del For, lo cual hace ignorar el bucle, generando un ciclo vacío. Usando un bucle vacío puedo generar retardos :  

```
for ( x=0 ; x <= algun_valor ; x++ ) ;
```

El valor asignado a `algun_valor` define el retardo.

# Estructuras de Iteración Definida (for)

- Pueden omitirse una ó todas las expresiones del FOR, pero NO los “;”

a) - Si se omite la primera expresión, NO SE ASIGNA VALOR INICIAL dentro del ciclo, pero puede asignarse antes :

Ejemplo: `j = 1;`

`suma = 0;`

`for ( ; j <=10 ; ++j )`

`suma += j; // Suma los números del 1 al 10.`

b) Si se omite la segunda genero un ciclo INDEFINIDO, porque al no estar la expresión 2, se establece que la prueba siempre será verdadera.

Ejemplo: `for (j=1; ; j++)`

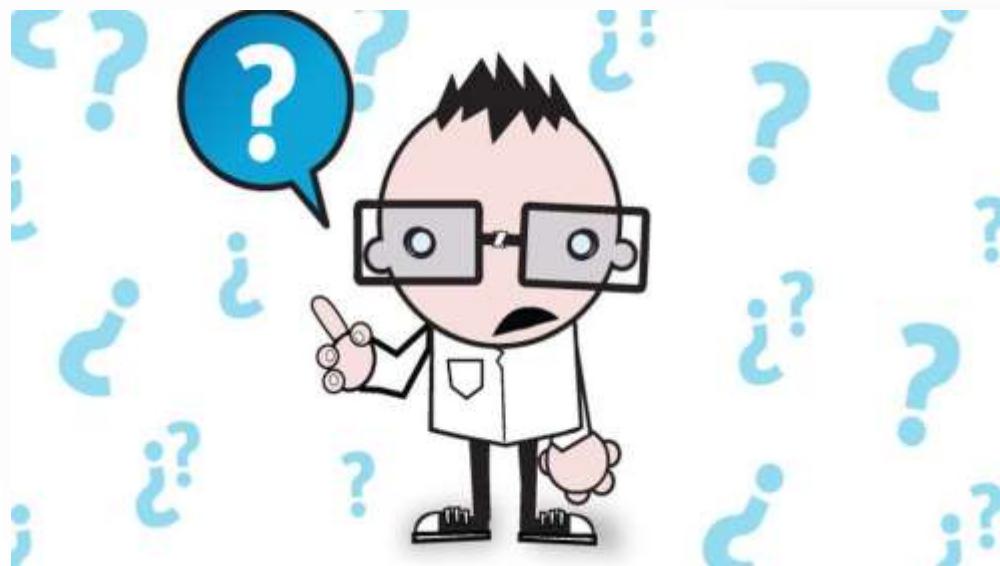
c) Si se omite la tercera, la puedo realizar dentro del ciclo For :

`j =1; suma=0;`

`for ( ; j <=10 ; )`

`suma += j++;`

`printf("%d%d", j, suma);`



# Preguntas????



©PAWS

# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones de software para la educación y la investigación en ciencias y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Estructura Iteración Indefinida o Condicional (while / do-while)

# Estructura de Iteración Indefinida

- La sentencia FOR no da respuesta a los casos en que desconocemos la cantidad de datos a procesar, sin contarlos previamente.
- Tampoco nos permitirá usar el programa para otro juego de datos que no posea la misma cantidad de valores que el anterior.
- Existen otras estructuras iterativas que permiten una mayor flexibilidad en la resolución de los problemas, sin tener que contar previamente la cantidad de datos.
- Esta estructura requiere que pueda establecerse previamente una condición para poder finalizar, que en general es sencilla de definir.

# Estructuras de Iteración Indefinida (while)

## CONDICIÓN LÓGICA

- **Sentencias Secuenciales**
- **Sentencias de Selección**
- **Sentencias de iteración**

# Estructuras de Iteración Indefinida (while)

```
while      (   Condición Lógica   )  
{  
    Bloque de sentencias ;  
}
```

Condición Lógica

Bloque de  
sentencias

# Ejemplo de while

Ingresar las edades de los alumnos del curso. Informar el promedio de edades. El fin de carga se detecta con una edad igual a cero.

Edades

Suma = 0

Cont = 0

“Ingresar las edades de los alumnos – cero para terminar”

edad

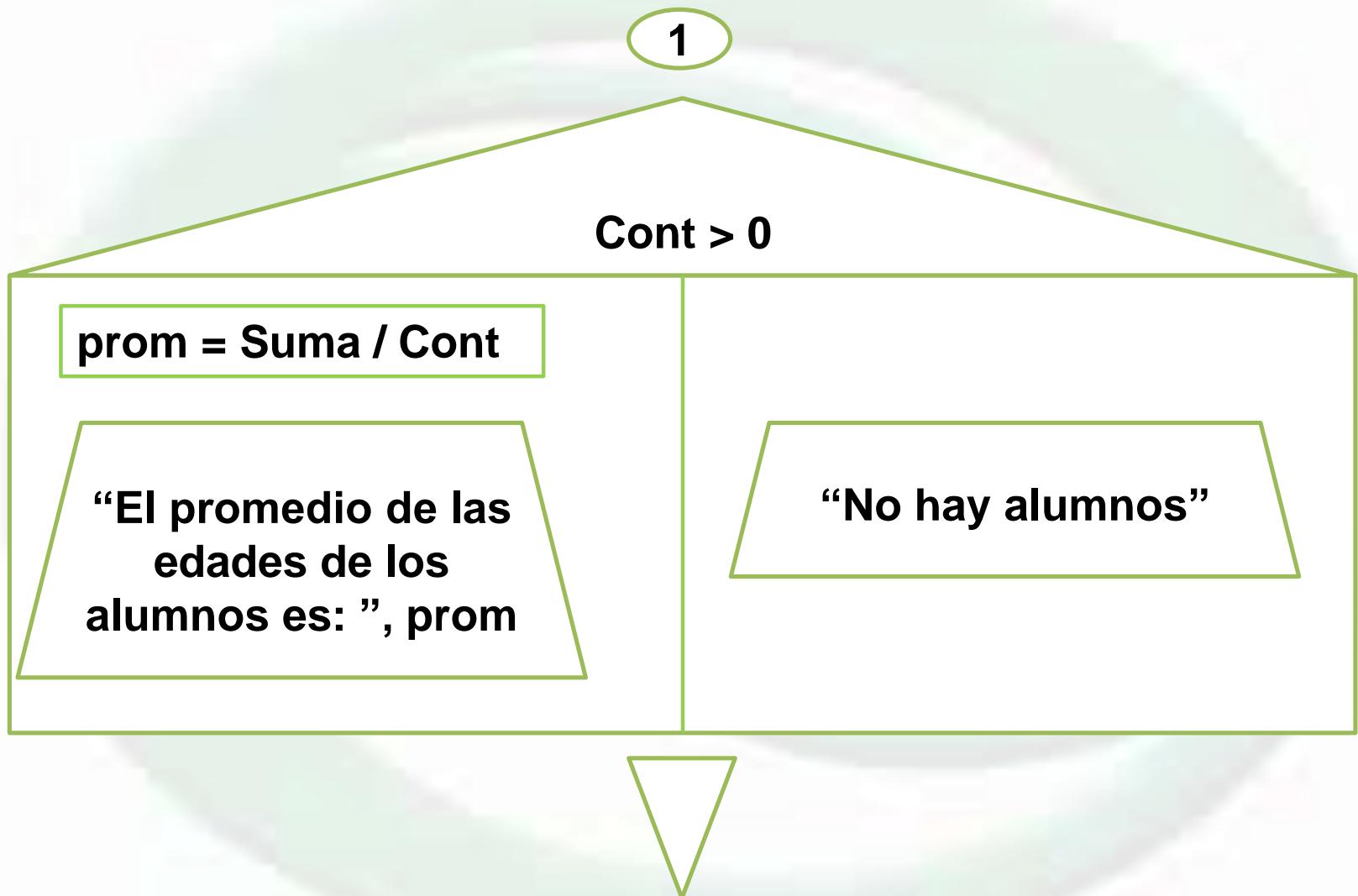
edad > 0

Suma += edad

Cont++

edad

# Ejemplo de while



# Ejemplo de while

```
#include <stdio.h>

int main ()
{ int Suma = 0, Cont = 0, prom, edad;
  printf ("\nIngrese las edades de los alumnos - Cero para terminar... ");
  scanf ("%d", &edad);
  while (edad > 0)
    { Suma += edad;
      Cont++;
      printf ("\nIngrese las edades de los alumnos - Cero para
terminar... ");
      scanf ("%d", &edad);  }

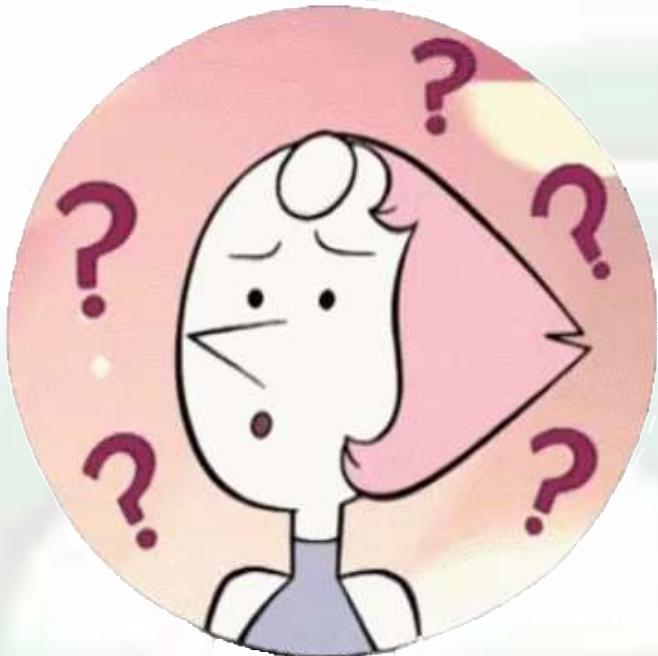
  if (Cont > 0)
    { prom = Suma / Cont;
      printf ("\nEl promedio de las edades de los alumnos es %d",
prom);  }
  else
    printf ("\nNo hay alumnos");
  return 0;}
```

# Estructuras de Iteración while

## Observaciones:

- Notar que en el Ejemplo, la finalización de carga está condicionado al valor de una variable y existen DOS ordenes de lectura. ¿Que pasaría si elimino la orden de lectura dentro del ciclo ?
- No confundir con la sentencia IF, esta tiene una condición que se analiza y ejecuta una sola vez, en cambio el while genera un ciclo condicionado.
- Se puede generar un ciclo sin sentencia si colocamos el ; luego de la expresión del while.
- Es posible que el bucle del while no se ejecute nunca si la condición es falsa la primera vez que se evalúa.

**Rango de ejecución:  
0 a n veces**



# Preguntas????



# Elementos de Programación



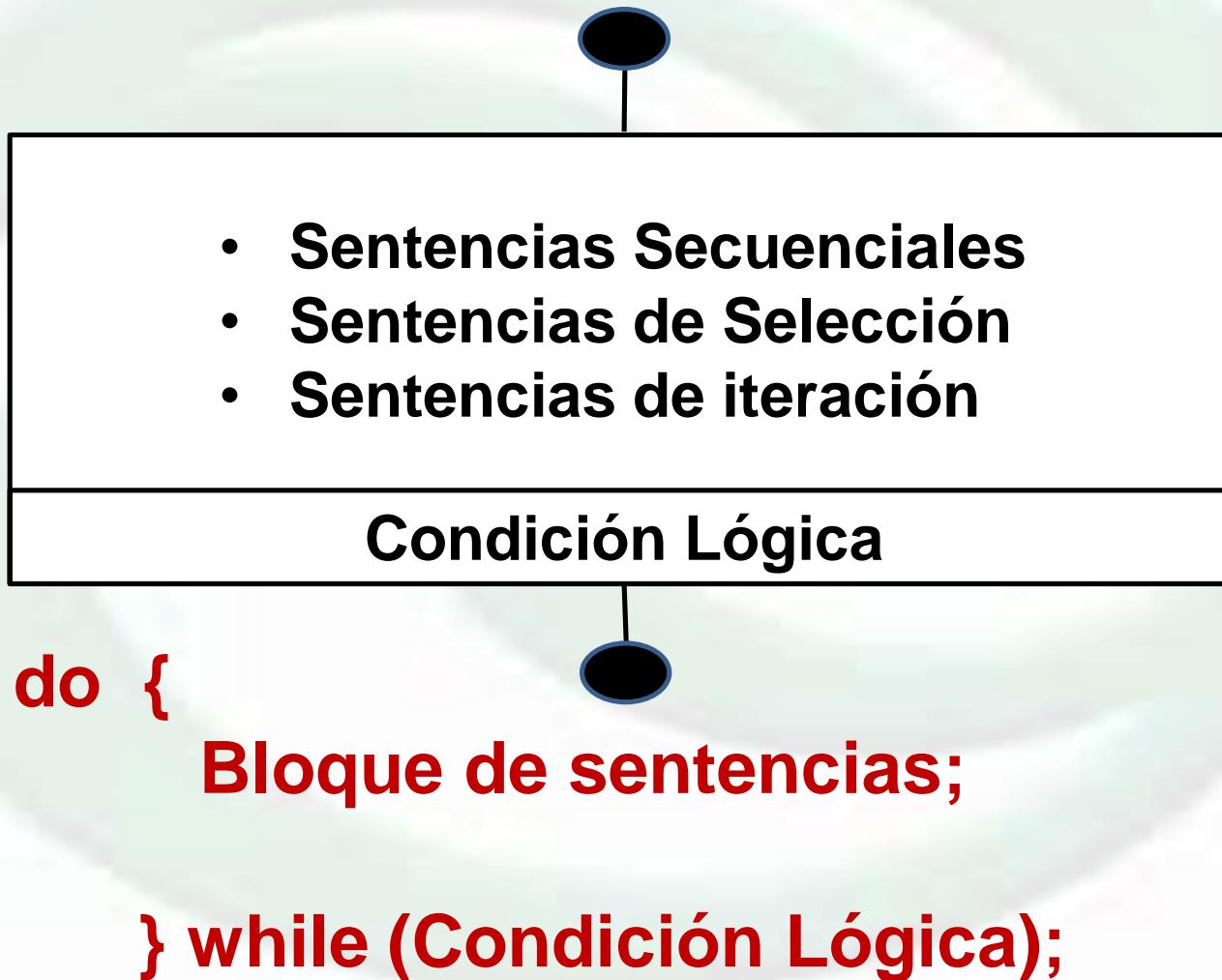
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones en programación y diseño de sistemas y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Estructura Do- while

### Máximos y Mínimos Bandera o señales

# Estructura Do - While

- Existe otra estructura de iteración condicionada, similar en gran medida al “while”. Su forma general es:



# Estructura do - while

- Significa hacer las sentencias UNA VEZ SIEMPRE y todas las otras veces que sean necesarias mientras la expresión arroje un resultado “verdadero”.
- Es útil cuando es necesario ejecutar una serie de sentencias por lo menos una vez, es un **ciclo (1 – n)**, a diferencia con el ciclo while que es **(0 - n)**.
- Una aplicación para el do-while puede ser para controlar el ingreso de los datos.
- Dentro del bucle, depende de la versión del C si se deben utilizar las {} .

# Estructura do - while

## Comparación entre **while** y **do-while**

- En ambas estructuras la repetición se produce cuando la condición es **verdadera**.
- En el **while** la condición se evalúa “**antes**” de ejecutar el bucle . Se repite de **0 a N veces**. Las variables de la condición deben ser inicializadas antes.
- En el **do/while** la condición se evalúa “ **después** ” de ejecutar una vez el cuerpo del bucle. Se repite de **1 a N veces**. Las variables que figuren en la expresión se pueden inicializar adentro.

# Ejemplo de do - while

Ingresar un número entero mayor o igual a cero. Informar el factorial de dicho número.

Factorial

fact= 1

“Ingrese un número entero mayor o igual a cero”

nro

nro < 0

J

1

nro

1

fact \*= J

“El factorial de ”, nro, “es ”, fact

# Ejemplo de do - while

```
#include <stdio.h>

int main ()
{ int fact, nro, J;

    printf ("\nIngrese un número entero mayor o igual a cero");

    do {
        scanf ("%d", &nro);
    } while (nro < 0);

    for (J = 1; J <= nro; J++)
        fact *= J;

    printf ("\nEl factorial de %d es %d", nro, fact);
    return 0;
}
```

# Máximos y Mínimos

Es un problema común querer encontrar el mayor o menor valor de una determinada serie de valores.

Se puede realizar de tres formas:

- a] Asignar a una variable el mayor (o menor) valor absoluto y empezar a comparar.
- b] Asignar a una variable el primer elemento ingresado como mayor (o menor) valor y a partir del segundo elemento comparar.
- c] Usar una señal para separar al primer elemento del resto del lote.

# Máximos y Mínimos

Adicionalmente, se puede necesitar informar la posición en la que ingresó el dato que resulta ser el máximo o mínimo elemento ingresado. Esto es, si dicho dato ingresó en primer lugar (posición 1), segundo lugar (posición 2), tercer lugar (posición 3), etc.

## Ejemplo

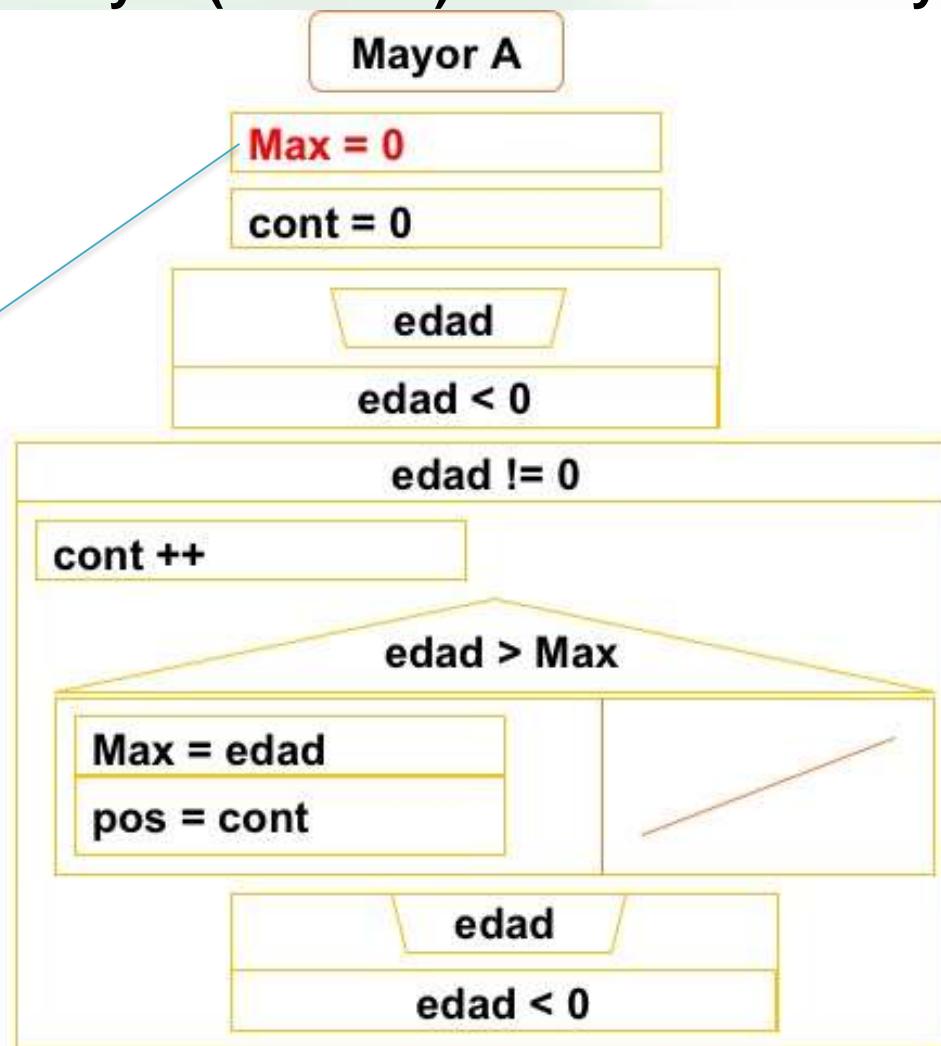
Ingresar las edades (entero, mayor que cero) de los alumnos de un curso e informar la mayor edad y en qué posición fue ingresada. La carga de datos finaliza con una edad igual a cero.

Nota: En caso de haber más de un máximo, informar cualquiera de ellos.

# Máximos y Mínimos

a] Asignar a una variable el mayor (o menor) valor absoluto y empezar a comparar.

Valor  
Absoluto

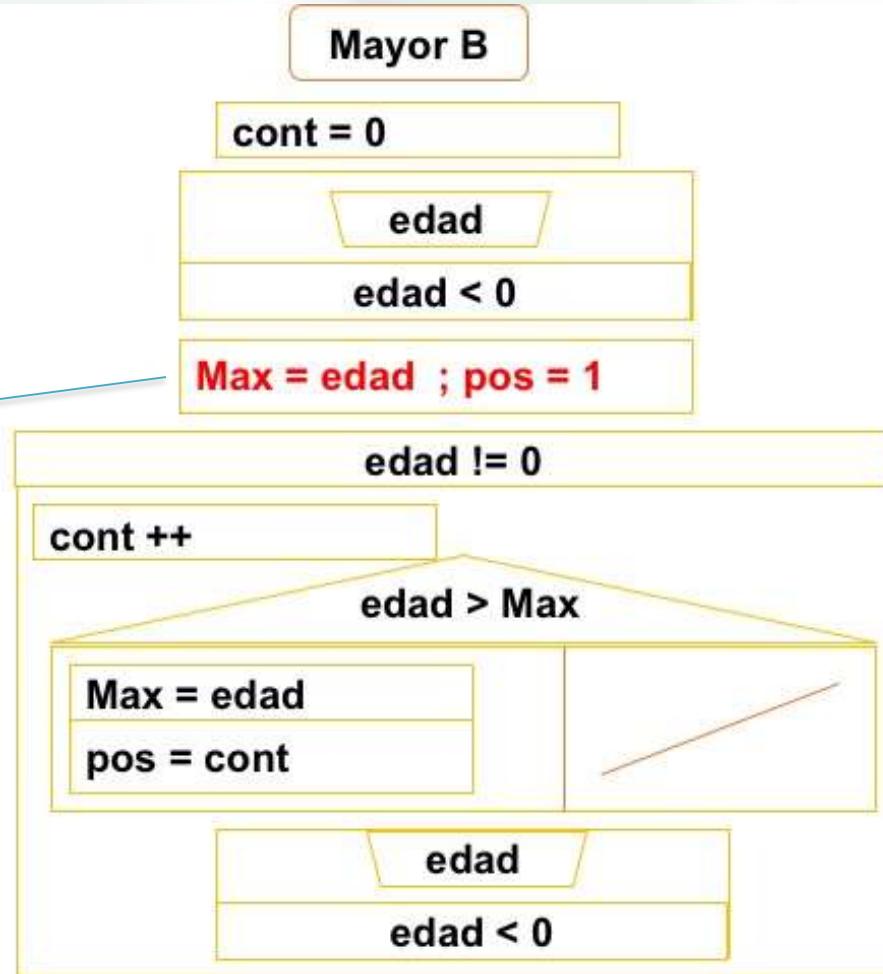


“La edad mayor es ”, Max, “y fue ingresada en la posición ”, pos

# Máximos y Mínimos

b] Asignar a una variable el primer elemento ingresado como mayor (o menor) valor y a partir del segundo elemento comparar.

Asignación del primer elemento como máximo



“La edad mayor es ”, Max, “y fue ingresada en la posición ”, pos

# Máximos y Mínimos

c] Usar una señal para separar al primer elemento del resto del lote.

Valor inicial de la variable tipo bandera

Cambio de valor de la variable tipo bandera

Mayor C

cont = 0

band = 0

edad

edad < 0

edad != 0

cont ++

band == 0 o edad > Max

Max = edad

pos = cont

band = 1

edad

edad < 0

“La edad mayor es ”, Max, “y fue ingresada en la posición ”, pos

# Banderas, señales o marcas (flags)

- En ciertos problemas es necesario retener el conocimiento de algún evento que ha transcurrido para luego tomar una decisión.
- Esto se puede lograr con una variable a la cual se le asigna solo dos valores, uno se asigna al inicio del proceso y el otro se asigna al suceder el evento buscado.
- En este caso la variable funciona como una señal, puede dejar pasar un número arbitrario de veces y luego al producirse la condición, cortar el flujo del proceso.

**Gracias por  
su atención  
¿ preguntas,  
dudas,  
murmuraciones  
o comentarios ?**



[www.imagenesplanet.com](http://www.imagenesplanet.com)

**Preguntas????**



# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones en programación, bases de datos y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>
```

## Funciones

# Ejercicio

## Número combinatorio:

Son números estudiados en combinatoria que corresponden al número de formas en que se puede extraer subconjuntos a partir de un conjunto dado.

Si llamamos al conjunto ***m*** y a los subconjuntos ***n***, con ***m*** y ***n*** mayores que cero y ***m*** mayor a ***n***, se tiene que el número combinatorio:

$$C_{m,n} = \frac{m!}{n! (m - n)!}$$

***m*** es el conjunto total de elementos

***n*** es el subconjunto

***C*** es la cantidad subconjuntos ***n*** que contiene ***m***

# Ejercicio

Por ejemplo, supongamos que tenemos 5 pares de zapatillas de distintos colores:



Azul



Rojo



Verde



Negro



Blanco

Estamos por irnos de vacaciones, pero en nuestra valija solamente hay lugar para llevar cuatro pares de zapatillas. Ahora bien, la pregunta es cuántas combinaciones de colores de zapatillas tomando de a cuatro pares ( $n$ ) de un total de 5 pares ( $m$ ).

Veamos:

# Ejercicio



1



2



3



4



5

# Ejercicio

Entonces, de un conjunto de 5 elementos (zapatillas), se pueden lograr 5 subconjuntos de 4 elementos.

Siendo  $m = 5$  y  $n = 4$ , el número combinatorio es:

$$C_{m,n} = \frac{m!}{n! (m-n)!}$$

$$C_{5,4} = \frac{5!}{4! (5-4)!} = \frac{120}{24 * 1} = 5$$

# Ejercicio

Ingresar dos números enteros  $m$  y  $n$ , mayores que cero y  $m$  mayor a  $n$ , e informar su número combinatorio.

**10 minutos para hacerlo!!!**

# Ejercicio – Versión 1

## Combinatorio

$m$

$m \leq 0$

$n$

$n \leq 0$

$m \leq n$

$fm = 1$

$J$

1 m 1

$fm * = J$

1

1

$fn = 1$

$J$

1

$n$

1

$fn * = J$

$J$

1

$m - n$

1

$fmn = 1$

$fmn * = J$

$C = fm / (fn * fm)$

“El numero combinatorio de ”,  $m$ , “y ”,  $n$ , “es: ”,  $C$



[VOLVER](#)

# Funciones

En general todos los lenguajes de programación disponen de una facilidad especial, que es la de poder escribir el programa como un conjunto de pequeños programas, llamados subprogramas.

Estos subprogramas funcionan como módulos independientes y si están debidamente enlazados por el programa principal resuelven eficientemente el problema planteado.

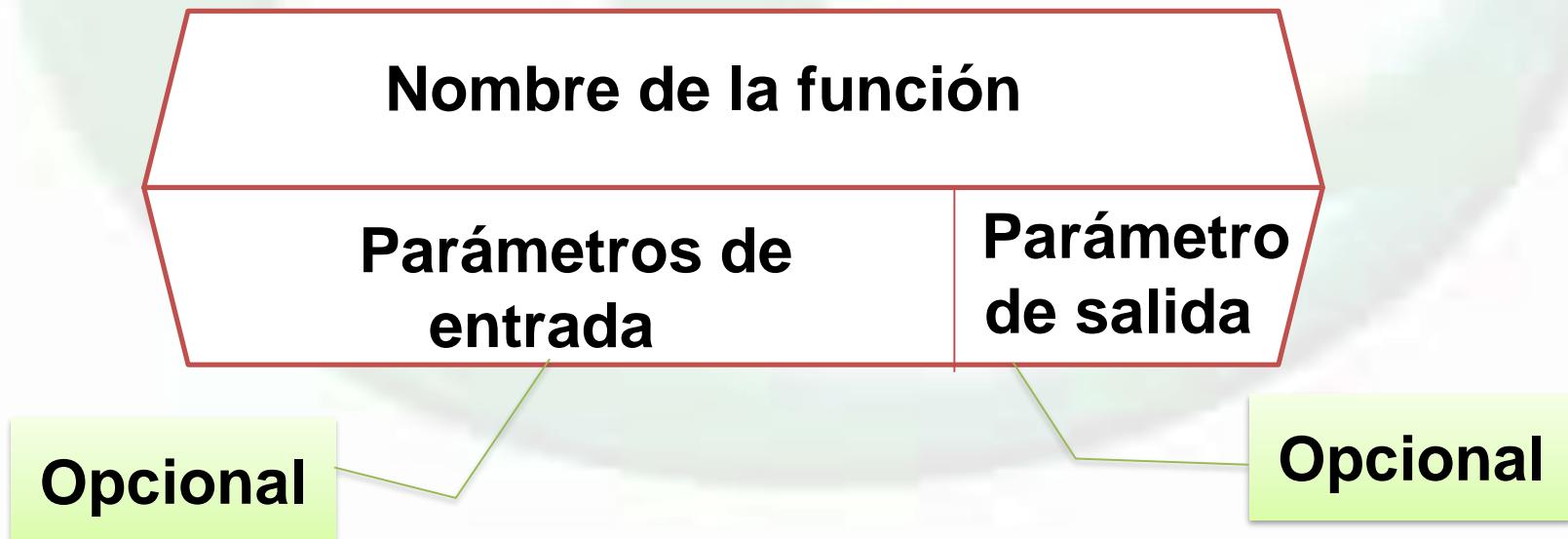
En el lenguaje C existe un solo tipo de subprogramas llamado **FUNCIÓN** y la función de programa principal la realiza una función llamada `main()`.

El programa principal simplemente ejecuta, pasa el control, a las funciones en el orden lógico estipulado. También se admite que una función , de manera análoga, llame a otra función.

# Funciones

De esta forma, el desarrollo de la estructura de un programa, puede ir paralelo al diseño descendente de la programación estructurada, o sea que podemos ir resolviendo el programa y definiendo simultáneamente las funciones que utilizaremos.

## Formato de la función en el diagrama



# Funciones

- **Facilita la programación porque divide el problema en subproblemas (Modularización).**
- **Simplifica la puesta a punto, ya que se puede corregir o cambiar una función sin tocar el resto del programa . Idem para los futuros cambios (es más caro el mantenimiento que hacerlo nuevo).**
- **Permite trabajar simultáneamente a varios programadores.**
- **Permite la reutilización de estas funciones sin volver a programarlas, solo las utilizo.**
- **Ahorra memoria, reduce el tamaño ocupado cuando uso la misma función varias veces.**

# Funciones

## Codificación en C

Existen tres lugares:

### 1. Declaración de la función o *Prototipo*

La declaración de la función se realiza inmediatamente después de la inclusión de las bibliotecas y de la declaración de constantes.

Formato general:

Tipo de dato de la variable de salida      Nombre de la función (tipo de dato de la/s variable/s de entrada);

Ejemplo:

```
int Factorial (int);
float Promedio (float, int);
void Mostrar (int);
int IngrDatoVal ();
```

# Funciones

## 2. Llamado o invocación a una función

Se realiza desde el main o desde una función (una función puede llamar a una o más funciones).

Formato general:

Variable donde se = Nombre de la función (lista de variable de guarda la salida);

opcional

opcional

Ejemplo:

```
fm = Factorial (m);
Prom = Promedio (suma, cont);
Mostrar (Prom);
m = IngrDatoVal ();
```

# Funciones

## 3. Definición de una función

Se realiza después de finalizar el main.

Formato general:

Tipo de dato de la variable de salida      Nombre de la función (lista de variable de tipo y *alias* de entrada)

Ejemplo:

```
int Factorial (int x)
float Promedio (float s, int c)
void Mostrar (int p)
int IngrDatos ()
```

[VOLVER](#)

# Ejercicio – Versión 2

## Combinatorio

m

$m \leq 0$

n

$n \leq 0$

$m \leq n$

$fm = \text{Factorial}(m)$

$fn = \text{Factorial}(n)$

$fmn = \text{Factorial}(m - n)$

$C = fm / (fn * fmn)$

“El numero combinatorio de”, m, “y”, n, “es:”, C

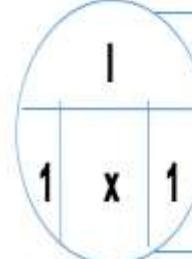
## Factorial

int x

int

$f = 1$

$f^*=1$



return f

# Ejercicio – Versión 3

## Combinatorio

`m = IngrDatosVal (0)`

`n = IngrDatosVal (0)`

`esDatosVal (m,n) == 0`

`fm = Factorial (m)`

`fn = Factorial (n)`

`fmn = Factorial (m - n)`

`C = fm / (fn * fmn)`

“El numero combinatorio de”, m, “y”, n, “es:”, C

## IngrDatosVal

`int lim`

`int`

`dato`

`esDatosVal (dato, lim) == 0`

`return dato`

## esDatosVal

`int x, int lim`

`int`

`X > lim`

`return 1`

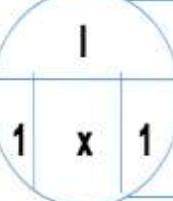
`return 0`

## Factorial

`int x`

`int`

`f=1`



`f*=1`

`return f`

# Ejercicio – Versión 4 (extremo)

## Combinatorio

**m = IngrDatoVal (0)**

**n = IngrDatoVal (0)**

**esDatoVal (m,n) == 0**

“El numero combinatorio de”, m, “y”, n, “es:”,  
Factorial (m) / (Factorial (n) \* Factorial (m - n))

## IngrDatoVal

**int lim**

**int**

**dato**

**esDatoVal (dato, lim) == 0**

**return dato**

## esDatoVal

**int x, int lim**

**int**

**X > lim**

**return 1**

**return 0**

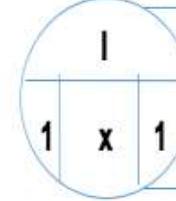


## Factorial

**int x**

**int**

**f=1**



**f\*=1**

**return f**

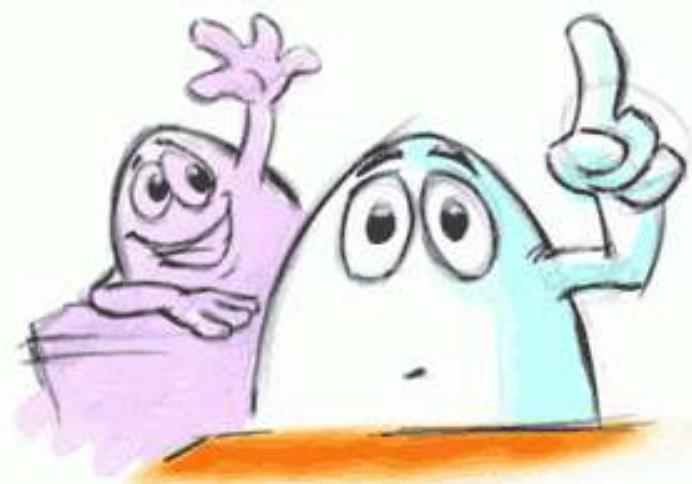


Si tienes dudas...  
¡¡¡Pregunta!!!



[MenudosPeques.net](http://MenudosPeques.net)

# Preguntas????



# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f90..."/>
<meta http-equiv="Content-Type" content="text/html; charset=..."/>
<title>Gesatecsl. Soluciones de software para empresas y grafic...
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Cálculos con fechas

# Ejercicios

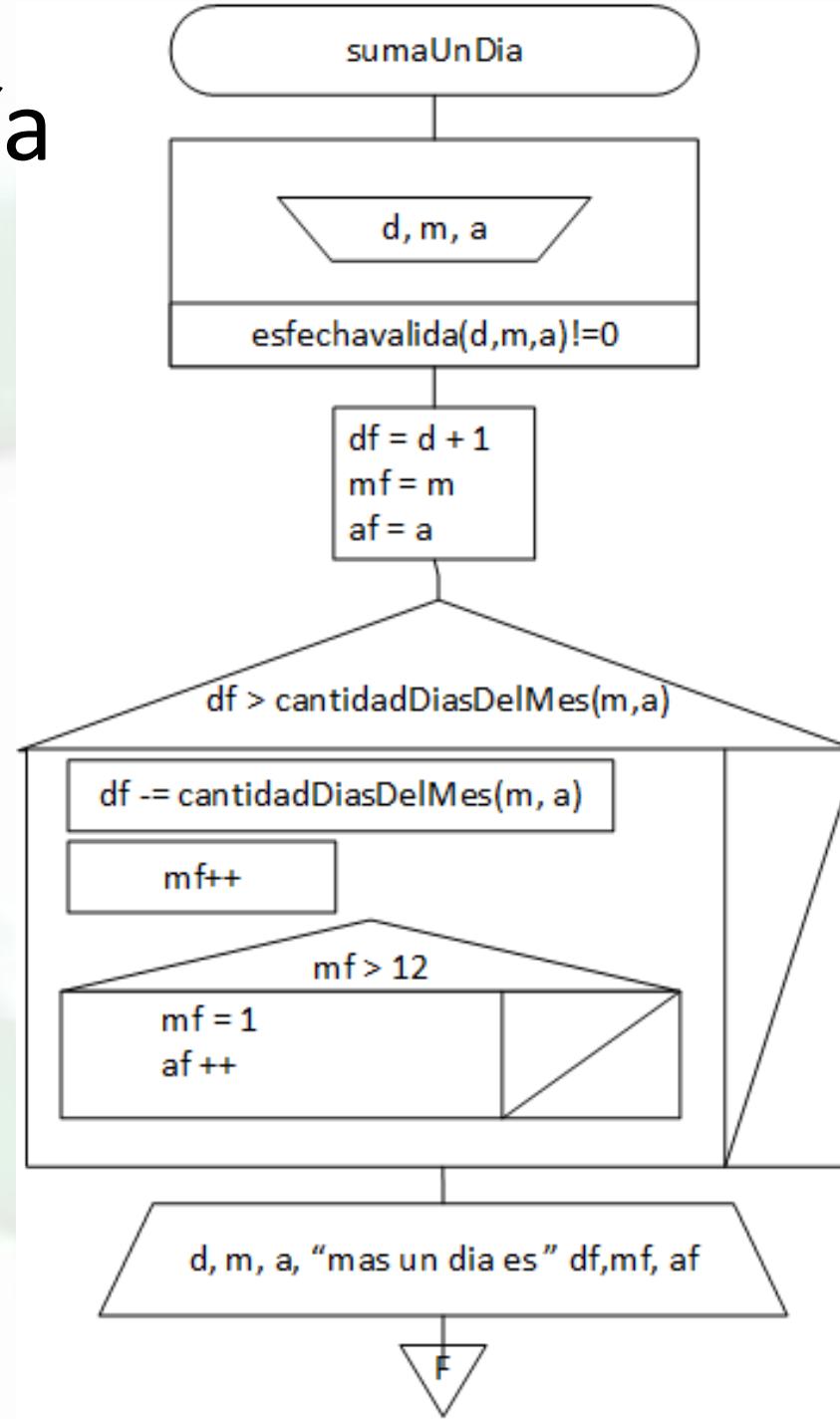
1. Sume un día a la fecha dada.
2. Reste un día a la fecha dada.
3. Sume n días a la fecha dada.
4. Reste n días a la fecha dada.



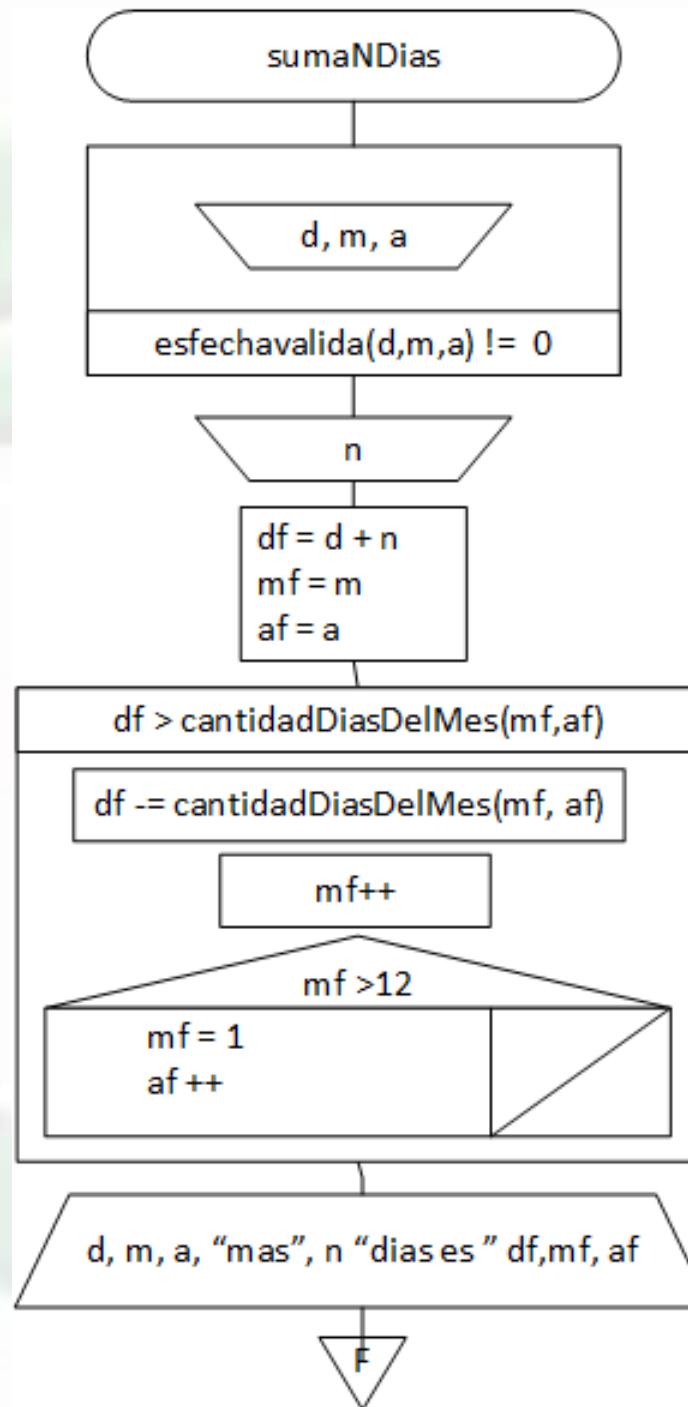
## ¿Analizamos el problema?

- ¿Qué pasa si me paso los límites de días del mes?
- Si cambia el mes ¿Se pasa de los límites de meses del año?
- Si cambia el año ¿El año está dentro de límites establecidos? o...
- Si cambia el año ¿Será éste bisiesto afectando los datos antes mencionados?

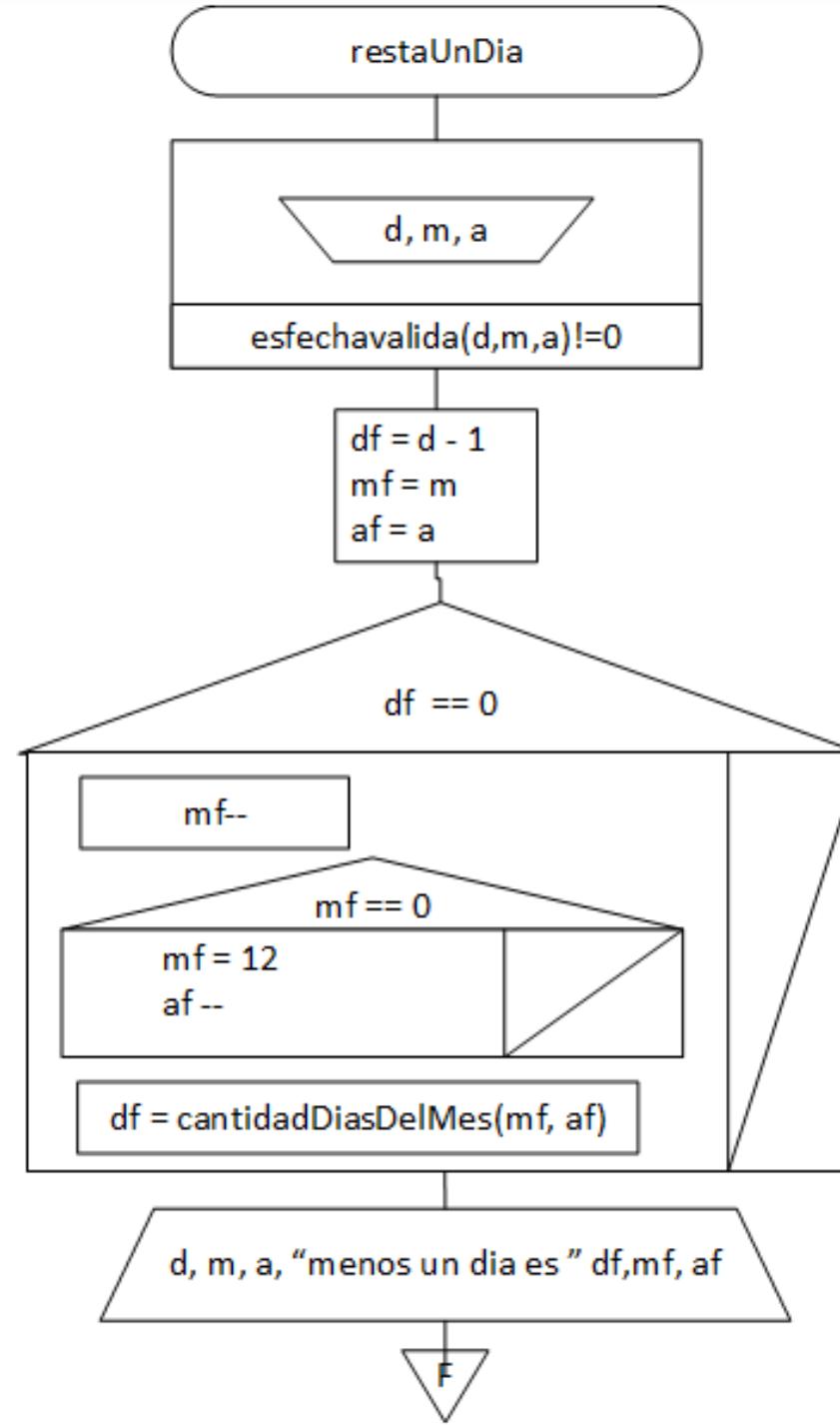
# Sume un día a la fecha dada



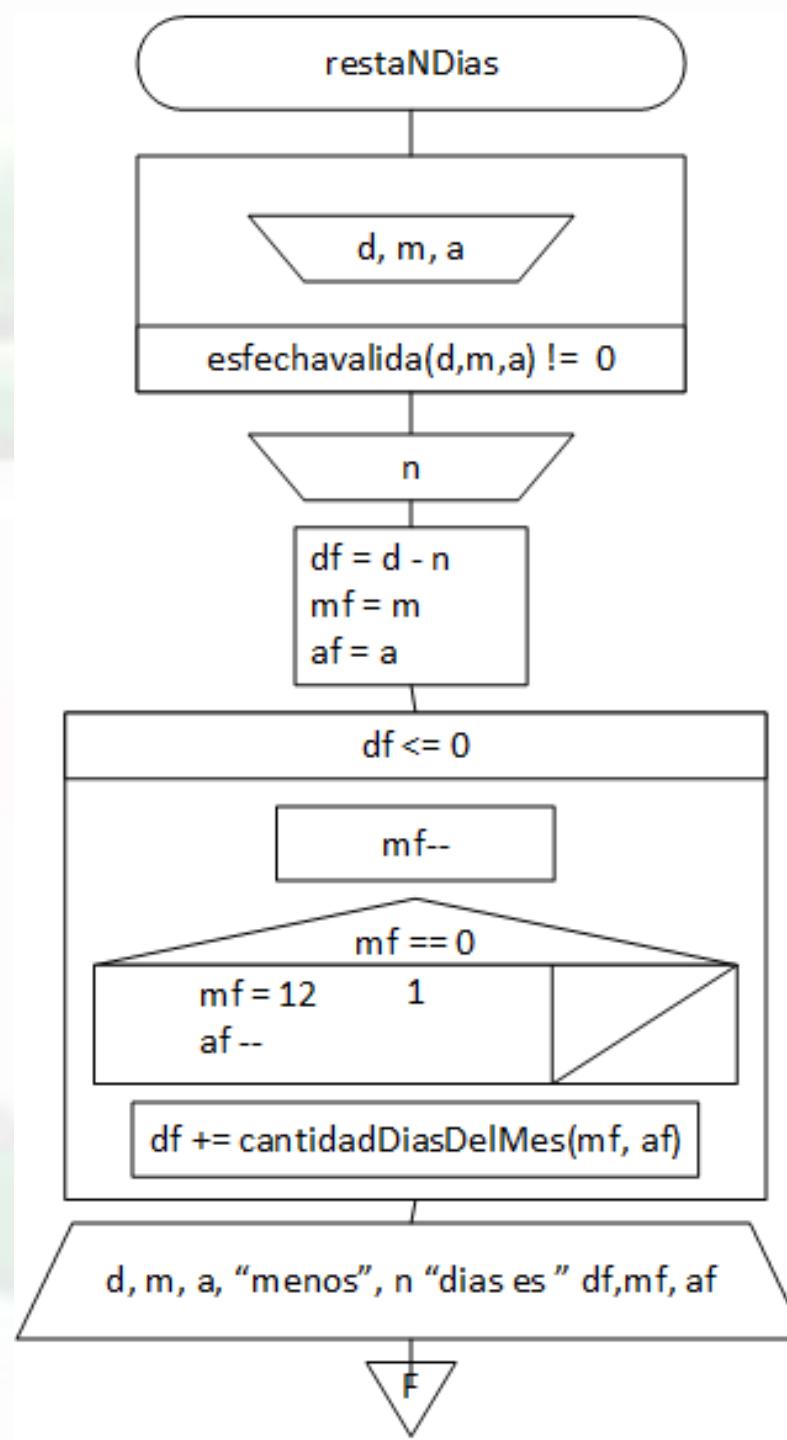
# Sume n días a la fecha dada



# Reste un día a la fecha dada



# Reste n días a la fecha dada



# ¿CONSULTAS



MUCHAS GRACIAS  
POR SU ATENCIÓN.

MANOS A LA OBRA y...

SUERTE



# Elementos de Programación

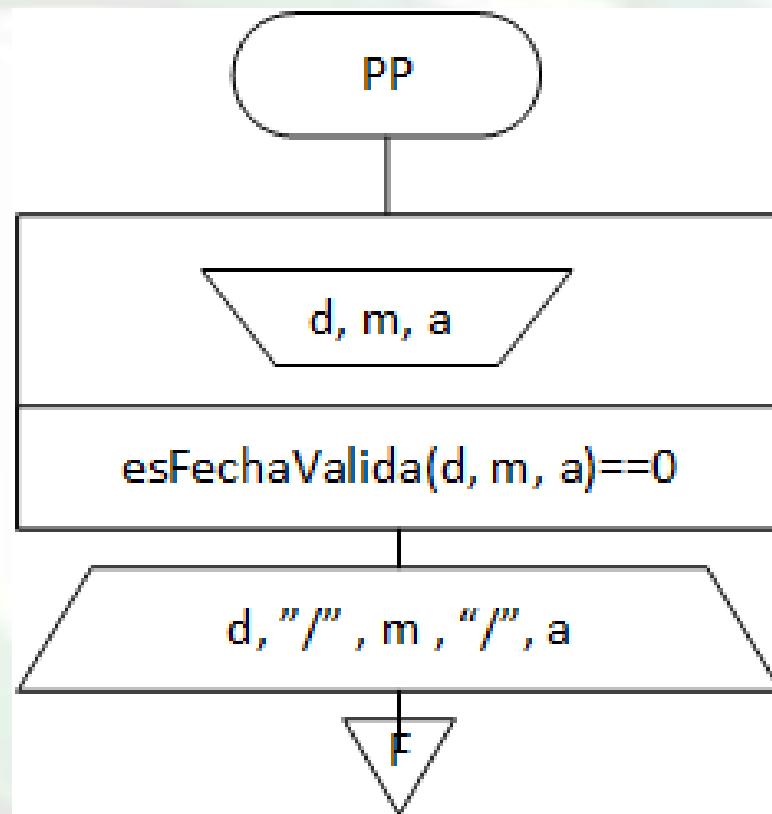


```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f90..."/>
<meta http-equiv="Content-Type" content="text/html; charset=..."/>
<title>Gesatecsl. Soluciones de software para empresas y grafic...
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Validando fechas

# Ejercicio

- Ingresar e imprimir una fecha válida.

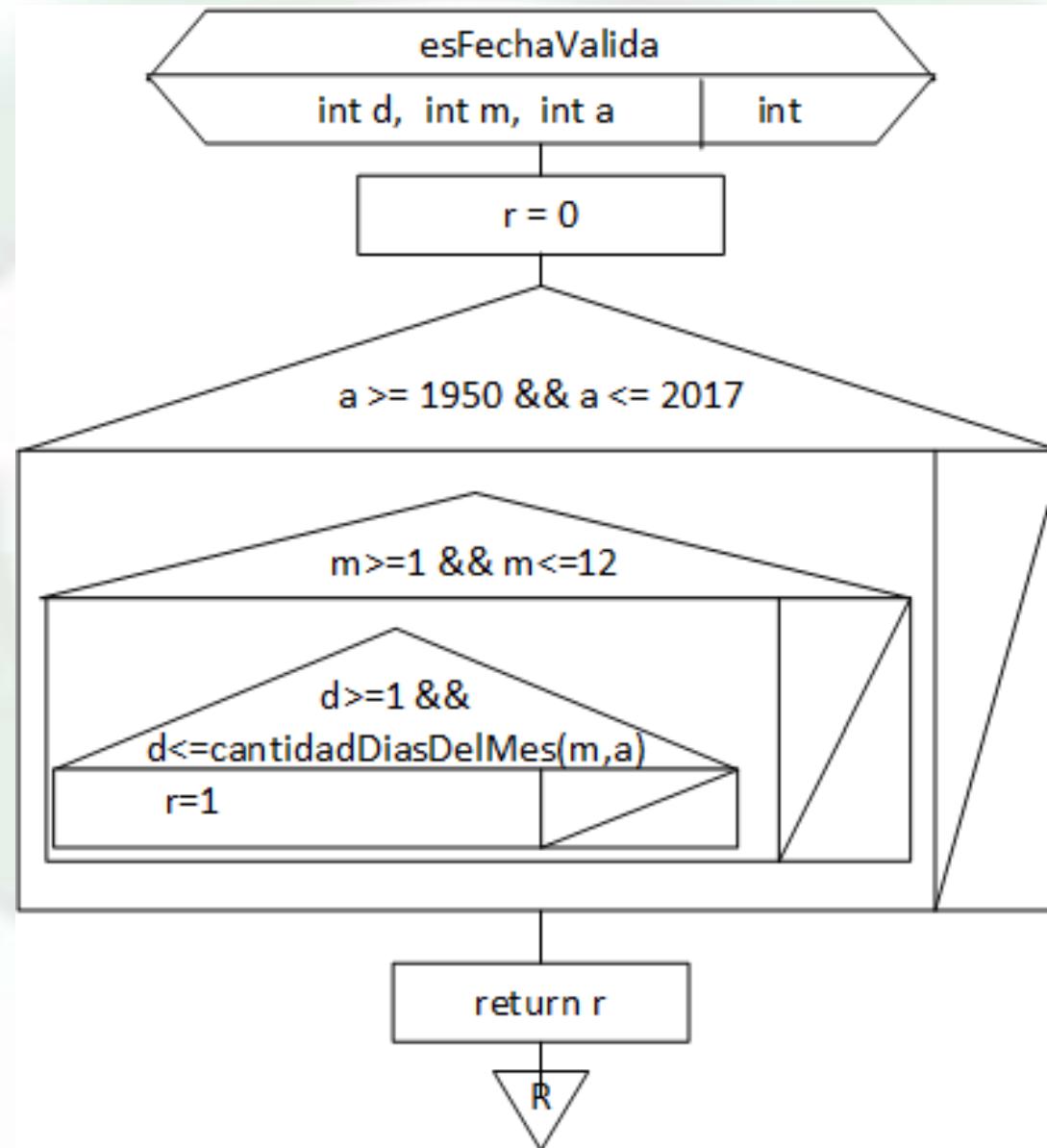


# Funciones necesarias?

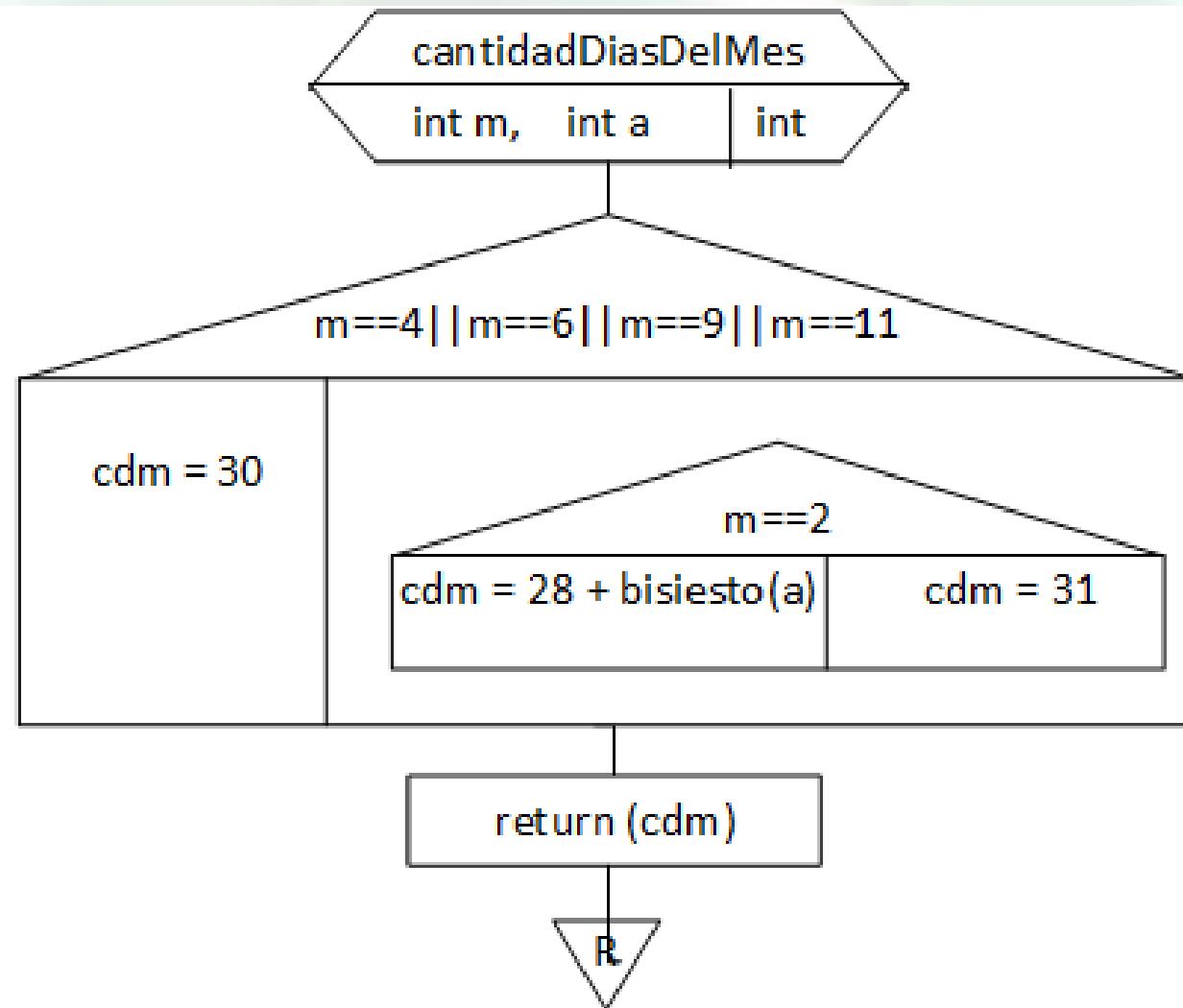


- ❖ ¿Es fecha válida?
  - ¿Es correcto el día?
    - ✓ ¿Cuántos días tiene ese mes?
      - ¿El año es bisiesto?

# Función esFechaValida()



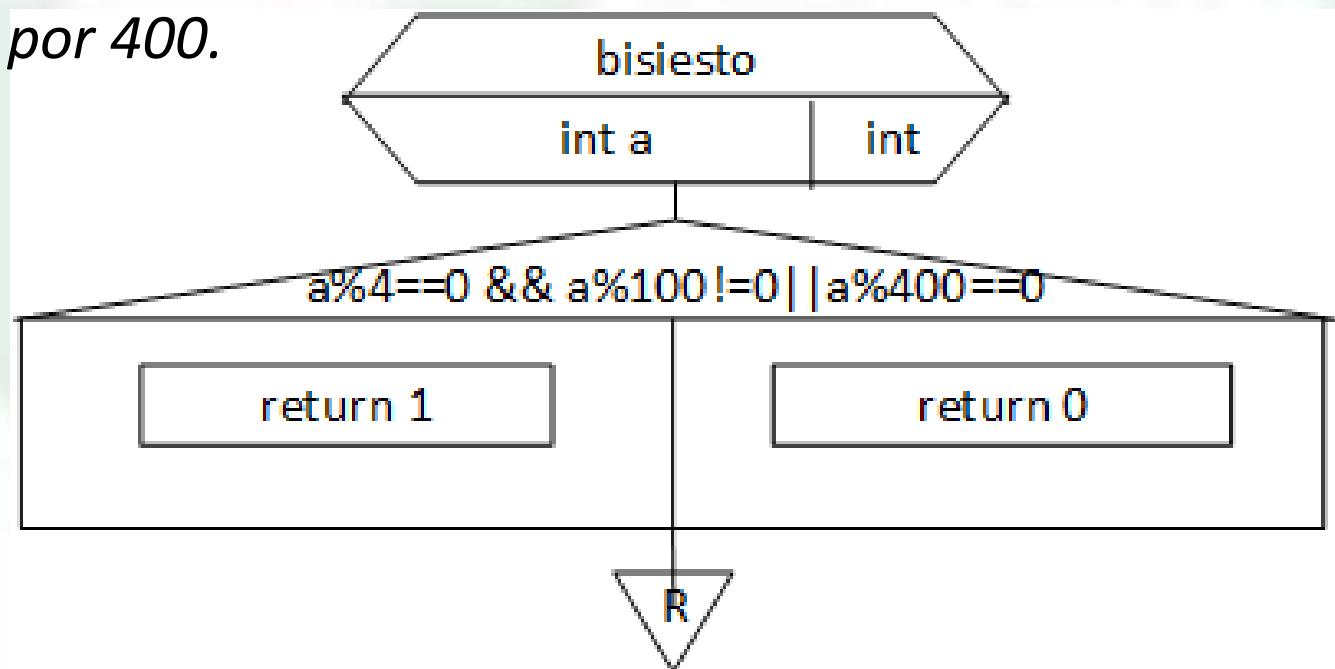
# Función cantidadDiasDelMes()

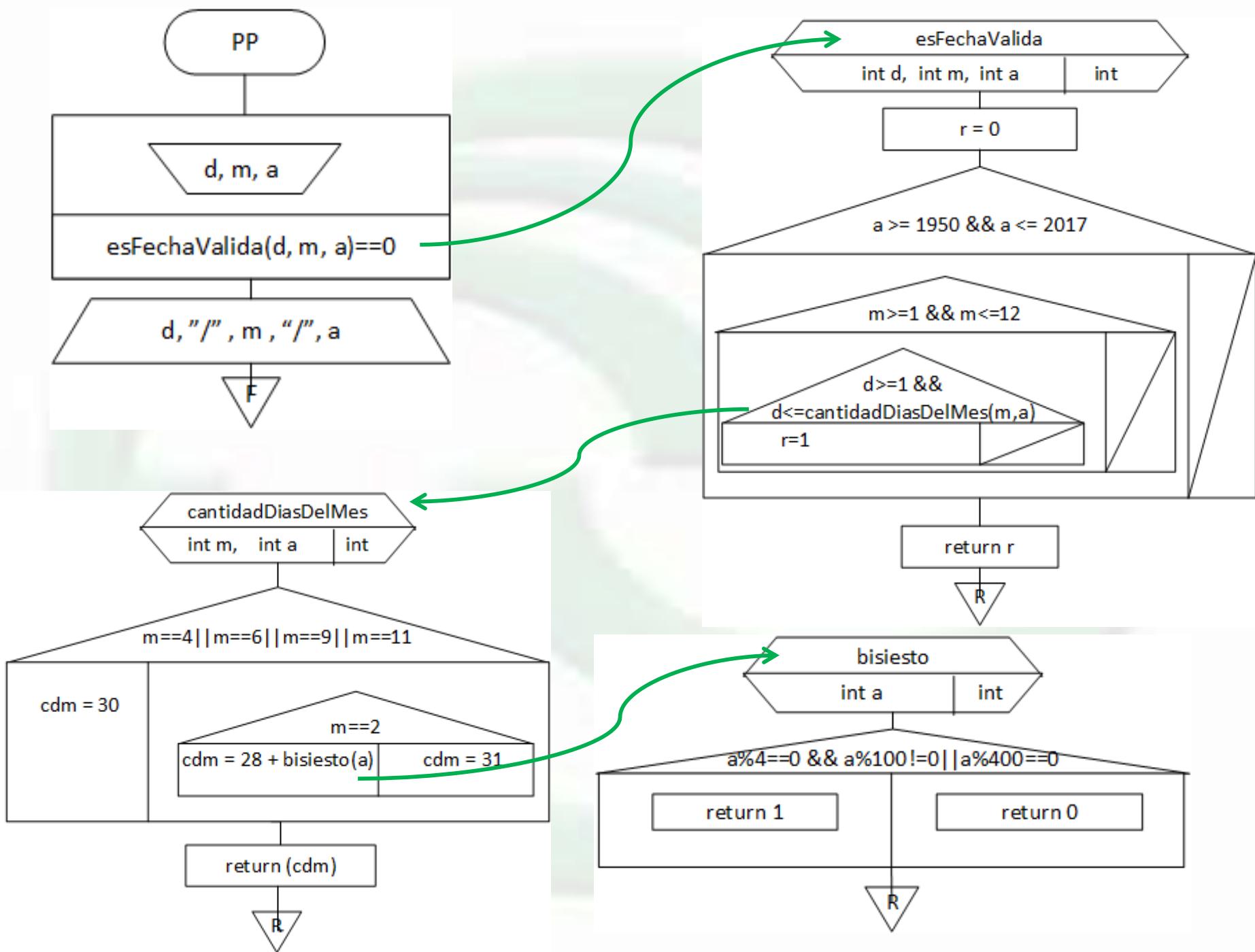


# Función bisiesto()

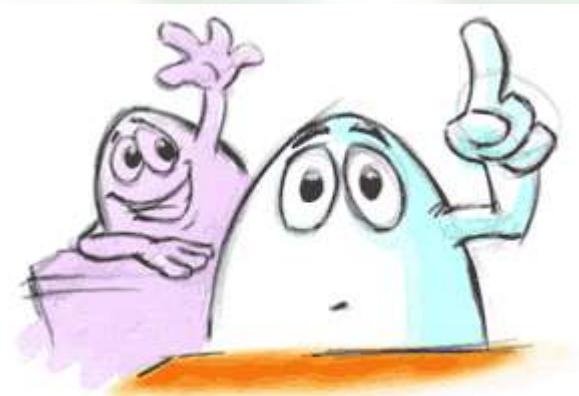
**Condiciones del año sabiendo que el mismo dura 365 días 6 horas 9 minutos 9,76 segundos:**

1. *Divisible por 4 y*
2. *No ser divisible por 100.*
3. *Si fuese divisible por 100 también debe ser divisible por 400.*





# ¿CONSULTAS?



# Responder justificando

- En la práctica, ¿es valido que las variables “d”, “m” y “a” se nombren igual en el PP y 3 funciones distintas al mismo tiempo?
- Podemos reemplazar la condición **esFechaValida(d, m, a)==0** por **esFechaValida(d, m, a)!=1** o por **!esFechaValida(d, m, a)**

# Ejercicios a resolver

- Dada una fecha válida confeccionar funciones independientes que:
  1. Sume un día a la fecha dada.
  2. Reste un día a la fecha dada.
  3. Sume n días a la fecha dada.
  4. Reste n días a la fecha dada.

MUCHAS GRACIAS  
POR SU ATENCIÓN.

MANOS A LA OBRA y...

SUERTE



# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f91...
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<title>Gesatecsal. Soluciones para la informática y gráficas y...
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>

```



# Vectores, arreglos o arrays

# Ejercicio

Ingresar las edades de los alumnos (entero, mayor que 15) de un curso de no más de 100 alumnos, aunque no se sabe cuántos alumnos presentes hay. La edades se ingresan de a una por vez, y finaliza la carga con una edad igual a cero.

Informar:

- a) El promedio de edades ingresadas.

*Como esto ya fue hecho....*

**5 minutos para hacerlo!!!**

# Ejercicio

Edades

suma = 0

cont = 0

ed = Ingrdatoval2 (15, 0)

ed != 0

suma = suma + ed

cont ++

ed = Ingrdatoval2 (15, 0)

cont > 0

“El promedio  
de edades es: ”,  
suma / cont

“No hay alumnos”

Ingrdatoval2

int lim1, int lim2

int

dato

esdatoval (dato, lim1) == 0 y dato != lim2

return dato



esdatoval

int x, int lim

int

X > lim

return 1

return 0



# Ejercicio

*Pero nada es perfecto, así que ....*

b) Informar cuántos alumnos superan la edad promedio.

**5 minutos para hacerlo!!!**

# Vectores, arreglos o arrays

- Cuando se necesita guardar información en la memoria del computador se utilizan variables, a las que se asigna un nombre y tienen un tipo de datos asociado. Por lo tanto un dato guardado en una variable puede ser simplemente el carácter “A” o un valor numérico entero o real. En todos los casos, cada variable representa a un único dato individual.
- Esta estructura es útil para todos los problemas en los que la cantidad de variables utilizadas es limitada. Pero cuando la cantidad de datos excede un número razonable es conveniente usar estructuras de datos que representen muchos elementos de información.
- Para ello existen los tipos de datos estructurados, donde un único *identificador* puede representar a *múltiples datos individuales*, pudiendo cada uno de ellos ser referenciado por separado.

# Vectores, arreglos o arrays

Los **vectores** o **arreglos** (array en inglés) son una estructura compuesta por un número de **elementos homogéneos**, agrupados bajo un **único nombre**, diferenciándose cada elemento por su **posición**.

## Características

1. Tienen un único nombre de variable que representa a todos los elementos, y estos se diferencian por su índice o subíndice.
2. Todos los elementos deben ser del mismo tipo.
3. Se puede acceder en forma directa o indirecta a cada elemento individual del arreglo.
4. Los elementos del arreglo se almacenan en posiciones contiguas de memoria.
5. Son estáticos, es decir, se crean en “**tiempo de ejecución**”.

# Vectores, arreglos o arrays

Los **arreglos** pueden ser **unidimensionales (vectores)** o **bidimensionales (matrices)**.

## Vectores

VEjemplo



0 1 2 3 4 .....



n-4 n-3 n-2 n-1

$n$  elementos

Posición o índice del vector

Contenido o elemento del vector

# Vectores, arreglos o arrays

## Declaración de un vector

Al declarar un arreglo, el compilador reserva un bloque de memoria contigua lo suficientemente grande como para guardar el arreglo completo, pero se debe ser muy cuidadoso con la dimensión, porque el lenguaje “C” no comprueba límites. Por lo tanto se puede seguir ingresando valores en forma peligrosa, superando el tamaño y dañando incluso el sistema operativo. El “C”, no emitirá mensaje alguno alertando esta situación.

### Formato general

Tipo de dato              Nombre del vector [tamaño del vector];  
de los elementos

```
int VEdades[100];
float Vimpp[500];
```

# Vectores, arreglos o arrays

## Uso de índice de un vector

- Cada referencia a un arreglo unidimensional incluye el nombre y el índice encerrado entre corchetes.
- El índice determina que elemento se procesa.
- A los componentes de un arreglo, se les puede asignar valores de igual forma que a cualquier otra variable, con la condición que sean del tipo correspondiente.
- Ejemplo: a [7]= 25; //Asigna el valor entero 25 a la posición 8 del vector “a”.  
b [3]= 4.4; //Asigna un valor de tipo float en la cuarta posición del vector “b”.
- Todos los elementos del vector “a” deben ser enteros. Un vector puede ser de tipo real, como el vector “b”, pero no así su índice.

# Vectores, arreglos o arrays

- El índice puede ser el valor de una expresión.
- Por ejemplo:

**Si i = 8;      e[10\*i -1]= 40; /\*Asigna 40 a e[10\*8-1] es decir a e[79]\*/**

- También el subíndice puede ser el contenido de un elemento de un arreglo de enteros.
- Ejemplo:

**vector[otrovector[3]]= 200;**

Para hacer referencia a un elemento de un arreglo se puede utilizar como subíndice, una constante, una variable o cualquier expresión de tipo entero.

# Vectores, arreglos o arrays

## *Lectura y escritura de un vector*

- Los arreglos no se pueden leer/escribir en una sola operación o sentencia.
- La lectura y escritura de un arreglo se debe realizar dentro de **estructuras iterativas** leyendo o escribiendo elemento a elemento.

# Vectores, arreglos o arrays

## ***Vectores como parámetros de funciones***

- Al transferir **por valor**, se copian los parámetros actuales en sus correspondientes parámetros formales.
- Dicha operación se realiza automáticamente cuando se llama a la función, no modificando el valor de los argumentos.
- Con esta modalidad podemos transferir constantes, variables y expresiones.
- En los pasajes por referencia, no se transfieren los valores, sino las direcciones de las variables, que contienen esos valores.

# Vectores, arreglos o arrays

- Los parámetros actuales pueden modificarse.
- Con este método, podremos pasar arreglos, funciones y direcciones de variables.
- El lenguaje “C” al transferir un vector a una función, interpreta el nombre del arreglo como *la dirección de la posición de memoria que contiene al primer elemento del vector*.
- Esta dirección se asigna al correspondiente parámetro formal cuando se llama a la función.
- Por lo tanto el argumento formal se convierte en un puntero al primer elemento del arreglo.
- Con esta modalidad de transferencia, se debe ser muy cuidadoso, dado que si un elemento del vector es modificado dentro de la función, el cambio será reconocido en el programa que lo llamó.

# Ejercicio – Versión 2

Edades\_v2

**cant = CargaVec (VEdades, 100)**

**cant > 0**

**sum = SumaVec (VEdades, cant)**

**prom = sum / cant**

**“El promedio de edades es: \_\_\_\_\_, prom”**

**“No hay alumnos”**

**MayProm = ContarMayorA (VEdades, cant, prom)**

**“La cantidad de alumnos que supera la edad promedio es: ”, MayProm**

# Ejercicio – Funciones

CargaVec

int V[], int ce

int

i = 0

ed = IngrdatoVal2 (15, 0)

ed != 0 y i < ce

V[i] = ed

i ++

ed = IngrdatoVal2 (15, 0)

return i

IngrdatoVal

int lim

int

dato

esdatoVal (dato, lim) == 0

return dato



esdatoVal

int x, int lim

int

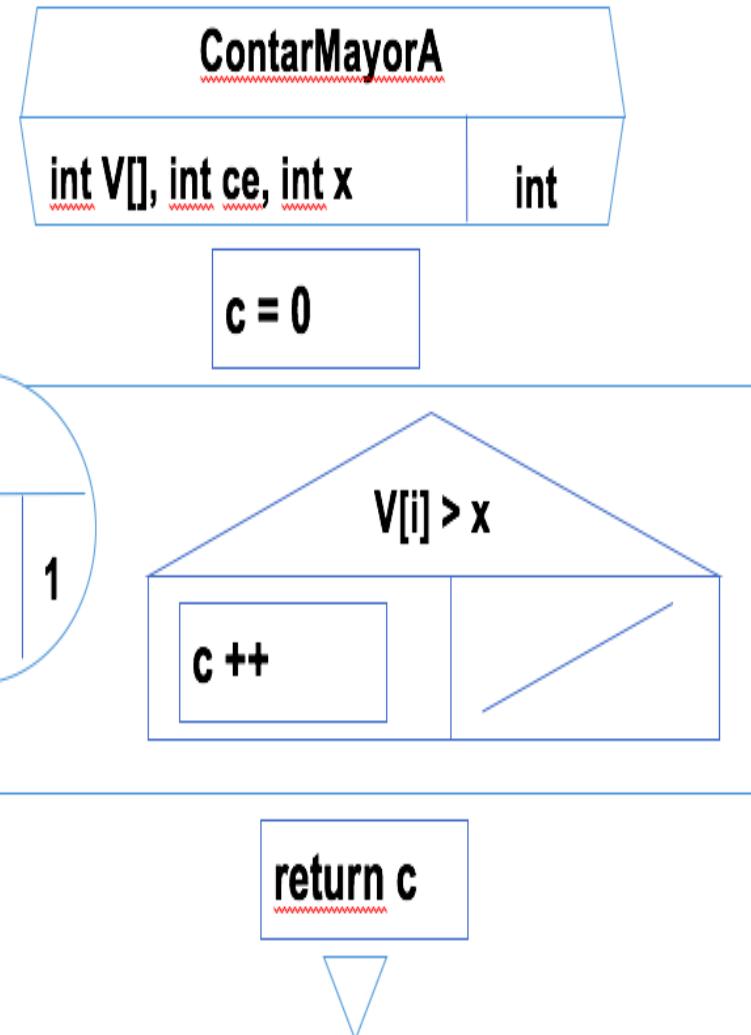
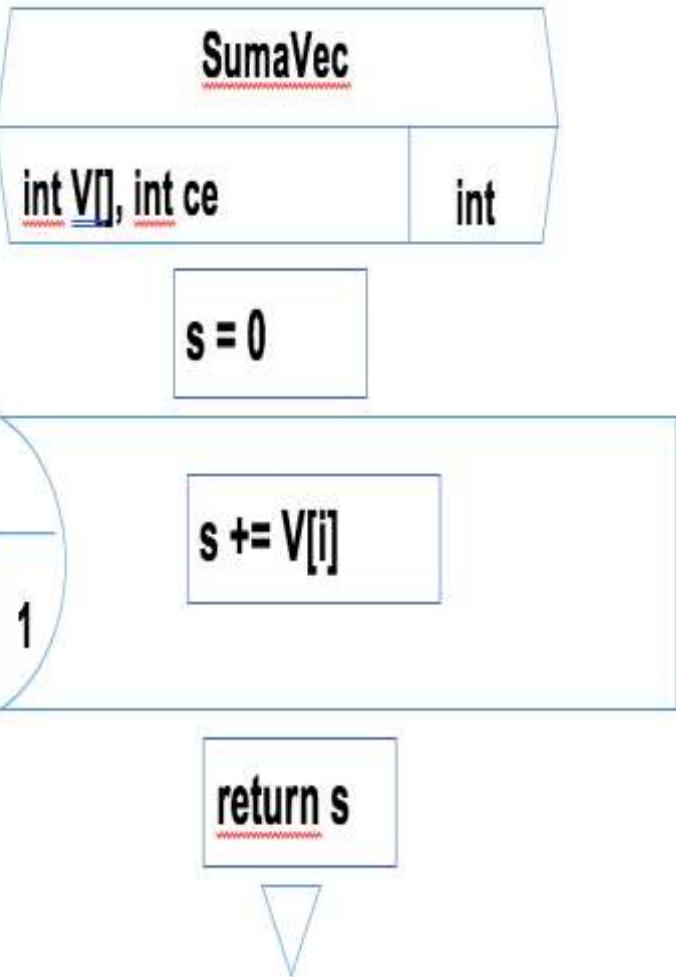
X > lim

return 1

return 0



# Ejercicio – Funciones





Preguntas????



## Ejercicio – Versión 3

**Modificar el ejercicio para ingresar por cada alumno dos datos:  
DNI y edad. El fin de carga ahora es DNI = 0.  
Informar lo pedido anteriormente mas el siguiente punto:**

**c) Informar el DNI de todos los alumnos que tengan una edad menor o igual al promedio de las edades.**

**A trabajar!!!!**



# Elementos de Programación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t
<meta http-equiv="Content-Type" content="text/html;
<title>Gesatecsl. Soluciones para la Gestión de Proyectos y la Información
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Operaciones con Vectores: Búsqueda

# Operaciones con Vectores: Búsqueda de un elemento en un vector

Un problema importante en el procesamiento de datos, es la búsqueda en un conjunto de datos de un elemento específico, y la recuperación de información asociada al mismo.

Existen diferentes métodos de búsqueda.

- Búsqueda Secuencial o lineal
- Búsqueda binaria o dicotómica

# Operaciones con Vectores: Búsqueda Secuencial

Es la más simple de las búsquedas, fácil de codificar y eficiente.

Consiste en recorrer todo el vector, desde el primer elemento hasta el último o hasta encontrar el elemento, y de uno en uno.

En forma general, se maneja como una **función**.

Si desde el programa principal, se desea conocer si existe el elemento, será suficiente con preguntar cuál es el valor que retorna la función. Si es -1 quiere decir que el elemento no existe, en caso contrario su valor es la posición en el vector.

# Operaciones con Vectores: Búsqueda Secuencial

**Ejemplo:**

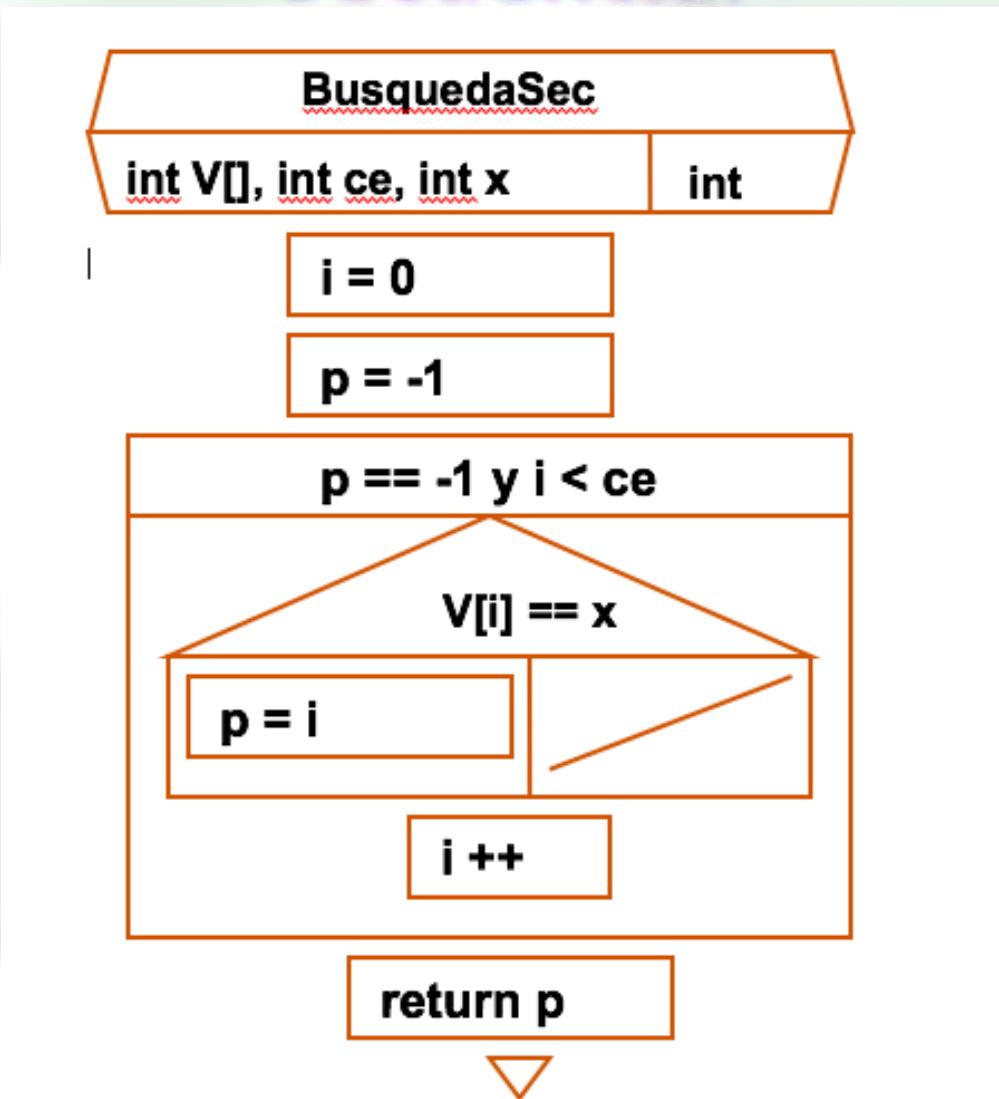
Supongamos tenemos un vector de 15 posiciones con valores enteros. Queremos saber si un determinado valor  $x$  está dentro del vector.

12	-1	112	76	-9	14	50	55	0	32	27	10	14	71	88
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



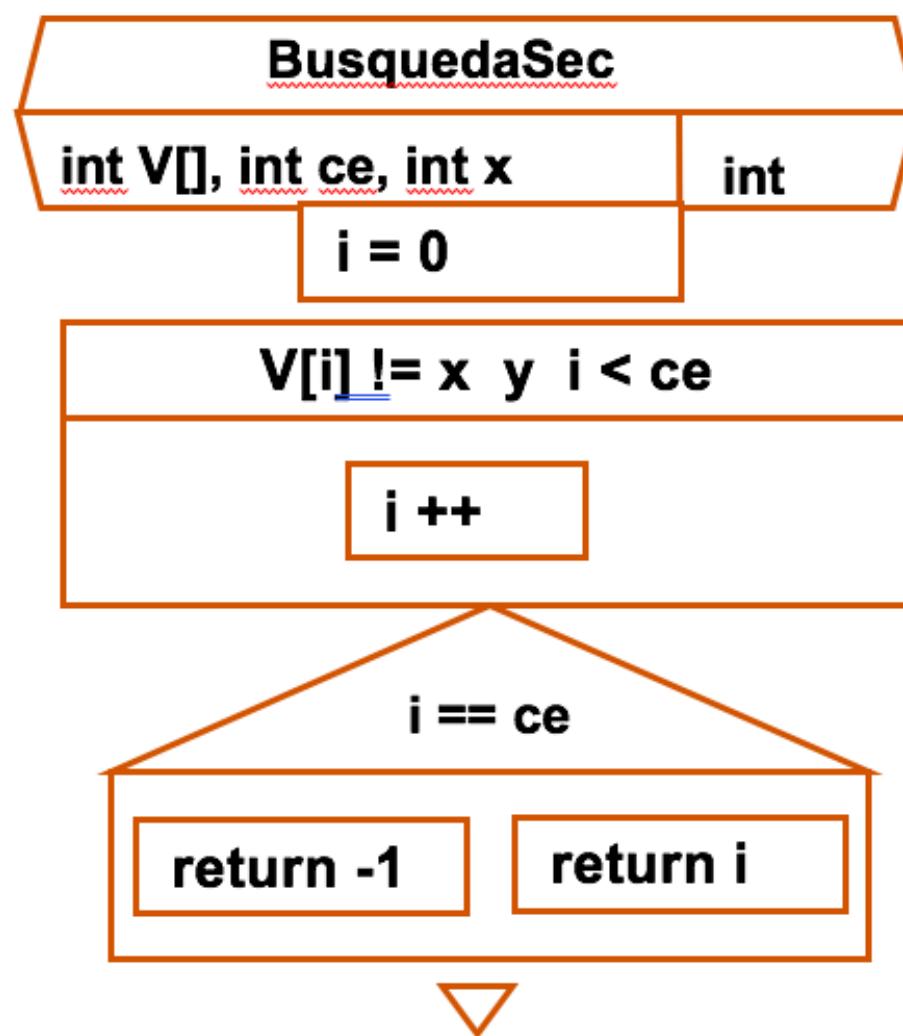
$x$

# Operaciones con Vectores: Búsqueda Secuencial



# Operaciones con Vectores: Búsqueda Secuencial

**Otra versión:**





# Preguntas????



ESTAMOS  
PARA ACLARARTE  
CUALQUIER DUDA!!

NO DUDES EN PREGUNTAR!!

# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones en programación, bases de datos y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Vectores: Acceso directo

# Vectores: Acceso Directo

En muchos ejercicios el acceso a un vector no se realiza en forma secuencial, ya sea para insertar un elemento o para leerlo. Esta situación sucede especialmente cuando el índice del vector coincide con alguno de los datos del problema que estamos tratando de resolver a través de un algoritmo.

Sin embargo, en estos casos hay que tener en cuenta que en el lenguaje C, **los vectores comienzan desde la posición cero.**

Esto hace que siempre estemos desfasados en una unidad, que se suma o se resta, dependiendo de la acción que estemos realizando sobre el vector.

# Vectores: Acceso Directo

Por ejemplo:

Un negocio de electrodomésticos tiene 8 vendedores que están identificados del 1 al 8. Cada día, ante cada venta, se ingresa el número de vendedor y el importe de la venta. Al finalizar la jornada, se ingresa un vendedor igual a cero.

Informar qué vendedor vendió más en el día y qué vendedor vendió menos.

Nota 1: En caso de haber más de un vendedor que vendió la máxima venta, informarlos a todos.

Nota 2: En caso de haber más de un vendedor que vendió la mínima venta, informar el primero que aparezca.

# Vectores: Acceso Directo

## *Análisis:*

Son ocho vendedores, aunque no se saben si están los ocho. Un mismo vendedor puede vender varias veces en el mismo día. La condición de fin de carga de ventas es un vendedor igual a cero.

**10 minutos para plantearlo**

# Vectores: Acceso Directo

Se puede usar un vector de ocho posiciones

VVend

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

0      1      2      3      4      5      6      7

Vendedores (menos uno)

*Cada posición acumula  
las ventas de ese  
vendedor*

10 minutos para resolverlo!!

# Vectores: Acceso Directo

Vendedores

PonerEnCero (VVend, 8)

CargarVec (VVend, 8)

Max = MaximoVec (VVend, 8)

MostrarIgualQue (VVend, 8, Max)

posmin = PosicionMinVec (VVend, 8)

“El vendedor que menos vendió en el día es”,  
posmin + 1

PonerEnCero

float V[], int ce

$V[i] = 0$

i		
0	ce-1	1

CargarVec

float V[], int ce

suc = IngrdatoVal (0, 8)

suc != 0

imp

imp <= 0

$V[suc - 1] += imp$

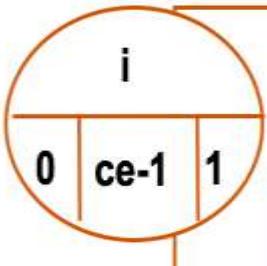
suc = IngrdatoVal (0, 8)

# Vectores: Acceso Directo

**MaximoVec**

float V[], int ce

float



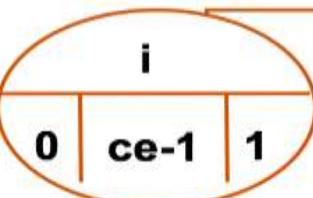
$i == 0 \text{ o } V[i] > M$

$M = V[i]$

return M

**MostrarIgualQue**

float V[], int ce, float x



$V[i] == x$

$i + 1$

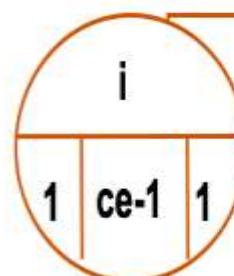
**PosicionMinVec**

float V[], int ce

int

$m = V[0]$

$pm = 0$



$V[i] < m$

$m = V[i]$

$pm = i$

return pm



# Preguntas????



ESTAMOS  
PARA ACLARARTE  
CUALQUIER DUDA!!

NO DUDES EN PREGUNTAR!!

# Vectores: Acceso Directo

*Entonces...:*

Modificar el ejercicio resuelto para que muestre el promedio de ventas de cada vendedor.

**5 minutos para plantearlo**

# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f91..."/>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"/>
<title>Gesatecsl. Soluciones de software para empresas y graficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Vectores Ordenamiento Selección/Burbujeo

# TIPOS Y METODOS

- Métodos externos (se ordena en memoria externa)
- Métodos internos (se ordena en memoria principal)
  - Directos (Simples)
    - Selección
    - Burbuja
      - Simple
      - Mejorado
      - Mejorado II
    - Inserción
  - Indirectos (Avanzados)
    - Shell
    - Ordenamiento Rápido (Quicksort)
    - Ordenamiento por mezcla (Merge Sort)

# SELECCIÓN

A[0] A[1] A[2] A[3] A[4]

51	21	39	80	36
----	----	----	----	----

↓  
pasada 0

Pasada 0. Seleccionar 21  
Intercambiar 21 y A[0]

21	51	39	80	36
----	----	----	----	----

↓  
pasada 1

Pasada 1. Seleccionar 36  
Intercambiar 36 y A[1]

21	36	39	80	51
----	----	----	----	----

↓  
pasada 2

Pasada 2. Seleccionar 39  
Intercambiar 39 y A[2]

21	36	39	80	51
----	----	----	----	----

↓  
pasada 3

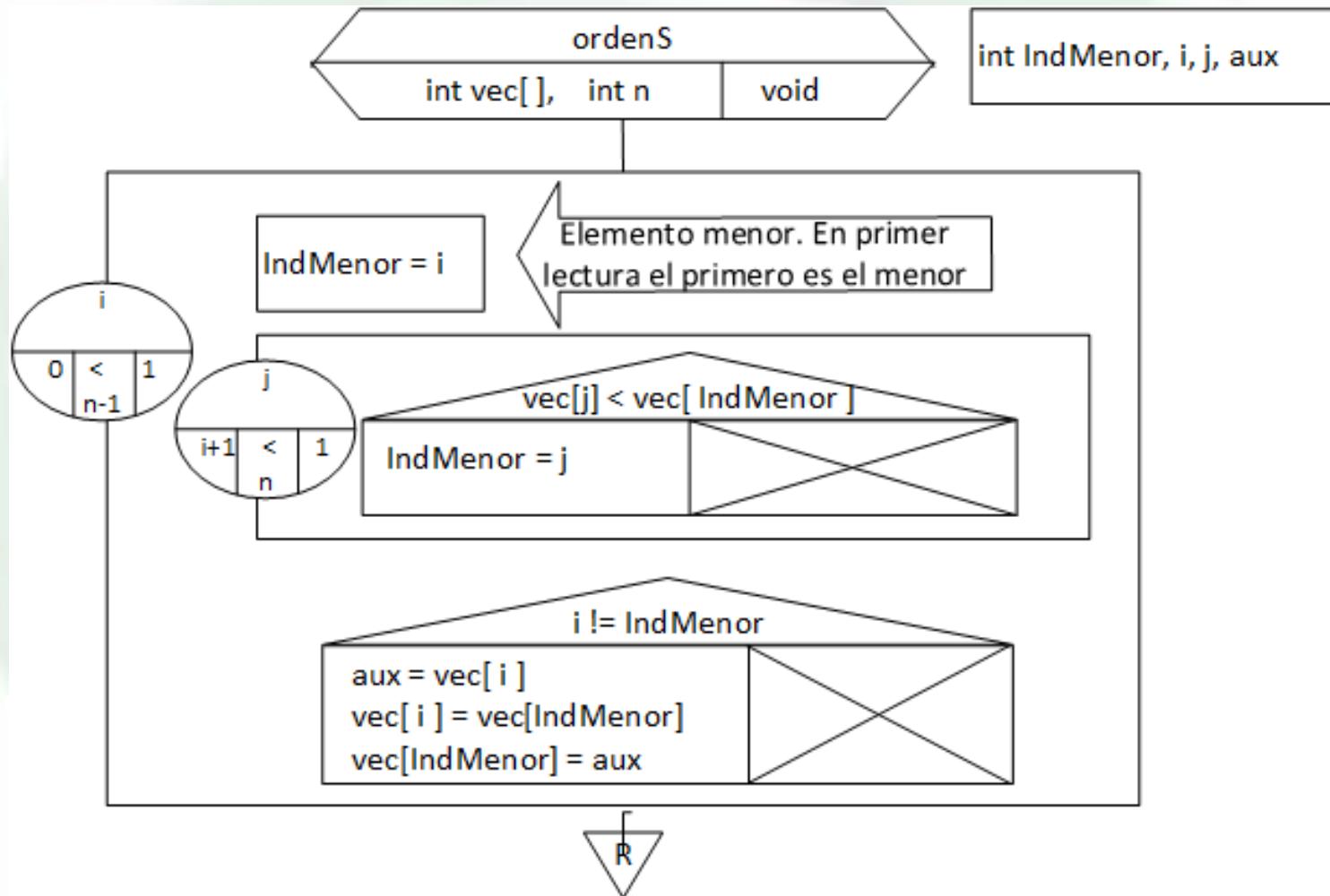
Pasada 3. Seleccionar 51  
Intercambiar 51 y A[3]

21	36	39	51	80
----	----	----	----	----

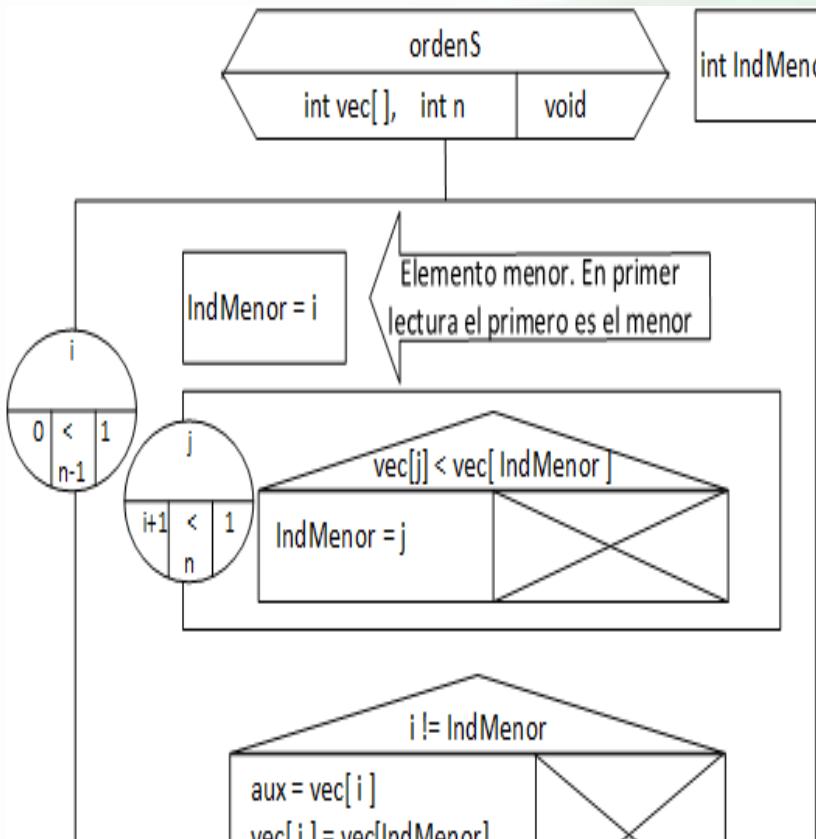
Lista ordenada

8  
5  
2  
6  
9  
3  
1  
4  
0  
7

# SELECCIÓN – DIAGRAMA LÓGICO



# SELECCIÓN – CÓDIGO C

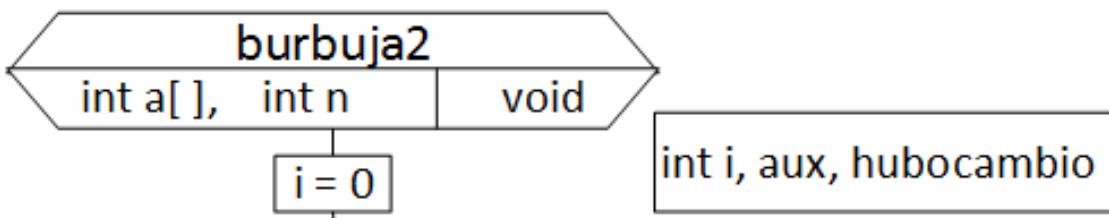


```

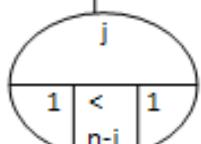
void ordenS (int vec[], int n)
{
    int indMenor, i, j, aux;
    for (i = 0; i < n-1; i++)
    {
        indMenor = i;
        for (j = i+1; j < n; j++)
            if (vec[j] < vec[indMenor])
                indMenor = j;

        if (i != indMenor)
        {
            aux = vec[i];
            vec[i] = vec[indMenor];
            vec[indMenor] = aux ;
        }
    }
}
  
```

# BURBUJEOS MEJORADO 1

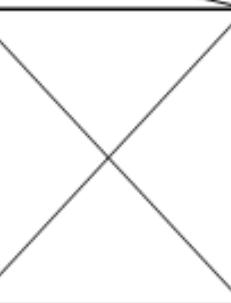


hubocambio=0



a[j-1] > a[j]

aux = a[j-1]  
a[j-1] = a[j]  
a[j] = aux  
hubocambio = 1



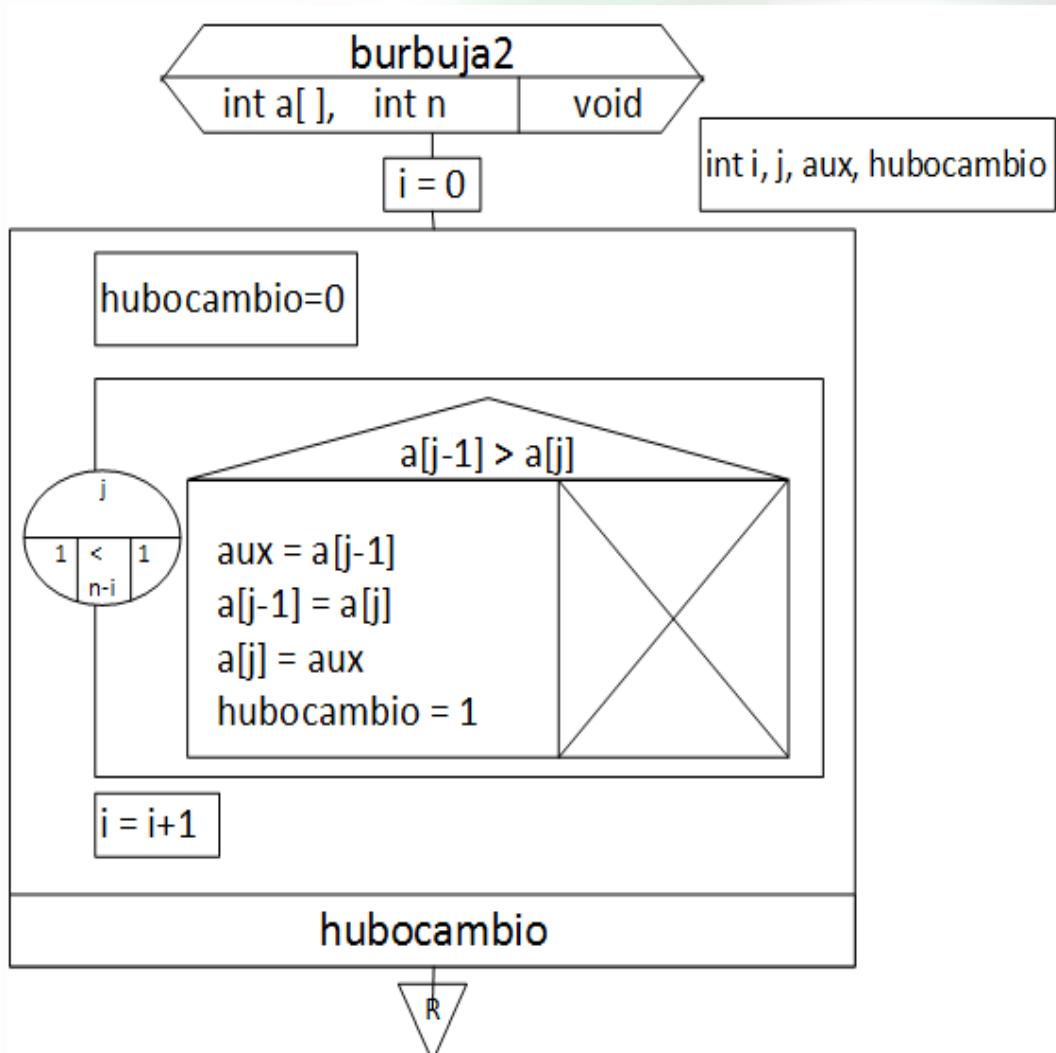
i = i+1

hubocambio



6 5 3 1 8 7 2 4

# BURBUJEOS MEJORADO 1



```
void burbuja2 (int a[ ], int n )  
{  
    int i,j, hubocambio, aux;  
    i=0;  
    do {  
        hubocambio=0;  
        for( j=1; j<n-i; j++)  
            if (a[j-1] > a[j])  
            {  
                aux = a[j-1];  
                a[j-1] = a[j];  
                a[j] = aux;  
                hubocambio=1;  
            }  
        i = i +1;  
    } while (hubocambio);  
}
```



**¿CONSULTAS?**

MUCHAS GRACIAS  
POR SU ATENCIÓN.

MANOS A LA OBRA y...

SUERTE



# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN" "http://www.w3.org/TR/1999/REC-html401-19991224/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones en programación y graficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>

```

## Arreglos bidimensionales: Matrices

# Ejercicio

Una empresa de ventas por Internet, desea hacer un estudio sobre la venta diaria. Para ello, por cada venta realizada durante el día recibe los siguientes datos:

- Hora de la venta (entero, de 1 a 24)
- Artículo vendido (entero, de 1 a 10)
- Cantidad vendida (entero, mayor que cero)

El fin de ventas del día se da con una venta con hora igual a cero.

Informar:

- a) El detalle de la cantidad de artículos vendidos por hora.

## 10 minutos para plantearlo

# Ejercicio

El problema con este ejercicio es que los datos dependen de ***dos variables de entrada***, y con un vector solamente tenemos una...

Una solución aparente sería tener un ***vector de vector***, es decir, un vector que en cada posición tuviera otro vector...

Representa a  
las horas

# Ejercicio



# Matrices

Una matriz es una estructura en la que se almacena un conjunto de datos del mismo tipo, agrupados como consecuencia de **dos** características comunes.

Contiene dos subíndices.

También es correcto afirmar que una matriz es un vector cuyos elementos son a su vez vectores.

Todos los componentes de la matriz tienen un único nombre, pero distintos valores de los índices.

## Declaración

Tipo de dato Nombre de la Matriz [cant. Filas][cant.columnas];

## Ejemplo:

```
int Mat[24][10];
```

```
Float Ventas [15][6];
```

# Matrices

Las matrices son arreglos de datos, es decir, son vectores. Sólo que no son unidimensionales sino multidimensionales. Solamente trabajaremos con arreglos bidimensionales.

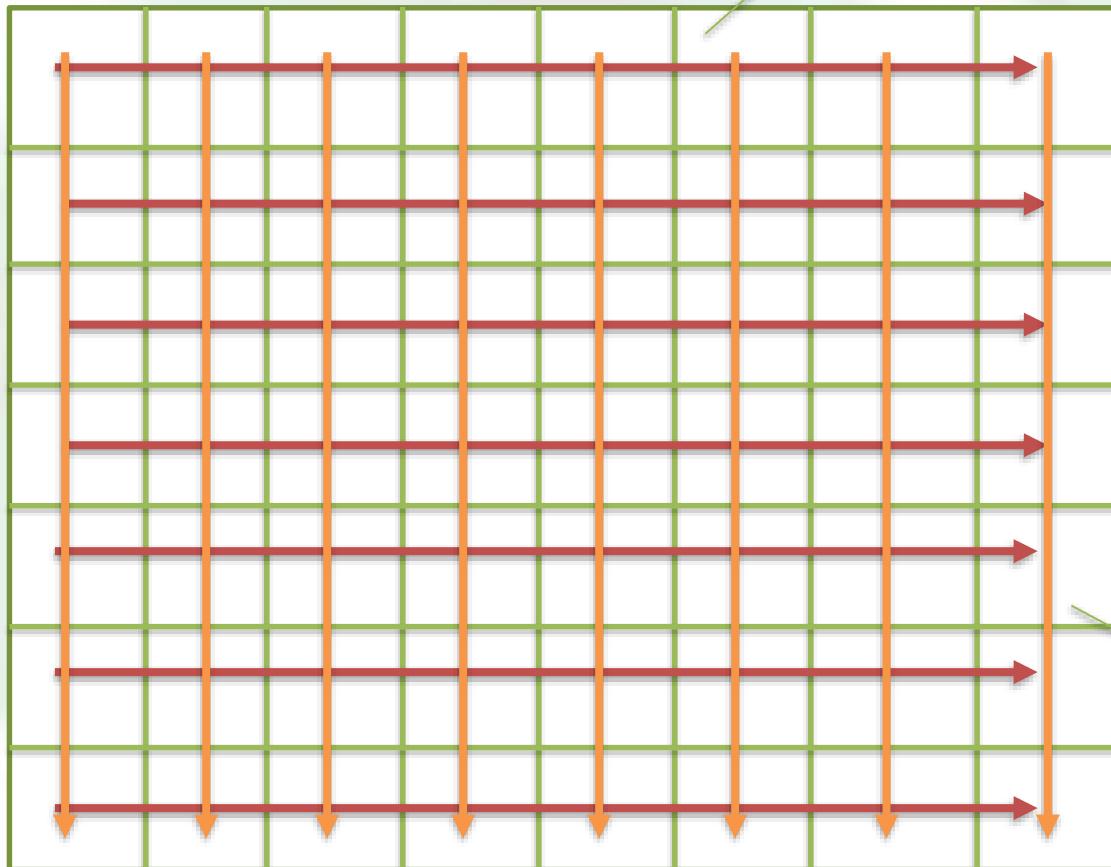
Por lo tanto, al igual que los vectores, puede accederse a una matriz en forma secuencial o en forma directa.

También para recorrer una matriz se utiliza estructuras iterativas, en general, ciclos **for**.

El problema es que como la matriz tiene dos dimensiones, el recorrido puede ser el sentido de cualquiera de los dos índices: filas o columnas

# Matrices

**Recorrido por fila**



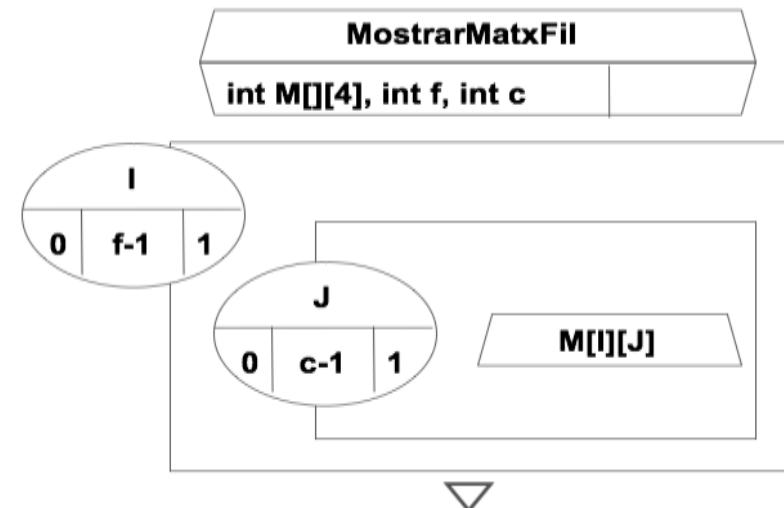
**Recorrido por columna**

# Matrices

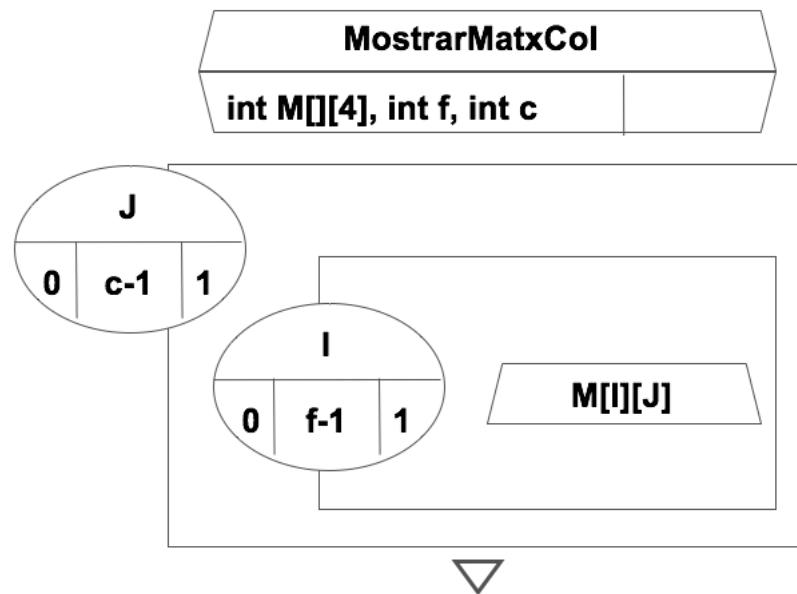
*Ejemplo:*

	0	1	2	3
0	-7	5	32	-1
1	15	17	53	0
2	2	3	22	7

*Por fila:*



*Por columna:*



# Ejercicio – solución

Matrices

PonerMatEnCero (mat, 24, 10)

CargaMat (mat, 24, 10)

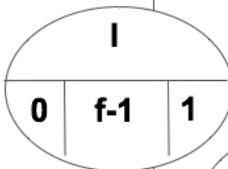
“Hora/Artículos 1 2 3 4 5 6 7 8 9 10”

MostrarMatxFil (mat, 24, 10)



PonerMatEnCero

int M[][][10], int f, int c

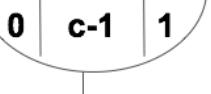


I

0

f-1

J



M[I][J]=0



CargaMat

int M[][][10], int f, int c

Hora = IngrDatosVal (0, f)

Hora != 0

art = IngrDatosVal (1, c)

cant

cant <= 0

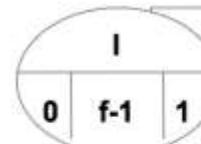
M [Hora - 1][art - 1] += cant

Hora = IngrDatosVal(0,f)



MostrarMatxFil

int M[][][10], int f, int c



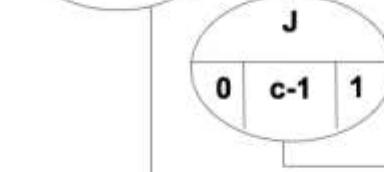
I

0

f-1

1

I + 1



J

0

c-1

1

M[I][J]





# Preguntas????



© PAMS



# Elementos de Programación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f90..."/>
<meta http-equiv="Content-Type" content="text/html; charset=..."/>
<title>Gesatecsl. Soluciones de software para empresas y graficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>
```



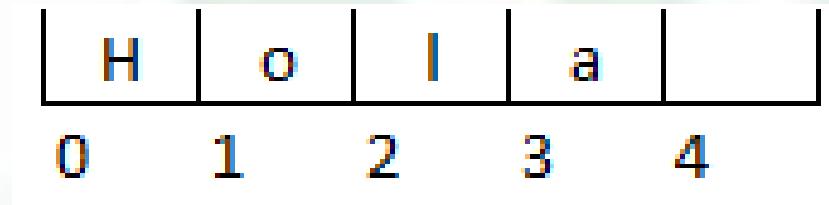
A 3D white character with a blue arm band is shown from the side, holding a large pencil and writing code on a white surface. The character is wearing a blue cap and shorts. The background behind the character is a blurred green and white gradient.

## Manejo de Cadenas

# Diferencia entre arreglos de char y cadenas de caracteres

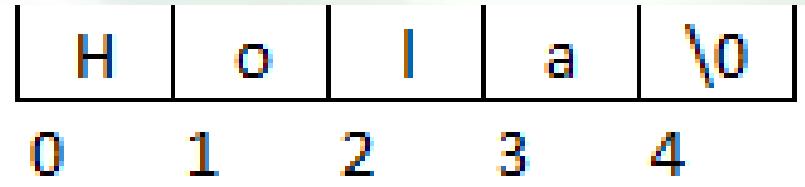
Arreglo de char:

Cad[4]



Cadena (String):

Cad[5]



# Diferencia entre arreglos de char y cadenas de caracteres

Arreglo de char:

Se trata como cualquier otro vector.

Cadena (String):

- Puede tratarse como vector (Agregando valor nulo “\0”).
- Se maneja con funciones específicas. (string.h)

# Declaración

- Biblioteca de cabecera :
  - string.h
- Declaración de variable:
  - char v\_char(tamaño+1)
    - En donde:
      - tamaño es el tamaño máximo de lo que se desea guardar y
      - 1 equivale a la posición del terminador. (Tener en cuenta que C no controla no pasarse de tamaño).

# Inicialización de String

- En la declaración
  - `char cadena_hola[]="Hola"; //Vector 5 elementos`
  - `char otro_hola[]={'H','o','l','a','\0'}; //Idem anterior`
  - `char cadena_vacia[]=""; //Vector de 1 elemento '\0'`
- Fuera de la declaración
  - `gets (nombre) //Permite espacios`
  - `scanf ("%s", nombres) /*Corta ante la aparición de un espacio*/`

# Inicialización de String

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

int main()
{
    char nombre[25], nombres[25];
    printf("Ingreso con gets():\n");
    gets(nombre);
    printf("\nIngreso con scanf():\n");
    scanf("%s", nombres);
    printf("Resultado con gets: %s\n", nombre);
    printf("Resultado con scanf: %s\n", nombres);
}
```

:\ntos\Material Gabriel\Cadenas\PruebaCar

```
Ingreso con gets();
Gabriel Garcia
```

```
Ingreso con scanf();
Gabriel Garcia
Resultado con gets: Gabriel Garcia
Resultado con scanf: Gabriel
```

# Inicialización de String

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

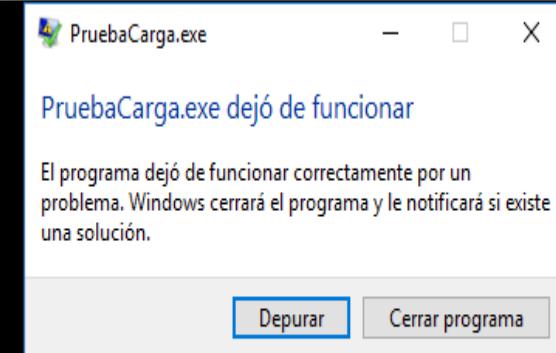
int main()
{
    char nombre[25], nombres[25];
    printf("Ingreso con gets():\n");
    gets(nombre);
    printf("\nIngreso con scanf():\n");
    scanf("%s", nombres);
    printf("Resultado con gets: %s\n", nombre);
    printf("Resultado con scanf: %s\n", nombres);
}
```

E:\ElEmProgramacion:

Ingreso con gets();  
Me pienso pasar deliberadamente del tamaño máximo.

Ingrese con scanf();  
Me pienso pasar deliberadamente del tamaño máximo.

Resultado con gets: Me pienso pasar deliberadamente del tamaño máximo.  
Resultado con scanf: Me



# Uso de fgets(vec, tam, origen)

```
#include<stdio.h>
#include<string.h>
int main()
{   char cadena[25];
    printf("Ingreso con fgets();\n");
    /* cadena:vector de string
       25: Tamaño del vector -Límite 24
       stdin: Origen de la cadena. En este caso flujo de entrada*/
    fgets(cadena,25,stdin);
    printf("Resultado con fgets: %s \n", cadena);
    printf("Longitud de la cadena ingresada: %d", strlen(cadena));
}
```

"E:\ElemProgramacion\1ro2017elementos\Material Gabriel\Cadenas\PruebaCarga2.exe"

Ingreso con fgets();

Me pienso pasar deliberadamente del tamaño establecido.

Resultado con fgets: Me pienso pasar delibera

Longitud de la cadena ingresada: 24

# Funciones para manejo de cadenas

- **strlen(cadena)** (string length)
  - Retorna un entero que indica longitud de una cadena sin incluir el carácter de finalización.
- **strcpy(cadena destino, cadena de origen)** (string copy)
  - Copia una cadena a otra.
- **strcat(cadena receptora, cadena a transferir)** (string concatenate)
  - Concatena dos cadenas dejando el resultado en la primera.
- **strcmp(cad1, cad2)** (string compare)
  - Compara dos cadenas y devuelve:
    - 0 si son iguales
    - >0 si cad1 es mayor a cad2.
    - <0 si cad1 es menor a cad2.
- **strstr(cad1, cad2) //No la usamos por devolver punteros** (string string)
  - Busca la existencia de cad2 dentro de cad1.
- **strcasecmp(cad1, cad2) stricmp(cad1, cad2)** (string compare ignore)
  - Equivalente a strcmp(), no diferencia entre mayúsculas y minúsculas. No son ANSI C sino C++, strcasecmp no es una función sino una macro que llama a strcmp().

# Declaración de vector de cadena

## Vector de string

En forma general:

Tdato Nombre **vector**[tamaño vector][tamaño string];

Ejemplos:

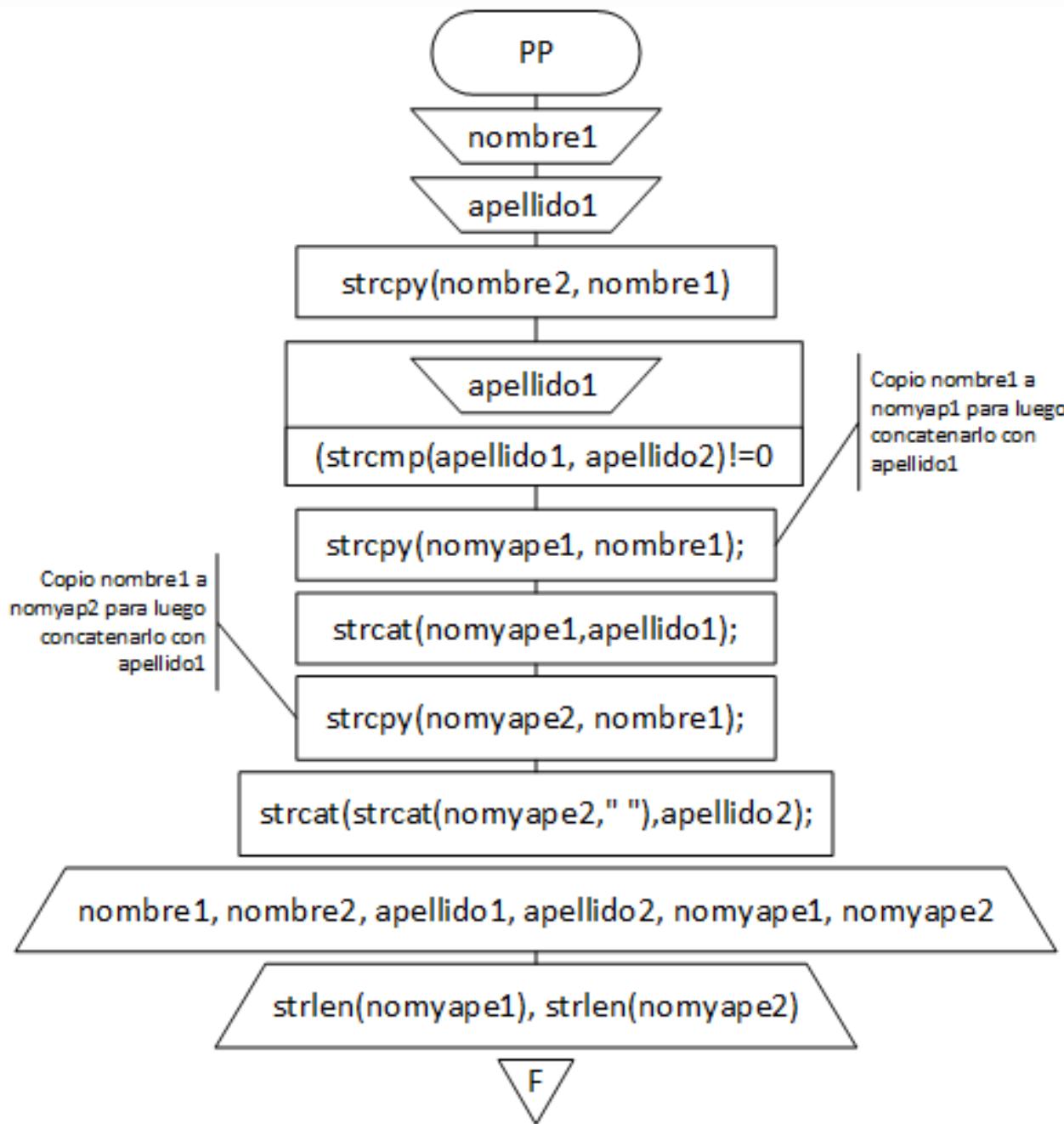
**char ApeyNom [100][31];**

**char Telef [25][12];**

**char Sucursal [8][26];**

# Ejercicio

1. Ingrese nombre y apellido en variables separadas.
2. Copiar el nombre en una variable adicional.
3. Ingrese nuevamente el apellido en una variable nueva y reingrese hasta asegurarse que sean iguales.
4. En otra variable unir apellido y nombre.
5. Repetir el punto 4 permitiendo que quede un espacio entre ambos.
6. Muestre todas las variables cargadas y el tamaño utilizado en las variables de los puntos 5 y 6.



```

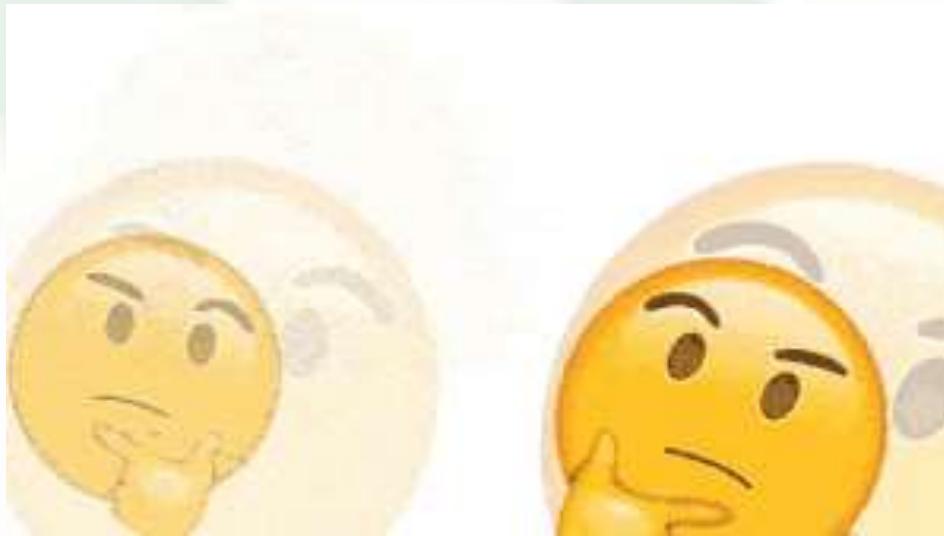
#include<stdio.h>
#include<string.h>
void main()
{
    char nombre1[11], nombre2[11], apellido1[11], apellido2[11],
        nomyape1[21], nomyape2[21];
    puts("Ingrese Nombre");
    fflush(stdin);
    gets(nombre1);
    puts("Ingrese Apellido");
    fflush(stdin);
    gets(apellido1);
    strcpy(nombre2, nombre1);
    do
    {
        puts("ingrese nuevamente el apellido");
        fflush(stdin);
        gets(apellido2);
    }while (strcmp(apellido1, apellido2) !=0);
    //Preparo nomyape1
    strcpy(nomyape1, nombre1);
    strcat(nomyape1, apellido1);
    //Preparo nomyape2 y
    strcpy(nomyape2, nombre1);
    strcat(strcat(nomyape2, " "), apellido2);
    printf("\n nombre1: %s \n nombre2: %s", nombre1, nombre2);
    printf("\n apellido1: %s \n apellido2: %s", apellido1, apellido2);
    printf("\n nomyape1: %s \n nomyape2: %s", nomyape1, nomyape2);
    printf("\n Longitud nomyape1: %d", strlen(nomyape1));
    printf("\n Longitud nomyape2: %d", strlen(nomyape2));
}

```

E:\ElemProgramacion\1ro2017elementos\Material Gabriel\Cadenas\Ejer  
 Ingrese Nombre  
 Gabriel  
 Ingrese Apellido  
 Garcia  
 Ingrese nuevamente el apellido  
 Larrosa  
 Ingrese nuevamente el apellido  
 Garcia  
 nombre1: Gabriel  
 nombre2: Gabriel  
 apellido1: Garcia  
 apellido2: Garcia  
 nomyape1: GabrielGarcia  
 nomyape2: Gabriel Garcia  
 Longitud nomyape1: 13  
 Longitud nomyape2: 14



# Preguntas????





MUCHAS GRACIAS

# Elementos de Programación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t
<meta http-equiv="Content-Type" content="text/html;
<title>Gesatecsl. Soluciones de software para casas y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css" />
```

## Estructuras (structs)

# Ejercicio

Se desea hacer un estudio de los alumnos de la Universidad que están cursando el 4to. año de la carrera de Ingeniería (como máximo hay 5000 alumnos). Para ello, de cada alumno se ingresan los siguientes datos:

- ✓ DNI (entero, entre 1 y 99.999.999)
- ✓ Apellido y Nombre (string, de 30 caracteres)
- ✓ Domicilio (string, de 30 caracteres)
- ✓ Sexo (carácter, 'F' o 'M')
- ✓ Fecha de Nacimiento (enteros, día, mes y año)
- ✓ Cantidad de materias aprobadas (entero, mayor o igual a cero)
- ✓ Promedio de notas de las materias aprobadas (real, mayor a cero)

La información finaliza con un DNI igual a cero.

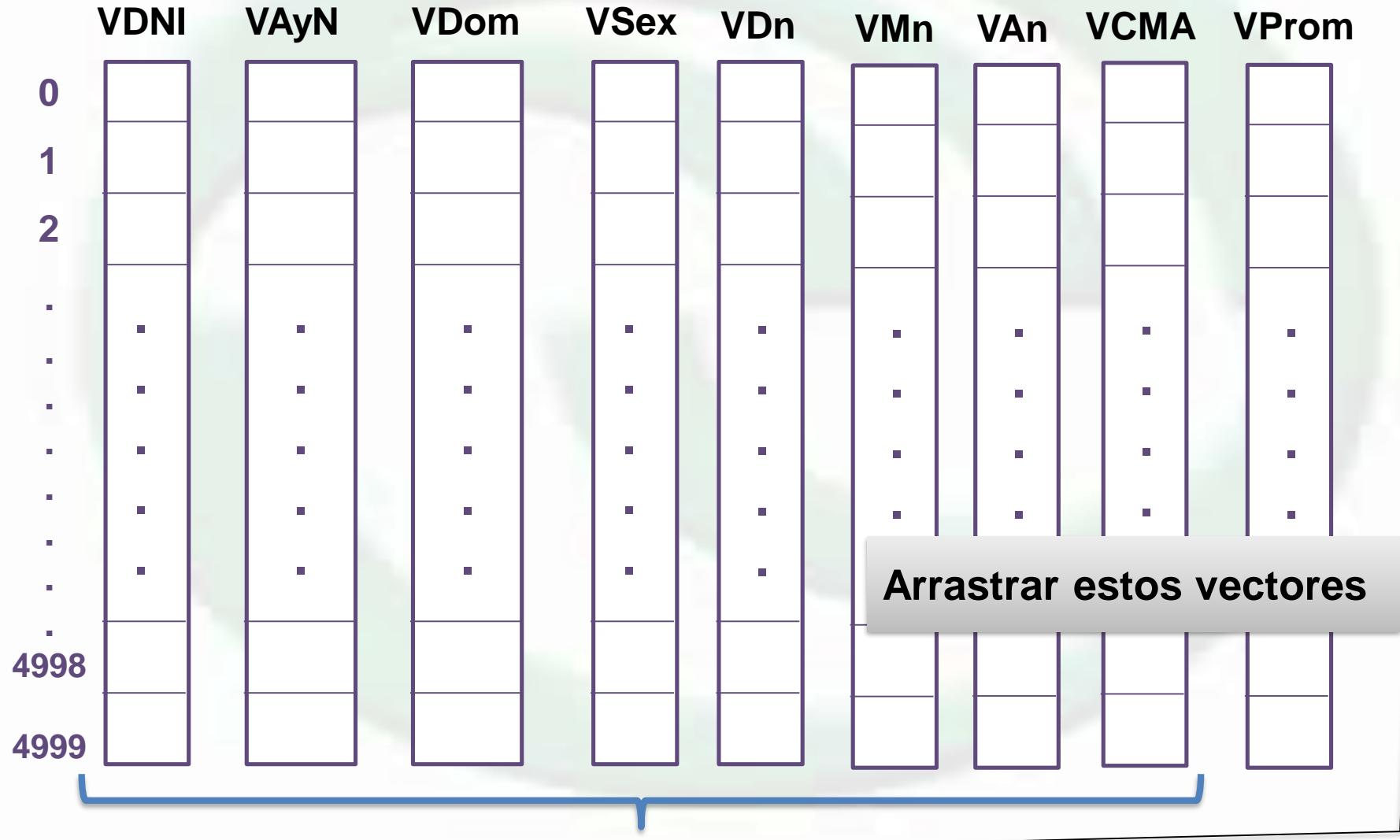
Informar:

- a) Los datos de los alumnos ordenados descendente por promedio.

# Ejercicio

Análisis:

Ordenar por este vector



# Estructuras (struct)

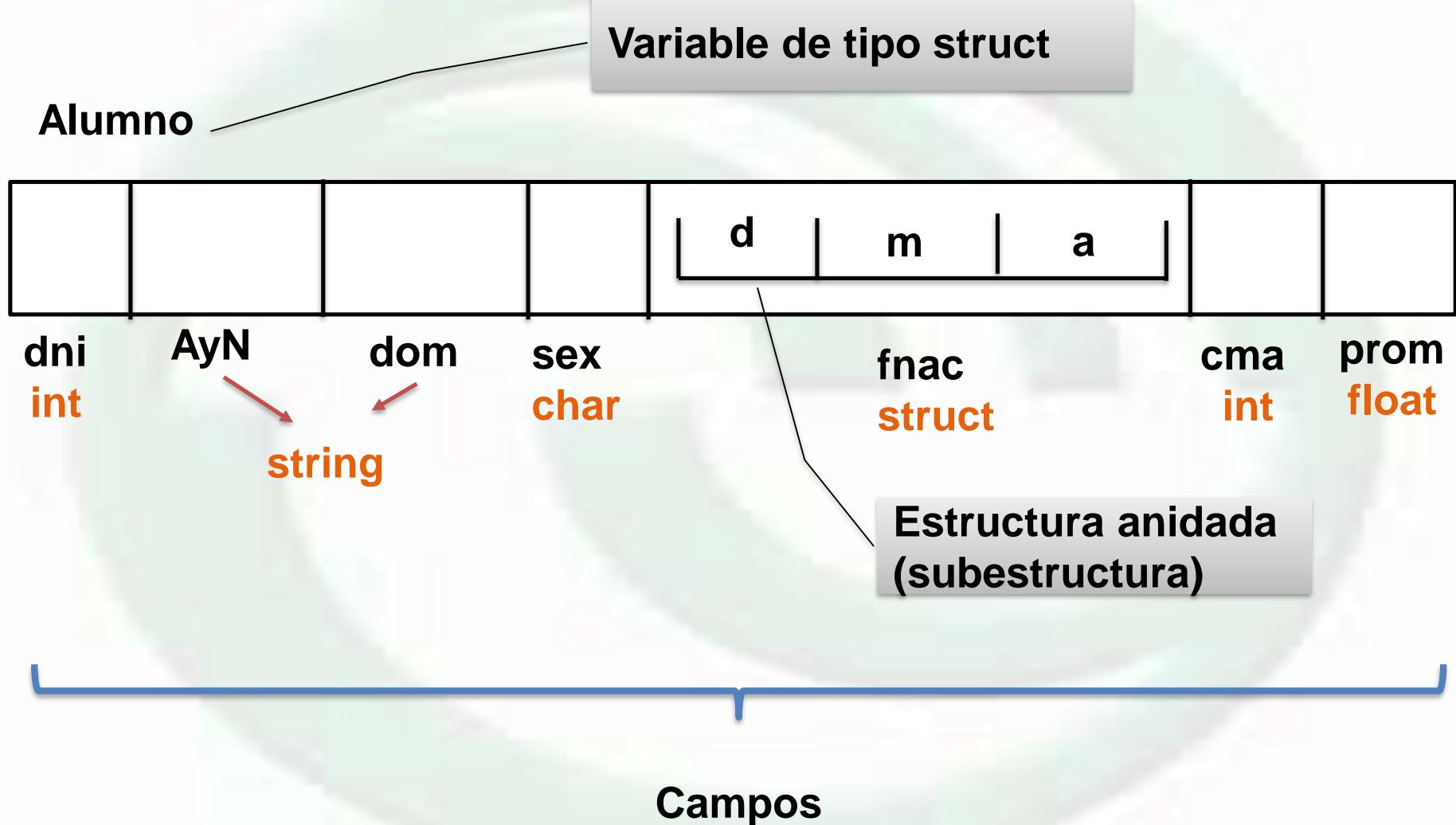
Los datos tipo “array” (vectores y matrices), requieren que sean todos sus componentes del mismo tipo.

El lenguaje “C” da la posibilidad de utilizar otro tipo de datos estructurado, el tipo **Struct** (estructura), en el cual los elementos constituyentes no necesitan ser del mismo tipo.

Cuando se desea procesar datos *lógicamente* relacionados entre sí y que “**no**” son del mismo tipo y referirnos a estos en forma colectiva o individual, usamos el tipo de datos Struct.

Como ejemplo se puede definir ciertos datos de un alumno, que agruparemos en una variable llamada Alumno, que será definida de tipo Struct.

# Estructuras (struct)



# Estructuras (struct)

**La estructura es un tipo de datos con un número fijo de componentes, no necesariamente del mismo tipo, que son accedidos por el “*nombre*” y no por el *subíndice*.**

**El Nombre de campo debe ser ÚNICO dentro de una estructura. Se admite como campo a un Array u otra estructura (*estructuras anidadas*).**

**Las ventajas que reporta son dos, por un lado permite juntar en una definición a datos que se encuentran lógicamente vinculados y por la otra permite simplificar ciertas operaciones ya que se puede operar sobre la estructura completa.**

# Estructuras (struct)

## **Declaración de una estructura:**

La estructura se declara de forma global, es decir, antes de ingresar al main, después de las indicaciones al preprocesador (include) y antes de los prototipos de funciones.

Hay varias maneras de declarar una estructura. Las más comunes son las siguientes:

```
struct Nombre estructura {tipo de dato campo 1;  
                        tipo de dato campo 2;  
                        .....  
                        tipo de dato campo n;};
```

## **Ejemplo:**

```
struct TFecha { int dia;  
                int mes;  
                int anio;};
```

# Estructuras (struct)

```
typedef struct {tipo de dato campo 1;  
               tipo de dato campo 2;  
               .....  
               tipo de dato campo n;} Nombre estructura;
```

## *Ejemplo:*

```
typedef struct { int dia;  
                 int mes;  
                 int anio;} TFecha;
```

Dentro del main, una vez declarado el tipo de dato struct, se puede declarar una variable:

```
TFecha fecnac;
```

# Estructuras (struct)

**Procesamiento de una estructura:**

Puede realizarse en 2 formas diferentes:

## 1. ESTRUCTURAS COMPLETAS:

Es el procesamiento más sencillo, solo se puede asignar una variable tipo estructura a otra si ambas tienen el mismo diseño.

Se puede hacer operaciones de lectura y grabación en “archivos” o usarlos como parámetros.

Una variable estructura **NO** puede utilizarse como operando en expresiones aritméticas ni lógicas.

**Ejemplo:**

TAlumno alum, alumno;  
alum = alumno;

# Estructuras (struct)

## 2. CAMPOS INDIVIDUALES:

Es el procesamiento de los campos individuales, es mucho más frecuente que las estructuras completas.

A cada campo se lo menciona con un “designador de campo”, que es una combinación del nombre de la variable tipo estructura y el nombre del campo.

La forma general es:

**nombre variable tipo estructura.nombre de campo [.nombre subcampo]**

**alumno.nota = 10;**

**strcpy (alumno.nombre, “MARTINEZ”);**

# Estructuras (struct)

## **Pasaje de una estructura a una función:**

Puede realizarse en 2 formas diferentes:

Las estructuras pueden ser utilizadas como parámetros de funciones tanto por valor como por variable.

Pueden pasarse por campos individuales o completos, siendo esto último lo más conveniente.

Además una función puede devolver algo del tipo de dato estructura.

Es el procesamiento más sencillo, solo se puede asignar una variable tipo estructura a otra si ambas tienen el mismo diseño.



# Preguntas????



© PAMS

# Elementos de Programación



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="google-site-verification" content="f9t..."/>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Gesatecsl. Soluciones para la web y las aplicaciones y gráficos</title>
<style type="text/css">
<!--
body {
    margin-left: 0px;
    margin-top: 0px;
    margin-right: 0px;
    margin-bottom: 0px;
}
-->
</style>
<link href="css.css" rel="stylesheet" type="text/css"/>
```

## Corte de Control

# Corte de Control en archivos secuenciales

- ✓ Los registros en los archivos secuenciales están grabados en posiciones físicamente contiguas.
- ✓ Resulta en algunos casos interesante o necesario, ordenar un archivo secuencial por un determinado campo antes de procesar los registros.
- ✓ Esta clasificación u ordenación resulta necesaria cuando existen en el archivo varios campos que repiten su contenido.
- ✓ El problema consiste en determinar el momento preciso en que finalizan los elementos de entrada de un grupo para comenzar con otro.

# ¿Qué es un Corte de Control?

Corte de Control: Es cuando se interrumpe el circuito de instrucciones que se estaban ejecutando.

- ❖ Caso típico de corte de control: Procesos donde se solicita determinados procedimientos para grupos de registros que mantienen cierta homogeneidad y no hay ningún indicio de la cantidad de grupos de registros que contiene el archivo.
- ❖ Es necesario determinar el momento preciso en que finalizan los elementos de entrada de un grupo para comenzar con otro, es decir, se cambia de grupo.
- ❖ En síntesis: Detectar el momento en que cambia el valor (contenido) de la variable «*campo de control*».

# Requisitos para poder hacer Corte de control

- Que los datos de entrada que vienen de un archivo estén **ordenados** por el campo por el cual se quiere realizar el corte.
- Que existan **varios subconjuntos** para que tenga sentido el corte de control.
- Que cada subconjunto tenga **varios elementos o registros**.
- El corte se produce **desde los datos de un archivo**.

# Metodología de Corte de Control

Se debe tener en cuenta:

- Cuando se lee un registro de un archivo, su contenido se guarda en **memoria** en **variables asociadas** a dicho archivo. Luego, cuando se lee el siguiente registro, su contenido se almacena en las mismas variables *destruyendo* la información almacenada del registro anterior.
- Por lo tanto para saber si el campo de control del registro recién leído tiene el mismo contenido que el registro anterior, **será necesario haber almacenado en una variable auxiliar el contenido del campo de control del primer registro del grupo o bloque**, para poder compararlo con él, y de esta manera determinar en forma precisa cuando se produce la ruptura del proceso **corte de control**.

# Metodología de Corte de Control

- Cuando esto se produce (**la salida del corte**), estaremos seguros que el bloque o conjunto de datos que estábamos procesando ha finalizado. Es este el momento de realizar las operaciones relacionadas con la finalización del conjunto ( impresión de contadores, impresión de acumuladores, acumular para totales generales, etc.)
- Luego de esto se debe volver a **inicializar** la variable auxiliar con el contenido del nuevo campo de control del conjunto nuevo a procesar a fin de poder usarlo como referencia de este nuevo bloque.
- Se utiliza la estructura ***mientras (while)*** para repetir el conjunto de operaciones relativas a los registros del mismo conjunto. La condición de salida es que el campo auxiliar sea distinto al campo recién leído.

# Estructura de Corte de Control

Debe realizarse una primer lectura antes de entrar a la estructura.

Se deberá inicializar variables relacionadas con el comienzo de cada bloque o conjunto de control.

Antes de entrar a la estructura repetitiva, se inicializarán los acumuladores y contadores.

Se debe realizar la nueva lectura como última instrucción de la estructura repetitiva interna.

Las operaciones de todo el archivo se realizarán inmediatamente al salir de la estructura general.

Habrá una estructura repetitiva general cuya condición de salida será la finalización del archivo (EOF).

Condición de repetición

Estructura de repetición

Las operaciones relativas a la finalización de un bloque de control se realizarán inmediatamente al salir de la estructura asociada.

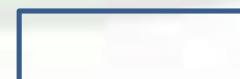
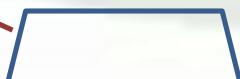
!feof(..)

VarAnt = reg.var

Cont = 0      sum = 0

VarAnt == reg.var Y !feof(..)

Realización de operaciones



# Tipos de Corte de Control

## Corte Simple

Se da cuando existe un solo campo de control y por lo tanto un solo corte de control.

## Corte Compuesto

Cuando existen varios campos de control de los cuales queremos obtener información. En otras palabras, existirán tantas estructuras repetitivas anidadas dentro de la estructura general como cortes de control haya, entonces la estructura más externa será aquella que contenga al resto y en consecuencia la mas interna la que representa a la entidad contenida en las demás y donde se produce en reingreso de registros del archivo.

# Características de Corte de Control

## Jerarquía de comparación

Depende del enunciado del problema, es decir, cuál es la división más importante y consecuentemente el diseño de salida de los resultados buscados.

Depende mucho de la organización jerárquica de los datos (Archivo), es decir de su **ordenamiento**.



Si el archivo no está ordenado **NO SE PUEDE** aplicar corte de control. (Solución: Arreglos / Vectores).

# Características de Corte de Control

## Corte por arrastre

En el caso de tener dos cortes, el segundo corte arrastra la pregunta anterior. Si tuviese un tercer corte el último corte arrastra las dos preguntas anteriores.

Cuando se tengan mas de un corte de control se deben transferir un valor por EOF, no solo a la primer pregunta, sino a todas, caso contrario puede producirse algunos errores en algunas preguntas de las estructuras.

# Ejemplo de Corte de Control

## Recordemos el ejercicio 1 de archivos:

1. Un banco tiene el archivo secuencial SALDOS, con los saldos de sus casi 20000 clientes de Caja de Ahorro, al inicio del mes. Cada registro tiene los siguientes datos:

- Número de Caja de Ahorro (entero, de 6 cifras)
- Apellido Y Nombre (alfanumérico de 30 caracteres)
- Saldo (real, mayor o igual a cero)

Existe además un segundo archivo secuencial con los movimientos del mes, llamado MOVI, ordenado por sucursal del banco. Cada registro contiene:

- Sucursal del Banco (entero de 4 cifras no correlativo)
- Número de Cuenta de Ahorro
- Mes (entero)
- Día (entero)
- Código de Operación (entero: 1 → depósito (suma al saldo); 2 → extracción (resta al saldo))
- Importe (real, mayor que cero)

En el caso de las extracciones, antes de proceder a realizar la resta del importe al saldo, evaluar si éste es mayor al importe (no puede quedar saldo negativo en la Caja de Ahorro). En este caso, guardar en el archivo SIN\_SALDO los datos que vienen en el archivo MOVI.

Confeccionar un programa para que:

- a) Se actualice el archivo SALDOS con los movimientos registrados.
- b) Imprima un listado con las cuentas que en el mes han realizado más de 5 extracciones.

# Ejemplo de Corte de Control

**Ahora realicemos el ejercicio 8 que modifica el ejercicio 1**

1. En el ejercicio 1 realice la siguiente actividad:
  - a) Informe qué cantidad de dinero se depositó en cada sucursal del banco.
  - b) Informe las cuentas de las cuales no se pudo extraer dinero por sucursal.

# Ejemplo de Corte de Control

Recordemos como quedaba el main:



PonerEnCero (VExt, 20000)

cant = CargarArch (VArt, 20000)

ProcesarArch (VArt, VExt, cant)

GrabarArch (VArt, cant)

MostrarMayorQue (VArt, Vext, cant, 5)



La pregunta es:  
¿dónde hacemos el Corte de Control???

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

ss  
aux

pf

aux

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

sucant = aux.suc

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

ss  
aux

pf

aux

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

sucant = aux.suc

sucant == aux.suc Y !feof (pf)

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

ss  
aux

pf

aux

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

sucant = aux.suc

sum\_suc = 0

sucant == aux.suc Y !feof (pf)

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

sum\_suc += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

ss  
aux

pf

aux

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

sucant = aux.suc

sum\_suc = 0

sucant == aux.suc Y !feof (pf)

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

sum\_suc += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

ss  
aux

aux.nca

pf

aux

ProcesarArch  
T\_Saldo V[], int E[], int ce

pf = fopen («MOV1», «rb»)

ss = fopen («SIN\_SALDO», «wb»)

pf

aux

!feof (pf)

sucant = aux.suc

sum\_suc = 0

sucant == aux.suc Y !feof (pf)

pos = Busqueda (V, ce, aux.nca)

pos != -1

aux.cop == 1

V[pos].sdo += aux.imp

sum\_suc += aux.imp

V[pos].sdo >= aux.imp

V[pos].sdo -=  
aux.imp

E[pos]++

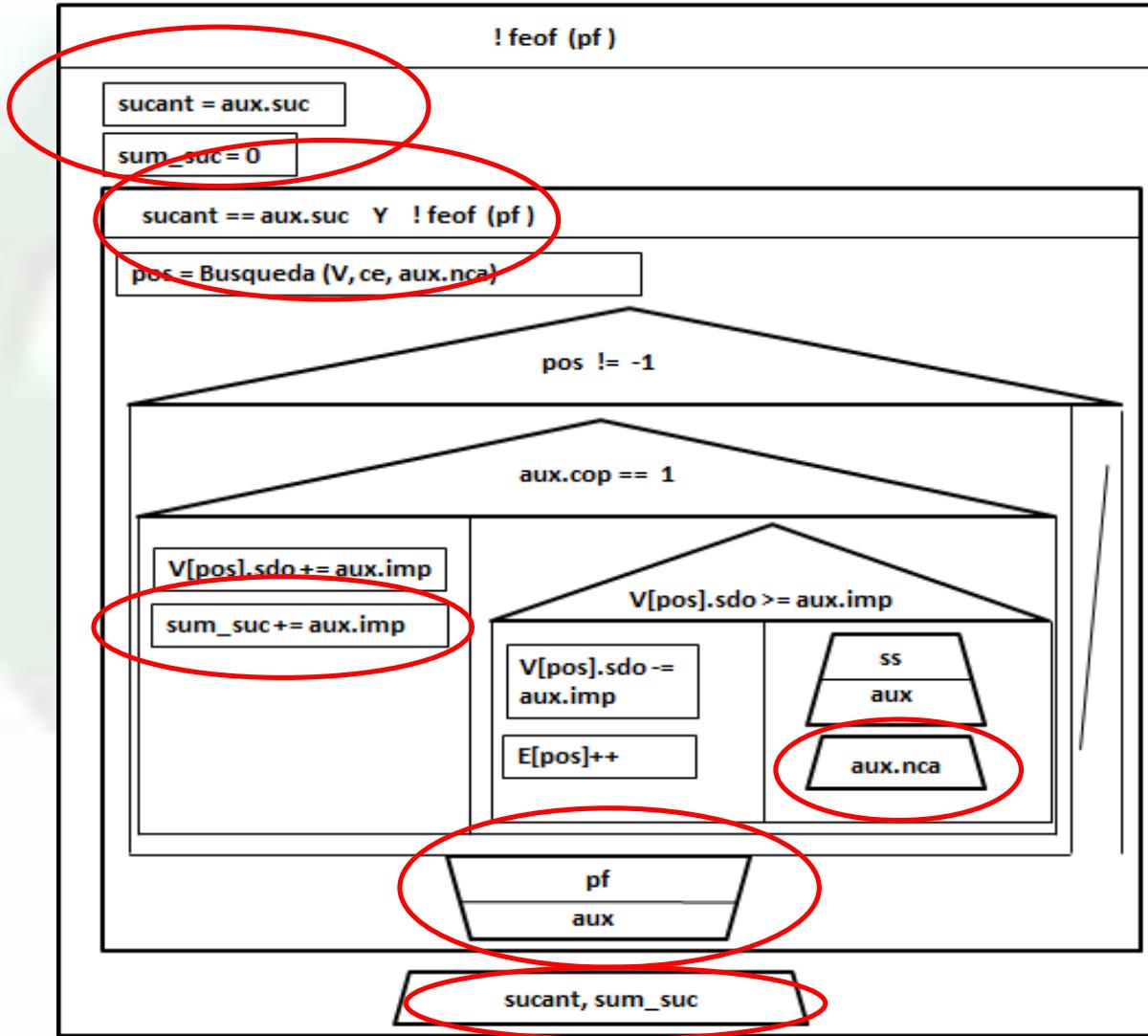
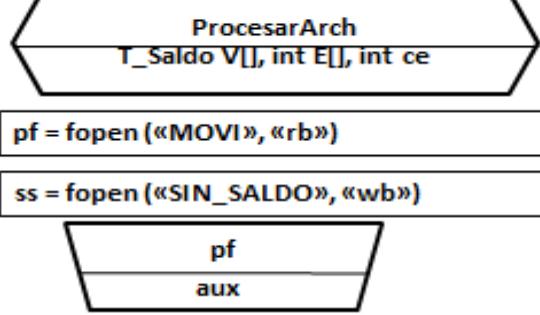
ss  
aux

aux.nca

pf

aux

sucant, sum\_suc





# Preguntas???

