

Fundamentos De Programación en C (C++ , C#, pseudocódigo)

Introducción:

El lenguaje C fue inventado por Dennis Ritchie en 1972 cuando trabajaba en el diseño del sistema operativo UNIX.

Compiladores e IDES (interfaces) del lenguaje C:

- **Borland:** C++ y **Tcc** Vers liberada en: Embarcadero.com. O C++Builder en RAD Studio en modo consola.
- G++ para Linux: Ejecutar: **gcc**
- **Visual Studio C++:** VS express versión gratuita de Microsoft. Crear proyecto (File – New – Project - Win32 Console application)
- **Dev C++:** Compilador de software libre con IDE de www.bloodshed.net: File - New Source File
- **Codebloccs** es un Opensource free C++ descargable desde la página <http://www.codeblocks.org/>. Utilizar el compilador GNUCC MinGW para Windows.
- **Eclipse** es un conjunto de herramientas de programación para varios entornos de desarrollo. Permite integrar C++ o Java. La página oficial es: <http://www.eclipse.org>
- **Embarcadero C++Builder** (Antiguo Borland) RAD para programar en C++ Visual. Incluye desarrollo desde consola.

Características de los lenguajes.

- **Compilados:** C, C++ , C#, V basic, Cobol, Delphi Pascal. Estos lenguajes son traducidos a código máquina y leídos directamente por el procesador. No necesitan de ningún ejecutor externo, son más rápidos, pero dependen de la plataforma o S. Operativo.
- **Interpretados:** Java, Python... Para ejecutar el código necesitan un programa externo que lo interprete ("runeador" o máquina virtual). Son algo más lentos ya que son traducidos en el momento de su ejecución, pero el mismo código puede ser interpretado en varias plataformas. Esto los hace más versátiles.

Características de C:

- Es un lenguaje de propósito general, de medio nivel. Permite programar a alto nivel y a bajo nivel.
- Es un lenguaje portátil. Los programas escritos en C son fácilmente transportables a otros sistemas.

Pasos para crear y ejecutar un programa en C:

- 1º. **Escribirlo en un editor:** Cualquier editor que genere ficheros de texto: serán los ficheros fuentes.
- 2º. **Compilarlo con un compilador:** El compilador "*traduce*" nuestro fichero fuente en un fichero objeto en código máquina.
- 3º. **Enlazarlo("Linkar"):** El enlazador produce un fichero ejecutable a partir de los ficheros objetos.
- 4º. **Ejecutarlo.**

Conceptos.

Programa: Conjunto de instrucciones que entiende un ordenador para realizar una actividad.

Todo programa tiene un objetivo bien definido. Para la resolución de un problema hay que plantear un algoritmo.

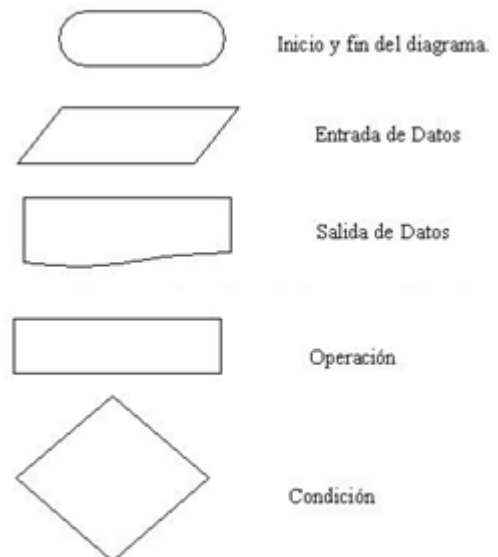
Algoritmo: Son los pasos a seguir para resolver un problema.

Ayudas al planteamiento del algoritmo

- **Diagrama de flujo:** Es la representación gráfica de un algoritmo. Los símbolos gráficos a utilizar para el planteo de diagramas se muestran al lado. Resulta mucho más fácil entender un gráfico que un texto. El diagrama de flujo nos identifica claramente los datos de entrada, operaciones y datos de salida.
- **Pseudocódigo:** Falso código que describe un algoritmo de programación de forma informal.

Tipos y diferencias de C

- **C** fue creado en 1972. Es lenguaje de programación muy popular para crear software de sistemas operativos, aunque también se utiliza para crear aplicaciones. Es un lenguaje "compilado" no "interpretado". Se trata de un lenguaje de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C.
- **C++** es un lenguaje de los años 1980. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido: (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.
- **C#** (pronunciado "Si Sharp" en inglés): Lenguaje de programación orientado a objetos desarrollado por Microsoft para su plataforma .NET. Su sintaxis básica deriva de C/C++ . El nombre C Sharp fue inspirado por la notación musical, sugiriendo que C# es superior a C/C++.



Procedimientos, acciones y funciones

Su objeto es agrupar un programa en subconjuntos por bloques comunes.

- **Acción:** Subprograma que realiza un proceso o algoritmo sin devolver ningún valor al programa principal.
- **Procedimiento:** Subprograma que recoge un valor del programa principal para realizar una acción con él.
- **Función:** Subprograma que recoge un dato del programa principal y le devuelve otro.

Sintaxis: `identificador_devuelto nombredefuncion (identificador_recogido)` si no tiene identif. se puede poner: void

Identificadores: Nombre que hace referencia a una función, variable o constante. No puede contener espacios en blanco, acentos ni caracteres extraños. Ojo: Distingue mayúsculas de minúsculas ni empezar por un número.

Función Main: Todo programa en C tiene una función principal **main** que puede iniciarse con la siguiente estructura:
`main()` o también `int main ()` o `void main (void)` ya que no lleva ningún parámetro entre paréntesis.

Palabras clave: Son palabras reservadas por el programa y no las podemos emplear como nombres de identificadores.

Comentarios: `/*` comentarios entre varias líneas `*/` o `//` comentario hasta final de línea

Operador de visibilidad :: Permite acceder a una variable global cuando está oculta por otra del mismo nombre.

Ordenes de entradas y salidas (i/o)

Mostrar mensajes en pantalla:

- En C: utilizamos `printf("entre comillas valor fijo");` sin comillas variable
- En C++: podemos utilizar la función `cout` de la librería `iostream`: `cout << "Hola " << endl;`
- En C#: utilizamos el objeto **"Console"**: `Console.WriteLine("Ingrese Horas trabajadas por el operario:");`

Entrada de datos por teclado:

- En C: Usar: `scanf (scan-format) analiza con formato. Ejemplo: scanf("%d",&horas);`
- En C++: podemos utilizar la función `cin` de la librería `iostream`. Ejemplo: `cin>>opcion;`
- En C#: Se utiliza la función `ReadLine` del objeto `Console` con la siguiente sintaxis: `linea = Console.ReadLine();`

Uso de Printf. Muestra un mensaje al usuario por la consola (antigua ventana de fondo negro del DOS símbolo del sistema)

Sintaxis: `printf ("mensaje");`

- Si el mensaje es de texto debe ir entre comillas. Recuerda finalizar con punto y coma ; cada final de línea.
- Si en el mensaje quieres añadir una *variable* debes añadir el signo % para darle formato: `printf("Hola, %s");`
- Si en el mensaje quieres añadir un código de escape no imprimible como un salto de línea o una tabulación debes ponerlo tras el símbolo `\` y se llaman secuencias de escape:

Secuencias de escape más usadas:

Código	Significado	código	Significado
<code>\b</code>	retroceso	<code>\0</code>	carácter nulo
<code>\f</code>	salto de página	<code>\\</code>	barra invertida (\)
<code>\n</code>	nueva línea	<code>\v</code>	tabulación vertical
<code>\r</code>	retorno de carro	<code>\a</code>	alerta (bell, campanilla)
<code>\t</code>	tabulación horizontal	<code>\ddd</code>	constante octal (tres dígitos como máximo)
<code>\"</code>	comillas (")	<code>\xdd</code>	constante hexadecimal (tres dígitosmáx.)
<code>\'</code>	apóstrofo(')		

Primer ejercicio. Mostrar información. Por la consola al usuario: Printf

- Si usas el IDE: **DEV C++** elige: Archivo – Nuevo – Código fuente.
- Si usas el IDE: **Embarcadero/Borland C++ Builder** elige del menú: File - New – Other - Console application
- Si usas el IDE: **Codebloks**: File – New – Project – Console application:
- Si usas el IDE: **Microsoft Visual C++**: Archivo – Nuevo proyecto – Aplicación de consola CLR
- Nombre del proyecto: HolaMundo. Guarda el archivo en tu carpeta
- En el archivo de C++ nuevo, escribe el texto del recuadro.
- La función principal se llama Main. Entre paréntesis recoge parámetros y entre llaves indica el principio y el final.
- Comprueba su funcionamiento según IDE o aplicación que uses:

Embarcadero: Pulsando F9 o en Run ►

CodeBlocs: Build and run  Comprueba la barra de mensajes (View - Logs) que no muestre mensaje de error.

DevC++: Compilar y ejecutar (F11)  en: Ver – Ventana de resultados se muestra si hay algún error.

```
// - PRIMER PROGRAMA EN C -
//-----
#include <stdio.h>
int main ( )
{
    printf ("Hola mundo... \n");
    printf ("y adios.");
    scanf("%d");
}
// los comentarios van seguidos de doble barra oblicua o entre /* */
// incluye un archivo de encabezado (header) con funciones básica e/s
// función principal, (suele retornar una valor vacío void o entero int)
// inicio del bloque de instrucciones de la función main
// muestra un mensaje y salta de línea con \n
// muestra otro mensaje en otra línea
// pausa, también puedes system("pause") o getch()
// fin de la función
```

Segundo ejercicio. Pedir información al usuario.

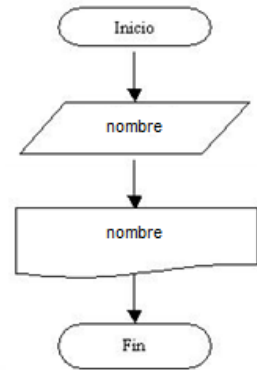
En este ejercicio vamos a preguntarle el nombre al usuario y lo guardaremos en una variable interna llamada *nom* para poder utilizarla más tarde.

En nuestros primeros programas es interesante documentar o explicar el código, añadiendo comentarios al final de la línea después de `//`.

- **Para pedir datos al usuario:** `Scanf - Getchar - Cin`
- **Para mostrar datos al usuario:** `Printf - Cout`

▷ Diagrama de flujo o flujograma:

Inicializa → Pide el nombre → Muestra el nombre → Finaliza



▷ Escritura del código fuente:

En un archivo nuevo escribe el código del recuadro:

```

//--- EJERCICIO 2 DE C ----- //añadimos un título en modo comentario //
#include <stdio.h> //incluye la librería de cabecera estándar Input Output
int main()
{
    char nom[10]; //creamos la variable del tipo carácter que guardará el nombre
    printf("¿Como te llamas?: "); //pide el nombre
    scanf("%s",&nom); //recoge el nombre y lo guarda en la variable nom
    printf("Hola, %s", nom); //muestra un mensaje con la variabe nom
    scanf("%d"); //pausa
}
  
```

Las variables:

Las variables almacenan temporalmente datos en la memoria y podemos cambiar su contenido durante el programa.

Tipos de variables:

Los fundamentales son: **void**, **char**, **int**, **float** y **double**, en los nuevos estándares se incluye también el tipo **bool** y **enum**

Modificadores de variables: permiten ajustar ciertas propiedades de cada tipo: `short`, `long`, `signed` y `unsigned`.

Tipo	Descripción	Long.en bytes	Rango
char	carácter	1	0 a 255
int	entero	2	-32768 a 32767
float	coma flotante	4	aprox. 6 dígitos de precisión
double	coma flotante doble precisión	8	aprox. 12 dígitos de precisión
void	sin valor	0	sin valor
bool	lógico (verdadero o falso)	1	0 a 1

Tipo entero:

Números sin parte fraccional o decimal. Sus valores se pueden escribir de uno de los tres modos siguientes:

- ▷ **Código Decimal:** símbolos del 0 al 9. Ej: 1, 0, -2.
- ▷ **Hexadecimal:** 16 símbolos. Escribir empezando por 0x. Ejemplos: 0xE, 0x1d, 0x8.
- ▷ **Octal:** 8 símbolos. Empiezan por 0. Ejemplos: 02, 010.

Tipo void: Significa sin valor, sin tipo, vacío.

`void` entre paréntesis indica que no tiene argumentos o para coger o retornar

Si se pone delante indica que la función no retorna ningún valor. *Nota:* El nuevo estándar recomienda que la función `main` retorne un valor entero en lugar de vacío.

Ejemplos tipo void:

```

#include <stdio.h>
int main() //recomendado
{
    printf("Version 1.");
    scanf("%d");
}

#include <stdio.h>
void main(void) //vacío
{
    printf("Version 2.");
    scanf("%d");
}
  
```

Tipos float y double

Son números con parte fraccionaria o decimal. El tipo `double` tiene el doble de precisión que el tipo `float`.

La sintaxis para las constantes de estos dos tipos:

`[signo] [dígitos] [.] [dígitos] [exponente [signo] dígitos]`

donde:

- signo → es + o -;
- dígitos → es una secuencia de dígitos;
- . → es el punto decimal;
- exponente → es E o e.

Los elementos que están entre `[]` son opcionales

El número no puede empezar por e o E.

Ejemplos de constantes de este tipo: 1.0e9, -3E-8, -10.1.

El tipo **bool** (booleaniana o lógica). verdadero/falso

Este tipo no existe en C por defecto. A partir de la versión C99 se puede incluir la librería: `#include <stdbool.h>` para poder utilizarla. Ejemplo: `bool correcto = true;`

Ámbito de las Variables: Pueden ser **Globales** o **locales** si se declaran dentro o fuera de la función. Dentro de la función, sólo existen o son válidas en su interior.

ARCHIVOS DE INCLUSION O CABECERA (librerías)

Cada archivo de cabecera (*Header file*) añade a nuestro programa todo un conjunto de funciones que expande el lenguaje. Es necesario hacer uso de la directiva `#include`. El signo `#` indica que la instrucción va para el compilador o preprocesador no para el programa ejecutable. El *nombredearchivo* se escribe entre los signos "`<`" "`>`" para archivos cabecera guardados en la carpeta predeterminada o entre ("`"`" "`"`") si son personales y se suelen guardar junto con el archivo fuente.

Ejemplos:

- `#include <stdio.h>`: Funciones de entrada/salida básicas del lenguaje C (`printf`, `scanf`) `#include <conio.h>`: (`getch()`)
- `#include <stdlib.h>`: Ampliación de funciones básicas con aleatorias, de memoria o de control (`rand`, `free`, `exit`)
- `#include <string.h>`: Para manipulación de cadenas de caracteres. `#include <time.h>`: Funciones para fecha/hora
- `#include <math.h>`: Funciones matemáticas comunes. `#include <complex.h>`: Funciones para números complejos.

FUNCIONES DE E/S:

Entradas-salidas en C:

Printf: Para mostrar el valor almacenado en una variable:
Sintaxis: `printf ("texto de salida", argumentos);`
`printf ("El valor a% d", a);`

Putchar('A') Para mostrar una letra o carácter.

Scanf: Lee datos de entrada y los almacena en una variable:
Ej: `Scanf ("%d", &a);`
Puede obtener varias variables a la vez:
`scanf ("%d %f", &var_entera, &var_float);`

Getchar Para recoger un carácter: ej: letra = `getchar()`;

Caracteres acentuados:

Alt+160 =á - Alt+130 =é - Alt+161 =í - Alt+162 =ó - Alt+163 =ú

```
//SUMAR DOS NUMEROS
#include <stdio.h>
main ()
{
    int nume1,nume2,suma=0;
    printf ("Suma en C.\n");
    printf ("Escriba el primer numero:");
    scanf ("%d",&nume1);
    printf ("Escriba el segundo numero:");
    scanf ("%d",&nume2);
    suma=nume1+nume2;
    printf ("%d mas %d son %d",nume1,nume2,suma);
    printf ("\n");
    printf ("Pulsa RETURN para terminar.");
    scanf ("%d"); //o getch()
}
```

Especificadores o argumentos

Codigo	Formato o variable
%c	Char – 1 caracter simple
%d o %i	Entero decimal
%e	Flotante o doble notación científica
%f	Flotante o doble decimal con signo
%g	Usa el más corto de los anteriores
%o	Entero octal sin signo
%s	Cadena de varios caracteres
%u	Entero decimal sin signo
%x	Entero hexadecimal sin signo
%%	Signo de tanto por ciento: %
%p	Pointer, puntero de dirección
%n	puntero a entero con nº caracteres

Ejemplos:

Sentencia printf ()	Salida
<code>("%f", 123.456)</code>	123.456001
<code>("%e", 123.456)</code>	1.234560e+02
<code>("%g", 123.456)</code>	123.456
<code>("%-2.5f", 123.456)</code>	123.45600
<code>("%-5.2f", 123.456)</code>	123.46
<code>("%5.5f", 123.456)</code>	123.45600
<code>("%10s", "hola")</code>	hola
<code>("%-10s:", "hola")</code>	hola :
<code>("%2.3s", "hola")</code>	hol
<code>("%x", 15)</code>	f
<code>("%o", 15)</code>	17
<code>("%05d", 15)</code>	00015

Funciones comunes:

abs	Retorna el valor absoluto	pow	Eleva a la potencia
asin	Retorna el arco seno	rand	Retorna valor aleatorio
cos	Retorna del coseno	srand	Pone punto de inicio para Rand
exit	Cierra archivos y termina el programa	sqrt	Raíz cuadrada
exp	Eleva a la potencia d	strcat	Concatena cadenas de texto
fabs	Retorna el valor absoluto	strcmp	Compara cadenas de texto
fclose	Cierra el archivo	strcomp	Compara cadenas de texto
feof	Indica si ha llegado al final del archivo	strcpy	Copia cadenas de texto
fgetc/getc	Lee un carácter del archivo	strlen	Cuenta caracteres del texto
floor	Redondea al valor entero	system	Pasa la orden al sistema Op.
fmod	Retorna el resto	tan	Retorna la tangente
fopen	Abre el archivo	time	Retorna segundos transcurridos
fputc	Escribe un carácter en el archivo	toupper	Convierte letra a mayúsculas
log	Retorna logaritmo natural	fabs	Retorna valor absoluto

Ejercicios /ejemplo:

```
// mostrar un numero
//-----
#include <stdio.h>
int main()
{
    printf("%d \n", 123.456);
    printf("%f \n", 123.456);
    printf("%e \n", 123.456);
    printf("%.2f \n", 123.456);
    return 0;
}
```

```
// -- La tercera parte: dividir por 3 un numero entero --
#include <stdio.h>
int main()
{
    int numero; //-> variable entera
    float resultado; //-> variable decimal
    numero=0; //-> asigna un valor
    printf("Escribe un número del 1 al 10: ");
    scanf( "%d",&numero); //pide un valor entero
    resultado=numero/3.00;
    printf("La tercera parte de %d es %.12f", numero, resultado);
    return 0;
}
```

Entradas y salidas en C++ y C#

Entradas-salidas en C++:

Streams – Cin - Cout:

En C++ a diferencia de C las entradas y salidas se leen desde **streams** (*canal, flujo o corriente*)

Para su uso hay que incluir: **#include <iostream.h>** entonces se pueden introducir y sacar datos con

cin: Lee datos (por defecto del teclado) ej: cin >> variable;

cout: Imprime el texto entre comillas tal y como está escrito.

>> : operador de inserción acompaña a cin

<< : operador de extracción acompaña a cout

Ejercicio **cout:**

```
#include <stdio.h> //incluye la librería de cabecera estándar Input Output
#include <iostream> //librería que contiene las funciones cin, cout
main () // función principal
{
    std::cout <<"Hola mundo"<< std::endl;
    std::cout <<"y adios."<< std::endl;
    scanf("%d"); //pausa
}
```

std: garantiza que el compilador acata los estándares un programa en C++

namespace std especifica que los miembros van a utilizarse frecuentemente en un programa

Operador de resolución de visibilidad :: Accede a una variable global cuando está oculta por otra del mismo nombre local.

Repita el ejercicio **Ejercicio2.cpp** cambiando las líneas:

!-- EJERCICIO USO DE cin-cout y namespace !--

```
#include <stdio.h> //incluye la librería de cabecera estándar Input Output
#include <iostream> //librería que contiene las funciones cin, cout
using namespace std; //incluye nombres estándares frecuentes en C++
int main () // función principal, (puede retornar void o int)
{
    double tiempo; //variable tipo numérica decimal de doble precisión
    const float acel=9.8; //constante tipo numérica decimal
    cout <<"Calculo de velocidad en caída libre\n";
    cout <<"Introduzca el tiempo:";
    cin >>tiempo;
    cout <<"En el instante t=" << tiempo << " la velocidad de caída es de "
    << acel*tiempo <<" m/s \n";
}
```

Entradas-salidas en C #: (modo consola)

- Para escribir los datos: métodos Write y WriteLine. La diferencia entre ambos es que Write escribe lo que sea sin añadir el carácter de fin de línea a la cadena.
- Para pedir datos: Los métodos Read y ReadLine no tienen parámetros, así que solamente los podremos usar así:
variable = Console.Read(); cadena = Console.ReadLine();

```
using System;

namespace HolaMundo
{
    class Class1
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hola Mundo");
            string a = Console.ReadLine();
        }
    }
}
```

```
Programación en C# (C sharp)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CalculoSueldo
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Introduzca las horas trabajadas:");
            linea = Console.ReadLine();
            horasTrabajadas = int.Parse(linea);
            Console.Write("Ingrese el pago por hora:");
            linea = Console.ReadLine();
            costoHora = float.Parse(linea);
            sueldo = horasTrabajadas * costoHora;
            Console.Write("El sueldo total del operario es:");
            Console.Write(sueldo);
            Console.ReadKey();
        } } }
```

- En el modo visual C#:

```
if (radioButton1.IsChecked==true)
{
    MessageBox.Show("Hola");
}
```

Constantes

const : Indica que el valor de la variable no se va a modificar en el transcurso del algoritmo.

Su sintaxis es: **const** <tipo_dato> <nombre_constante> = <valor>;

Ejemplo: **const float cambio = 0.92**

Constantes *definidas*:

Es otro modo de declarar una constante. Se declaran con la directiva *#define* por lo que se convierte al compilar.

Ejemplo: **#define CAMBIO 0.92**

Ejercicio - Conversor de Moneda

- Escribe el código de la izquierda. Observa el pseudocódigo y cómo se muestran dos variables a la vez en el mismo print

```

/** Cambio de moneda. En lenguaje C **/

#include <stdio.h>
main()
{
    float euros=0;           //variable para la cantidad en euros
    float dolares=0;        //variable para la cantidad en dólares
    const float cambio=0.92; //constante para el valor de cambio
    printf("CAMBIO DE MONEDA:\n");
    printf("Introduzca valor en dolares: ");
    scanf("%f", &dolares);
    euros=cambio*dolares;
    printf("%6.2f dolares equivalen a %6.2f euros\n", dolares, euros);
    scanf("%d");
}
    
```

Algoritmo en pseudocódigo

```

var    (variables)
    euros, dolares: real
fvar   (fin variables)
constantes
    cambio := 0.92;
fin constantes
inicio
    escribir ("Introduzca valor");
    leer (dolares);
    euros := dolares * cambio;
    escribir ("Resultado: ", euros);
fin algoritmo
    
```

- Quita la constante: **const** float cambio=0.9 y añade la directiva: **#define CAMBIO 0.92** Comprueba la diferencia.
- Al finalizar, guarda este archivo en tu carpeta con el nombre: **cambio.c**

Estructuras básicas de control

- 1.- Secuencial.
- 2.- bifurcación o alternativa.
- 3.- bucle o repetitiva.

Estructura secuencial.

Una estructura secuencial del programa no tiene bucles ni derivaciones. La aplicación ejecuta las líneas del programa en orden desde el inicio al fin.

Ejercicio secuencial: Pedir horas y coste/hora y mostrar el sueldo.

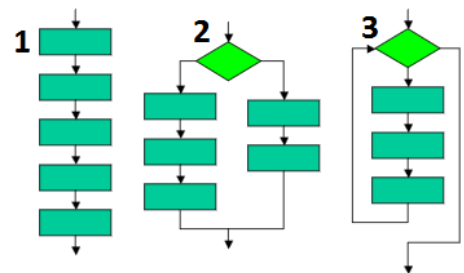
Vamos a crear tres variables: *horasTrabajadas*, *costoHora* y *sueldo*.

Las horas serán enteras pero el coste y el sueldo serán decimales. Así que definimos las siguientes variables en la función main:

- Variables enteras (*int*): **int** horasTrabajadas;
- Variables decimales (*float*): **float** costoHora, sueldo;

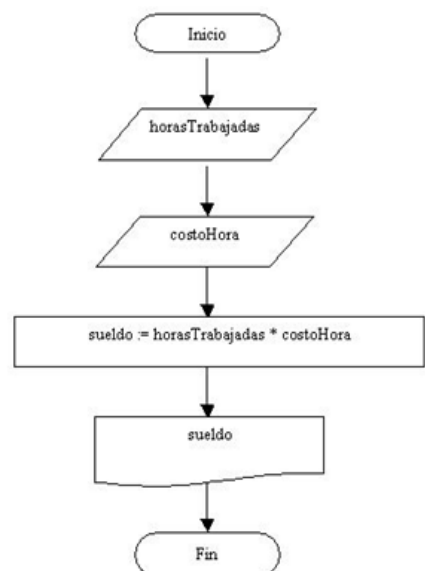
En C las palabras claves deben ir obligatoriamente en minúsculas.

El nombre de las variables no puede tener espacios. Se *propone* que el nombre de la variable comience con minúscula y en caso de estar constituida por varias palabras juntas, suele ir en mayúsculas el primer carácter de la segunda palabra o separados con un guión bajo (ejemplo: *costoHora* o *costo_hora*)



```

/** CALCULO DEL SUELDO **/
#include <stdio.h>           //incluye el archivo librería stdio
int main ()                //función principal
{
    int horasTrabajadas;   // 1 variable entera
    float costoHora, sueldo; // 2 variables decimales
    costoHora, sueldo = 0; // se inicializan con un valor
    printf("Introduzca las horas trabajadas:");
    scanf("%d",&horasTrabajadas);
    printf("Introduzca el precio/hora:");
    scanf("%f",&costoHora);
    sueldo = horasTrabajadas * costoHora;
    printf("El sueldo total del operario es: %f", sueldo);
    printf("\n");
    printf ("Pulsa RETURN para terminar.");
    scanf("%d");           //o getchar() para pausar la consola
}
    
```



Ejercicios de estructura secuencial.

<pre>// ----- PEDIR EL NOMBRE EN C ----- #include <stdio.h> //con la librería stdio int main() { char nombre[20]; int edad printf("Hola, dime tu nombre: "); scanf("%s",&nombre); //coge el nombre printf("Encantado, %s",nombre); printf("\nCuantos anyos tienes?"); scanf("%d",&edad) //coge la edad }</pre>	<pre>//----- PEDIR EL NOMBRE EN C++ ----- #include <iostream.h> //con librería iostream int main() { char nombre[20]; int edad cout<<"Hola, dime tu nombre: "<<endl; cin>>nombre; //coge el nombre cout<<"Encantado"<< nombre endl; cout<<"Cuantos anyos tienes? "<<endl; cin>>edad; //coge la edad }</pre>
---	---

Ejercicio para sumar dos variables

```
#include <stdio.h>
main ()
{
    int nume1,nume2,suma=0;
    printf ("Suma en C.\n");
    printf ("Escriba el primer numero:");
    scanf("%d",&nume1);
    printf ("Escriba el segundo numero:");
    scanf("%d",&nume2);
    suma=nume1+nume2;
    printf ("%d mas %d son %d",nume1,
            nume2,suma);
    printf("\n");
    printf ("Pulsa RETURN para terminar");
    scanf("%d"); //o getch()
}
```

Cálculo de la longitud y área del círculo

```
#include <stdio.h>           //incluye el archivo librería stdio
main (void)
{
    const float pi = 3.141592; // declara y asigna constante flotante Pi
                                //o puedes usar la constante M_PI si incluyes la librería <math.h>
    float radio, longitud, area; //declara 3 variables flotantes
    printf("Escribe el radio: ");
    scanf("%f",&radio); // pide el radio
    longitud = 2 * pi * radio; // o 2*M_PI*radio
    area = pi * radio * radio; // o 2*M_PI*radio
    printf ("Calculo de la longitud y area de un círculo de radio %g:\n", radio);
    printf ("LONGITUD: %f\n", longitud);
    printf ("AREA : %f\n", area);
    getch ();
}
```

Problemas propuestos. (Intentar hacer antes sin mirar el código).

1. a. Realizar la carga del lado de un cuadrado, mostrar por pantalla el perímetro y el área del mismo (Perímetro = valor del lado por cuatro ; área = lado al cuadrado)

```
#include <stdio.h>           //incluye el archivo cabecera sdttdio.h
main ()
{
    int lado, perimetro, area; //declaramos tres variables numéricas enteras
    printf("Introduzca el lado del cuadrado: ") ;
    scanf("%d",&lado); //pedimos el lado en formato entero decimal
    perimetro=lado * 4; //calculamos el perimetro
    area=lado*lado; //calculamos el área
    printf("El perimetro del cuadrado es: %d y su area de %d",perimetro, area);
    scanf("%d"); //pausa
}
```

1. b. *Ejercicio propuesto*: Calcular el área de un triángulo rectángulo, su perímetro y su hipotenusa.
2. Conversor de horas a minutos: Pedir las horas y mostrar los minutos:

```
#include <stdio.h>           ///incluye el archivo cabecera sdttdio.h
main ()
{
    int horas, minutos; //se declaran dos variables num. enteras
    horas =0; //asignamos un valor inicial a las horas
    printf ("Escribe las horas: ");
    scanf("%d",&horas); //pedimos las horas
    minutos = 60 * horas; //calculamos los minutos
    printf ("Hay %d minutos en %d horas.", minutos, horas); //mostramos en consola
    scanf("%d"); //pausa
}
```

3. Conversor de grados Fahrenheit a Celsius

```
#include <stdio.h>           //incluye el archivo cabecera sdttdio.h
main() //función principal
{
    float fahr, celsius; //se declaran dos variables decimales flotante
    printf("Grados fahrenheit :"); //muestra la pregunta
    scanf("%f", &fahr); //recoge el valor y lo pone en fahr
    celsius = (5.0 / 9.0) * (fahr - 32); //calcula los grados en celsius
    printf("%6.2f grados fahrenheit equivalen a %6.2f celsius\n", fahr, celsius);
    scanf("%d");
}
```

Ayuda en el planteamiento del algoritmo:

- **Pseudocódigo:** Falso código que describe un algoritmo de programación de forma informal.
- **Diagrama de flujo o flujograma:** Representación gráfica del proceso o algoritmo

Ejercicio de estructura secuencial: NotaMedia

Ejercicio para calcular y mostrar la nota media o promedio del alumno.

Enunciado: Se necesita un sistema que pida tres calificaciones parciales y muestre su promedio.

Guarda este proyecto en tu carpeta con el nombre: **NotaMedia** Lo necesitarás después.

➔ 1 En Pseudocódigo:

Pseudocodi castellano modo Uni

Algoritmo NotaMedia

variables

media, nota1, nota2, nota3 : real

fin variables

Inicio

Escribir ("Nota del primer parcial:")

Leer (nota1)

Escribir ("Nota del segundo parcial:")

Leer (nota2)

Escribir ("Nota del tercer parcial:")

Leer (nota3)

media := (nota1+nota2+nota3) / 3 //->según norma 2

o

media ← (nota1+nota2+nota3) / 3 //->según norma 3

Escribir ("Tu nota es:", media)

Fin algoritmo

Pseudocodi en Català modo URV

Algorisme NotaMedia es

Var

mitja, nota1, nota2, nota3 : real

Fvar

Inici

Escriure ("Nota primer parcial:");

Llegir(nota1);

Escriure ("Nota segon parcial:");

Llegir(nota2);

Escriure ("Nota tercer parcial:");

Llegir(nota3);

mitja := (nota1+nota2+nota3) / 3

Escriure ("Tu nota es:", mitja):

Falgorisme

➔ 3 En Lenguaje C

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

```
int main()
```

```
{
```

```
float nota1, nota2, nota3, media ;
```

```
printf("\n Dime tu primera nota: ");
```

```
scanf("%f",&nota1); //coge la nota1
```

```
printf("\n Dime tu segunda nota: ");
```

```
scanf("%f",&nota2); // coge la nota2
```

```
printf("\n Dime tu tercera nota: ");
```

```
scanf("%f",&nota3); // coge la nota3
```

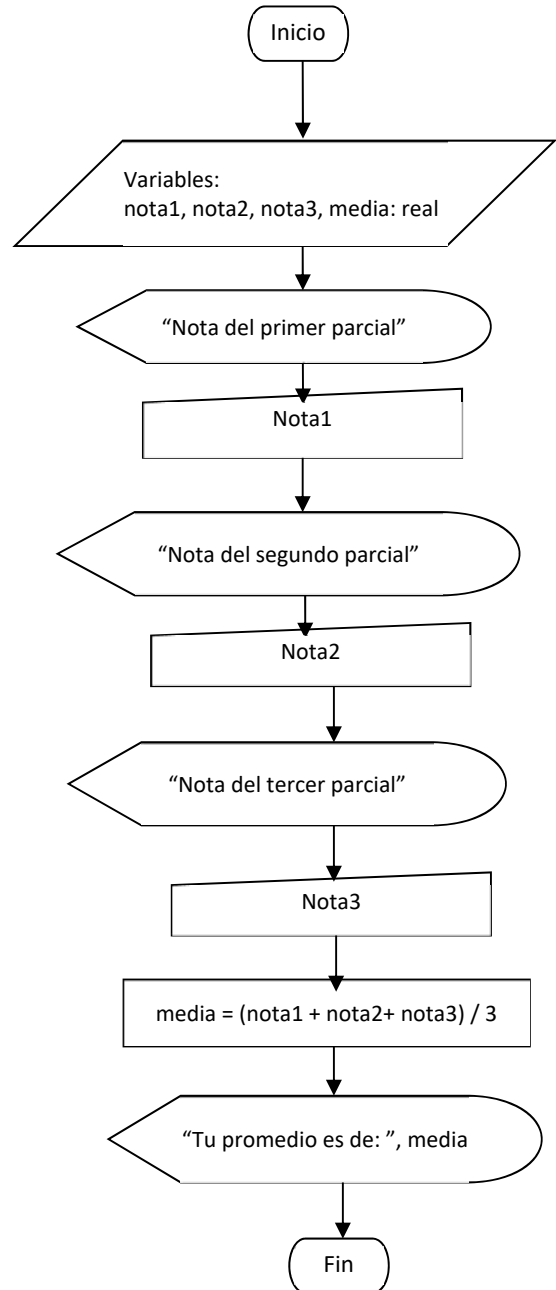
```
media=(nota1+nota2+nota3)/3; //calcula la media
```

```
printf("\n Tu promedio es de %.1f", media);
```

```
scanf("%s"); // pausa de consola
```

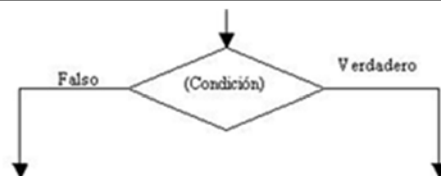
```
}
```

➔ 2 En Diagrama de flujo:



Estructura condicional de bifurcación

El programa realiza una u otra opción según una comparación condicional. Su instrucción base en C : **if (comparación) rama verdadera else rama falsa**
 Las bifurcaciones pueden ser simples, compuestas, anidadas o múltiples.



Comparadores lógicos:

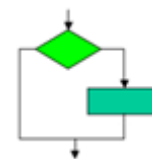
Igual: == Mayor: > menor: < mayor o igual que: >= menor o igual que: <= distinto: !=
 and se cumple a y b && or: se cumple a o b || not: not True es False

Ejemplo:

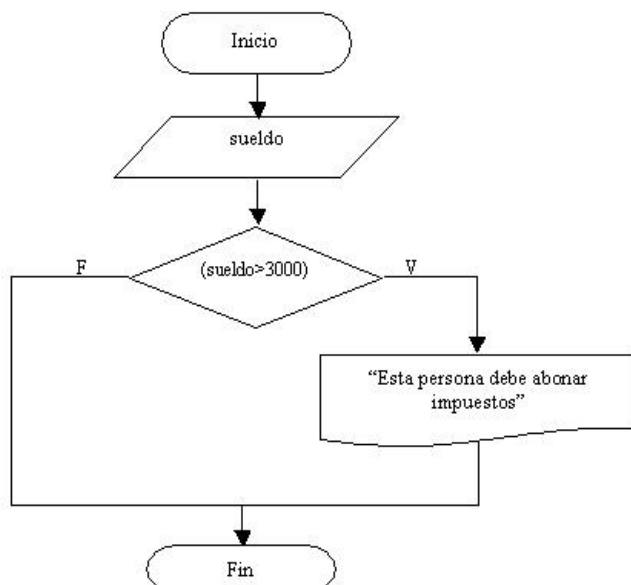
```
#include <stdio.h>

int main()
{
    char nombre[20]; //crea la variable nombre
    int edad=0; //crea la variable edad
    printf("Hola, dime tu nombre: "); //pide el nombre (una palabra)
    scanf("%s",&nombre); //coge el nombre y lo guarda en la variable nombre
    printf("Encantado, %s", nombre);
    printf("\n Dime cuantos años tienes: "); //pide la edad
    scanf("%d",&edad); //coge la edad y la guarda en la variable edad
    if (edad>=18) //si se cumple la condición ... ( no poner ; )
    {
        printf("%s, eres mayor de edad", nombre );
    }
    else //si no se cumple...
    {
        printf("%s, eres menor de edad", nombre );
    }
    printf("\n Pulsa Intro para salir");
    scanf("%s"); // pausa de consola (return 0 o getch )
}
```

Estructura condicional simple: Si sólo hay actuación por una rama de la condición



Ejercicio: Ingresar el sueldo de una persona, si supera los 3000 € mostrar un mensaje en pantalla indicando que debe abonar impuestos.



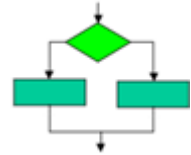
```
Programación en pseudocodi
Algorisme Sueldo es
var
    Sueldo: enter;
fvar
inici
    escriure("Ingrese el sueldo");
    llegirEnter(sueldo);
    si (sueldo>3000) llavors
        escriure("Debe abonar impuestos");
    fsi
falgorisme

Programación en C o C++
#include <stdio.h>
main ()
{
    float sueldo;
    printf("Ingrese el sueldo:");
    scanf("%f",&sueldo);
    if (sueldo>3000)
    {
        printf("Debe abonar impuestos");
    }
    scanf("%d");
}

Programación en C#
float sueldo;
string linea;
Console.Write("Ingrese el sueldo:");
linea=Console.ReadLine();
sueldo=float.Parse(linea);
if (sueldo>3000)
{
    Console.Write("Debe abonar impuestos");
}
Console.ReadKey();
```

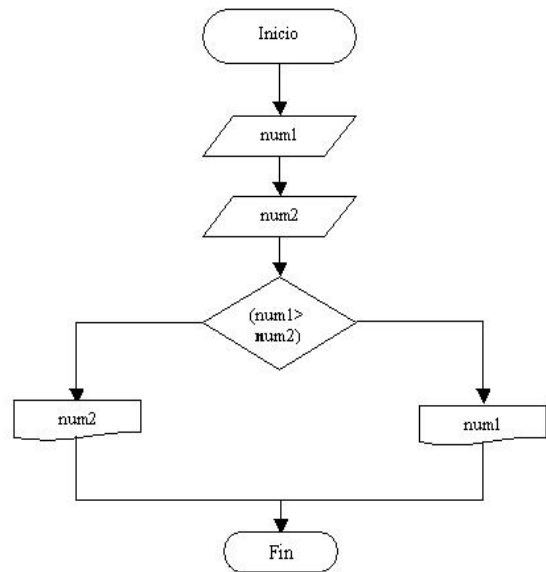
Estructura condicional compuesta.

Cuando tenemos actividades por las dos ramas de la condición, pero *nunca* por ambas a la vez. (Si sólo hay una instrucción en la rama, no es necesario englobar esta entre llaves)



Ejemplo 1: Realizar un programa que solicite dos números distintos y muestre por pantalla el mayor de ellos.

```
#include <stdio.h>
main ()
{
    int num1, num2;
    printf("Primer valor:");
    scanf("%d",&num1);
    printf("Segundo valor:");
    scanf("%d",&num2);
    printf("El número mayor es el ");
    if (num1 > num2)          //-> no poner el ;
    {                          //-> opcional
        printf("%d",num1);
    }
    else                      //-> opcional
    {
        printf("%d",num2);    //-> opcional
    }
    scanf("%d");             //-> opcional
}
```



Problemas propuestos

1.- Pedir dos números y los divide, pero evalúa antes si el denominador es cero.

En Lenguaje C /* División */

```
#include <stdio.h>
main ()
{
    int num1, num2;
    printf("\nPrimer número");
    scanf("%d",&num1);
    printf("\nSegundo número ");
    scanf("%d",&num2);
    if (num2 ==0)
        printf("\n No divisible por cero");
    else
        printf("\n Respuesta: %d",num1/num2);
}
```

Algorisme División es

```
var
    num1,num2: enter;
fvar
    inici
    escriure("Primer número");
    llegirEnter(num1);
    escriure("Segunado número");
    llegirEnter(num2);
    si (num2=0) llavors
        escriure("No divisible por 0");
    sino
        escriure("Respuesta:",num1/num2);
falgorisme
```

2.- Mostrar un mensaje indicando si un número introducido por teclado tiene uno o dos dígitos.

(Tiene dos dígitos si es mayor a 9)

En Lenguaje C

```
#include <stdio.h>
main()
{
    int num;
    printf("Introduzca un valor entero de 1 o 2 dígitos:");
    scanf("%d",&num);
    if (num<10)
    {
        printf("El numero %d tiene un dígito",num);
    }
    else
    {
        printf("El numero %d tiene dos dígitos",num);
    }
    scanf("%d");
}
```

En Lenguaje C#

```
int num;
string linea;
Console.WriteLine("Ingrese un valor entero de 1 o 2 dígitos:");
linea=Console.ReadLine();
num=int.Parse(linea);
if (num<10)
{
    Console.WriteLine("Tiene un dígito");
}
else
{
    Console.WriteLine("Tiene dos dígitos");
}
Console.ReadKey();
```

3.- Pedir un número entero y mostrar si es par o impar.

En Lenguaje C

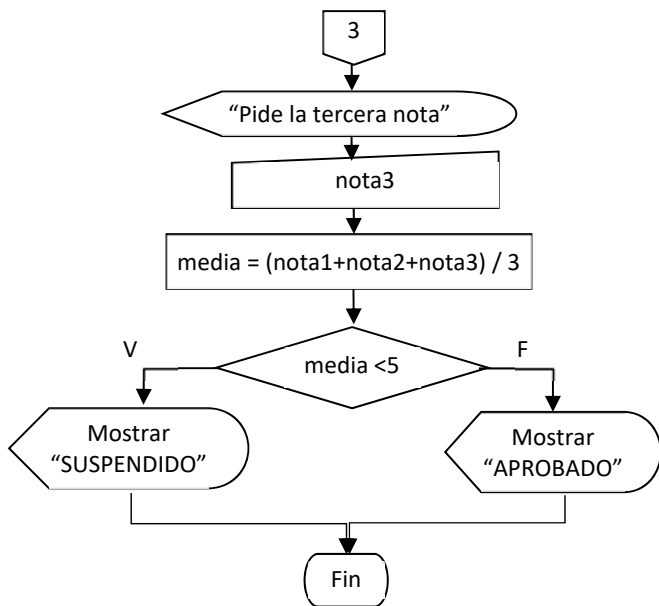
```
#include <stdio.h>
main ()
{
    int num;          //variable numérica entera
    printf("Entra un numero entero:");
    scanf("%d",&num);
    if (fmod(num,2)==0) //fmod función retorna
    resto
        printf("Es par");
    else
        printf("Es impar");
}
```

En pseudocodi

```
algorisme parell es
var
    num : enter
fvar
    EscriureMissatge("Entra un numero enter")
    LlegirEnter(num)
    Si num mod 2 = 0 // mod reste de le divisió
    llavors EscriureMissatge("Es parell")
    sino EscriureMissatge("Es senar")
fsi
falgorisme
```

Variante del ejercicio anterior NotaMedia: Abre el ejercicio anterior: **NotaMedia**

Añade el valor de aprobado o no aprobado si la media es 5 o superior. Guarda este ejercicio con el nombre: **NotaMedia2**



En Diagrama N-S: NotaMedia2

Inicio	
Variables: media, nota1, nota2, nota3 : real	
Escribir "Nota primer parcial:"	
Leer nota1	
Escribir "Nota segundo parcial:"	
Leer nota2	
Escribir "Nota tercer parcial:"	
Leer nota3	
media=(nota1+nota2+nota3) / 3	
media < 5	
Verdadero	Falso
Escribir "Promedio es:", media, " estas SUSPENDIDO"	Escribir "Promedio es:", media, " estas APROBADO"
Fin	

<p>En Lenguaje C. NotaMedia2</p> <pre> #include <stdio.h> int main() { float nota1, nota2, nota3, media ; char nombre[10]; printf("CALCULADOR DE NOTA MEDIA"); printf("\n Nota del primer parcial: "); scanf("%f",&nota1); //coge la nota1 printf("\n Nota del segundo parcial: "); scanf("%f",&nota2); //coge la nota2 printf("\n Nota del tercer parcial: "); scanf("%f",&nota3); //coge la nota3 media=(nota1+nota2+nota3)/3; printf("\n El promedio es %.1f", media); if (media<5) // no poner el ; { printf(" Estas suspendido"); } else { printf(" Estas aprobado"); } scanf("%s"); // pausa de consola } </pre>	<p>En Pseudocodi NotaMedia2</p> <pre> algorisme mitjana es var nota1, nota2, nota3, media ; reals fvar inici escriure("Nota del primer parcial: "); llegir(nota1); escriure("Nota del segon parcial: "); llegir(nota2); escriure("Nota del tercer parcial: "); llegir(nota3); media:=(nota1+nota2+nota3)/3; escriure("Promedio: ", media); si (media >= 5) llavors escriure ("Estas suspendido") sino escriure ("Estas aprobado") fsi falgorisme </pre>
---	---

Comparadores relacionales

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

Operadores	En pseudocodigo	en C
De asignación:	:= o ←	= +=, -=, *=
De comparación:	=	==
Relacionales:	>= ; <= ; ≠	>= ; <= ; !=
Aritméticos:	+, -, *, /, %(residuo)	+, -, *, /, %(residuo)
Arit. Unarios:		++n, --n, n++, n--
Lógicos:	i, o, no	&&, , !

Comparadores de texto:

La más utilizada **strcmp** aunque existen diversas funciones comparadoras de texto en la librería *string*. (veremos más adelante)

Sintaxis: **strcmp**(cadena1, cadena2) -> devuelve 0 si ambas cadenas son iguales.

Ejemplo: if (strcmp(clave, "ofimega") == 0) printf("correcto");

Ejercicios Pedir la clave: Comprueba y guarda estos ejercicios con el nombre: **PedirClave1.C y PedirClave2.C**

<pre> #include <stdio.h> main () { int pin; printf("Escribe el pin de acceso: "); scanf("%d",&pin); if (pin==1234) printf("correcto"); else printf ("incorrecto"); scanf("%d"); } </pre>	<pre> #include <stdio.h> main () { char clave[7]; printf("Escribe la palabra de acceso: "); scanf("%s",&clave); if(strcmp(clave, "ofimega") == 0) //strcmp -> compara texto printf("correcto"); else printf ("incorrecto"); scanf("%d"); } </pre>
--	--

El operador ?

Reemplaza la sentencia *if-else* y ahorra líneas.

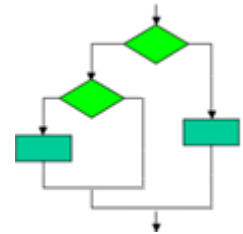
Ejemplo con if-else

```
if (n == 1)
    printf ("Mensaje 1");
else
    printf ("Mensaje 2");
```

Ejemplo con ?

```
n == 1 ?
    printf ("Mensaje 1");
    printf ("Mensaje 2");
```

```
// ----- Variante pedirClave 2 -----
#include <stdio.h>
main ()
{
    int pin;
    printf("Numero de PIN: ");
    scanf("%d",&pin);
    pin == 1234 ? printf ("Correcto"):printf ("Incorrecto");
    scanf("%d");
}
```



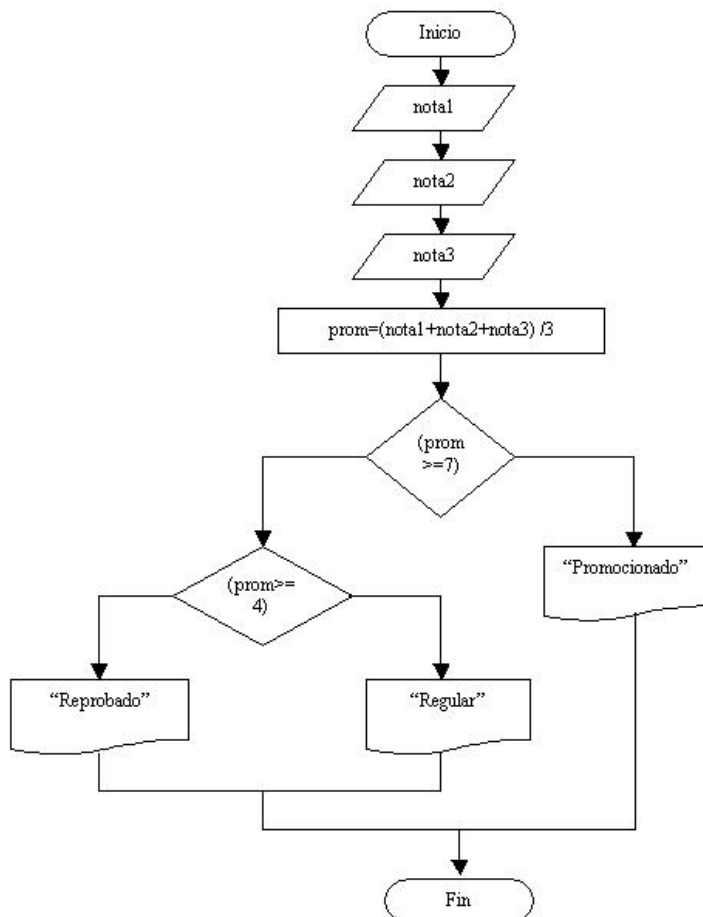
Estructuras condicionales anidadas

Cuando dentro de una estructura condicional hay otra estructura condicional.

Problema 4:

Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:

- ▶ Si el promedio es ≥ 7 mostrar "Notable".
- ▶ Si el promedio es ≥ 4 y < 7 mostrar "Regular".
- ▶ Si el promedio es < 4 mostrar "Suspendido".



```
#include <stdio.h>
main ()
{
    int nota1,nota2,nota3;
    printf("Nota 1er parcial:");
    scanf("%d",&nota1);
    printf("Nota 2o parcial:");
    scanf("%d",&nota2);
    printf("Nota 3er parcial:");
    scanf("%d",&nota3);
    int promedio=(nota1 + nota2 + nota3) / 3;
    if (promedio>=7)
    {
        printf("Notable");
    }
    else
    {
        if (promedio>=4)
        {
            printf("Regular");
        }
        else
        {
            printf("Suspendido");
        }
    }
    scanf("%d");
}
```

Problema 5: Pedir tres números y mostrar el mayor.

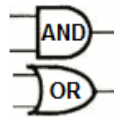
(ejemplo en lenguaje C)	(ejemplo en lenguaje C#)
<pre>#include <stdio.h> main () { int num1,num2,num3; printf("Numero 1:"); scanf("%d",&num1); printf("Numero 2:"); scanf("%d",&num2); printf("Numero 3:"); scanf("%d",&num3); if (num1>num2) { if (num1>num3) { printf("%d",num1); } else { printf("%d",num3); } } else { if (num2>num3) { printf("%d",num2); } else { printf("%d",num3); } } scanf("%d"); }</pre>	<pre>int num1,num2,num3; string linea; Console.Write("Ingrese primer valor:"); linea = Console.ReadLine(); num1=int.Parse(linea); Console.Write("Ingrese segunda valor:"); linea = Console.ReadLine(); num2 = int.Parse(linea); Console.Write("Ingrese tercer valor:"); linea = Console.ReadLine(); num3 = int.Parse(linea); if (num1>num2) { if (num1>num3) { Console.Write(num1); } else { Console.Write(num3); } } else { if (num2>num3) { Console.Write(num2); } else { Console.Write(num3); } } Console.ReadKey();</pre>

Operadores lógicos

Se emplean en estructuras condicionales para agrupar varias condiciones simples. **&& = Y** **|| = O**

Operador && (Y): Se lo lee como "Y". Ejecuta la rama del verdadero si la Condición 1 es verdadera **Y** la condición 2 es verdadera (ambas deben ser verdaderas).

Operador || (O): Se lo lee como "O". Ejecuta la rama del Verdadero si la condición 1 es Verdadera **O** la condición 2 es Verdadera (con una sola condición verdadera ya es suficiente).

**Ejercicio 1:** Variación del ejercicio de pedir clave. Guárdalo con el nombre: **PedirClave3:**

```
#include <stdio.h>
main ()
{
  char nombreUsu[7];
  int pin;
  printf("Nombre de usuario: ");
  scanf("%s",&nombreUsu);
  printf("\nNumero de PIN: ");
  scanf("%d",&pin);
  if ((strcmp(nombreUsu, "ofimega") == 0) && (pin==1234))
    printf("correcto");
  else printf ("nombre de usuario o pin incorrectos");
  scanf("%d");
}
```

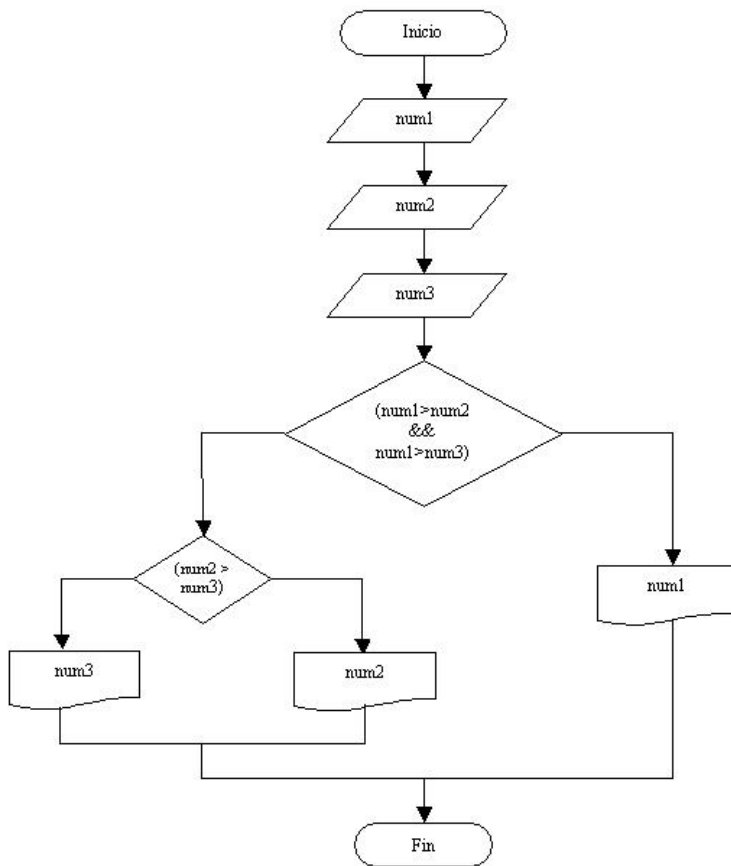


Ejercicio 1 Recupera el ejercicio de notas **NotaMedia2** y añade al final el recuadro contiguo. Si suspende alguna asignatura debe recuperarla de ambas maneras. Guarda en ejercicio con el nombre: **notaMedia3**

```
printf("\n"); // con operadores Y
if (nota1>=5 && nota2>=5 && nota3>=5)
{
  printf("No recuperas ninguna");
}
else
{
  printf("tienes que ir a recuperación");
}
```

```
printf("\n"); // con operadores O
if (nota1<5 || nota2<5 || nota3<5)
{
  printf("tienes que ir a recuperación");
}
else
{
  printf("No recuperas ninguna");
}
```

Problema 5: Repetir el programa para mostrar el número mayor utilizando el operador **&&**

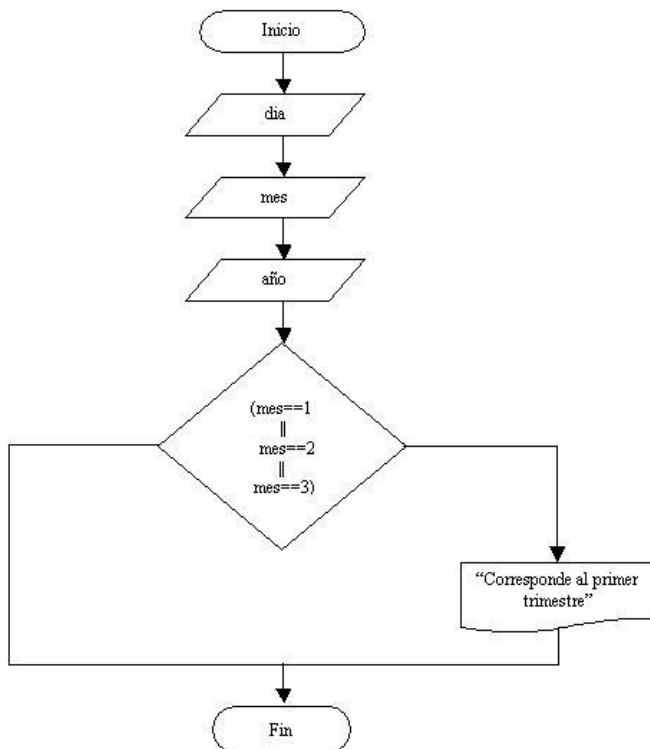


```

// P5. EL NUMERO MAYOR
//-----
#include <stdio.h>
main ()
{
  int num1,num2,num3;
  printf("Numero 1:");
  scanf("%d",&num1);
  printf("Numero 2:");
  scanf("%d",&num2);
  printf("Numero 3:");
  scanf("%d",&num3);
  if (num1>num2 && num1>num3)
  {
    printf("%d",num1);
  }
  else
  {
    if (num2>num3)
    {
      printf("%d",num2);
    }
    else
    {
      printf("%d",num3);
    }
  }
  scanf("%d");
}
  
```

Problema 6: Operador || (O)

Se carga una fecha (día, mes y año) por teclado. Mostrar un mensaje si corresponde al primer trimestre del año (enero, febrero o marzo) Cargar por teclado el valor numérico del día, mes y año. Ejemplo: día:10 mes:1 año:2010.



```

//P6. TRIMESTRE DEL AÑO
//-----
#include <stdio.h>
main ()
{
  int dia,mes,anyo;
  printf("Introduzca numero del dia:");
  scanf("%d",&dia);
  printf("Introduzca numero del mes:");
  scanf("%d",&mes);
  printf("Introduzca numero dle anyo:");
  scanf("%d",&ano);
  if (mes==1 || mes==2 || mes==3)
  {
    printf("Corresponde al primer trimestre");
  }
  else { printf("No es del primer trimestre");}
  scanf("%d");
}
  
```

Compara y discute los ejercicios PedirClave4 y PedirClave5:

<pre>#include <stdio.h> main () { char nombreUsu[6]; int pin; printf("Nombre de usuario: "); scanf("%s",&nombreUsu); printf("\nNumero de PIN: "); scanf("%d",&pin); if (strcmp(nombreUsu, "ofimega") == 0) { if (pin==1234) printf("correcto"); else printf ("Num de pin incorrecto"); } else { printf ("Nombre de usuario o pin incorrectos"); } scanf("%d"); }</pre>	<pre>#include <stdio.h> main () { char nombreUsu[6]; int pin; printf("Nombre de usuario: "); scanf("%s",&nombreUsu); if (strcmp(nombreUsu, "ofimega") == 0) { printf("\nNumero de PIN: "); scanf("%d",&pin); if (pin==1234) printf("correcto"); else printf ("Num de pin incorrecto"); } else { printf ("Nombre de usuario incorrecto"); } scanf("%d"); }</pre>
--	---

Ejercicio: Escribir un programa que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distintos a cero). Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si $x > 0$ Y $y > 0$, 2º Cuadrante: $x < 0$ Y $y > 0$, etc.)

<p><u>Pseudocodi</u> <u>Algorisme</u> quadrant <u>es</u> Var x, y; enter; Fvar Inici Escriure ("Ingreso coordenada x:"); Llegir(x); Escriure ("Ingreso coordenada y:"); Llegir(y); Si (x > 0 && y > 0) llavors Escriure("Se encuentra en el primer cuadrante"); Sino Si (x < 0 && y > 0) llavors Escriure("Se encuentra en el segundo cuadrante"); Sino Si (x > 0 && y < 0) Escriure("Se encuentra en el tercer cuadrante"); Sino Escriure ("Se encuentra en el cuarto cuadrante"); Fsi Fsi Fsi Falgorisme</p>	<pre>#include <stdio.h> main () { int x, y; printf("Ingreso coordenada x:"); scanf("%d",&x); printf("Ingreso coordenada y:"); scanf("%d",&y); if (x > 0 && y > 0) { printf("Se encuentra en el primer cuadrante"); } else { if (x < 0 && y > 0) { printf("Se encuentra en el segundo cuadrante"); } else { if (x > 0 && y < 0) { printf("Se encuentra en el tercer cuadrante"); } else { printf("Se encuentra en el cuarto cuadrante"); } } } scanf("%d"); // o system("pause"); }</pre>
---	--

Problemas propuestos

- Realizar un programa que pida cargar una fecha cualquiera, luego verificar si dicha fecha corresponde a Navidad.
- Se ingresan por teclado tres números, si todos los valores ingresados son menores a 10, imprimir en pantalla la leyenda "Todos los números son menores a diez" **y si** al menos uno de los valores ingresados es menor a 10, imprimir en pantalla la leyenda "Alguno de los números es menor a diez".
- Escribir un programa en el cual: dada una lista de tres valores numéricos distintos se calcule e informe su rango de variación (debe mostrar el mayor y el menor de ellos) y su diferencia
- De un operario se conoce su sueldo y los años de antigüedad. Se pide confeccionar un programa que lea los datos de entrada e informe:
 - Si el sueldo es inferior a 500 y su antigüedad es igual o superior a 10 años, otorgarle un aumento del 20 %, mostrar el sueldo a pagar.
 - Si el sueldo es inferior a 500 pero su antigüedad es menor a 10 años, otorgarle un aumento de 5 %.
 - Si el sueldo es mayor o igual a 500 mostrar el sueldo en pantalla sin cambios.

Condicionales múltiple switch ... case (opción)

Permite definir múltiples casos que puede llegar a cumplir una variable cualquiera, y qué acción tomar.

Sintaxis en C	Ejemplo en C	Ejemplo en pseudocódigo	Flujograma
<pre>switch (expressió) { case v1: instruccions; break; case v2: instruccions; break; . . . default: instruccions; break; }</pre>	<pre>int main() { int x; printf("Num. semana: "); scanf("%d",&x); switch (x) { case 1: printf("lunes"); break; case 2: printf("martes"); break; . . . default: printf("error"); break; } }</pre>	<pre>algorisme dia_semana es var x:enter; fvar inici escriure("Nombre:"); llegir(x); opció(x) 1: escriure("dilluns"); 2: escriure("dimarts"); . . . 7:escriure("diumenge"); en altre cas: escriure("incorrecte"); fopció falgorisme</pre>	

Ejercicio1: Califica la nota: suspenso, aprobado, notable.

Mediante if - else	Mediante switch - case
<pre>#include <stdio.h> main(void) { int nota; printf("introduce tu nota\n"); scanf("%i",&nota); if (nota >=7) { printf("Notable"); } else { if (nota >=5) { printf("Aprobado "); } else { printf("Suspendido"); } } scanf("%d"); }</pre>	<pre>#include <stdio.h> main(void) { int nota; printf("introduce tu nota\n"); scanf("%i",&nota); switch(nota) { case 6: //nótese que pueden estar desordenados case 5: printf("Aprobado\n"); break; case 7: case 8: case 9: case 10: printf("Notable\n"); break default: printf("Suspenso\n"); } system("pause"); }</pre>

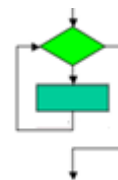
Ejercicio2: Crear un menú de funciones aritméticas

Lenguaje C	Pseudocodi
<pre>#include <stdio.h> main() { int opcion,a,b,resultado; printf("Ingrese un numero entero [a]: "); scanf("%d",&a); printf("Ingrese un numero entero [b]: "); scanf("%d",&b); printf("Escoja num menu\n"); printf("1. Sumar\n"); printf("2. Restar\n"); printf("3. Dividir\n"); printf("Elija una operacion: "); scanf("%d",&opcion); switch(opcion) { case 1: resultado=a+b; break; case 2: resultado=a-b; break; case 3: resultado=a*b; break; default: printf("Opcion no valida\n"); } printf("Resultado: %d,resultado); }</pre>	<pre>Algorisme menu és var a,b,resultado, opcion:enter; fvar inici escriure("Ingrese un numero entero [a]: "); llegir(a); escriure ("Ingrese un numero entero [b]: "); llegir (b); escriure("Escoja num menu"); escriure("1. Sumar"); escriure("2. Restar"); escriure("3. Multiplicar"); llegir(opcion); opció (opcion) 1: resultado=a+b; 2: resultado=a-b; 3: resultado=a*b; en altre cas:escriure("incorrecte"); fopció escriure ("Resultado: ",resultado); falgorisme</pre>

Bucles o estructuras repetitivas

Una estructura repetitiva ejecuta un conjunto o bloque de instrucciones varias veces. Normalmente:

- **Bucle infinito:** Se repite hasta que se anule la condición. Suele emplear el comando **While** o Do while
- **Bucle finito:** Se repite un número de veces determinado. Suele emplear el comando **For**



Estructura repetitiva: bucle while

Si la condición siempre es verdadera ejecutaría el bucle infinitamente.

Sintaxis en C	Ejemplo en C	Sintaxis pseudocódigo	Ejemplo en pseudocódigo
<pre>while (condición) { instrucciones; ... }</pre>	<pre>int main() { int i,num; i=0; printf("Num. de * : "); scanf("%d",&num); while(i!=num) { printf("*"); i++; } }</pre>	<p>Español: mientras (condicion) hacer <i>instrucciones;</i> fmientras</p> <p>Català mentre (condicio) fer <i>instruccions;</i> fmentre</p>	<p><i>algorisme asteriscs es</i> <i>var i,num:enter; fvar</i> <i>inici</i> <i>escriure("Num de * :");</i> <i>llegir(num);</i> <i>i:=0;</i> <i>mentre (i!=num) fer</i> <i>escriure("*");</i> <i>i:=i+1;</i> <i>fmentre</i> <i>falgorisme</i></p>

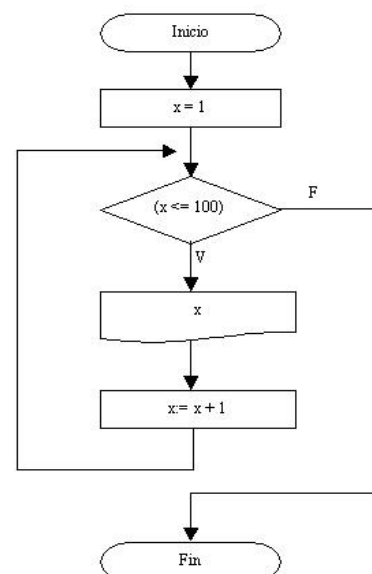
Instrucciones complementarias en C: **break:** termina el bucle antes.
continue: fuerza la siguiente iteración. **goto:** saltos incondicionales

Problema 7:

Realizar un programa que imprima en pantalla los números del 1 al 100.

- **Explicación:** While impone la siguiente condición (x <= 100), se lee MIENTRAS la variable x sea menor o igual a 100.
- Al ejecutarse la condición retorna VERDADERO si es menor o igual a 100. Al ser la condición verdadera se ejecuta el bucle while y la variable x se incrementa.
- Al llegar el contador a 100 retorna FALSO y se sale del bucle.
- La variable x debe inicializarse con algún valor antes de empezar a contar.

```
#include <stdio.h>
main ()
{
    int x;
    x = 1;
    while (x <= 100)
    {
        printf("%d", x);
        printf(" - ");
        x = x+1; //tambien: x++
    }
    scanf("%d");
}
```

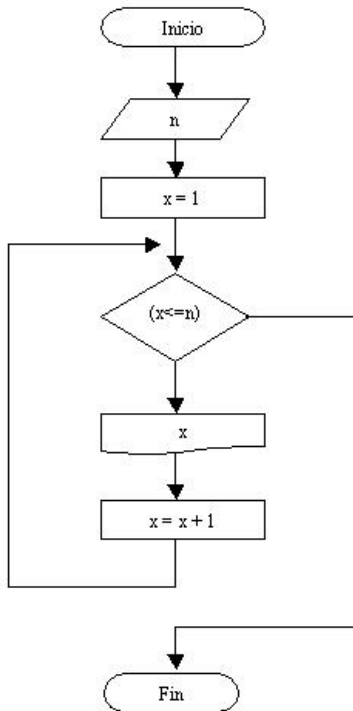


Problema PedirClave7: Pide la contraseña *infinitamente* hasta que sea correcta:
Problema PedirClave8: Pide *tres veces* la contraseña hasta que sea correcta:

<pre>//P7. Pedir la clave infinitamente //----- #include <stdio.h> main() { char contra[10]; printf("Ingrese la clave: "); scanf("%s",&contra); while (strcmp(contra, "ofimega") != 0) { printf("Incorrecto, repita de nuevo: "); scanf("%s",&contra); } }</pre> <div style="text-align: right;"> </div> <p>En pseudocodi: Algorisme Clau es var contra: enter; fvar inici escriure("Ingrese la clave"); llegir (num1); escriure("Segunado número"); llegirEnter(num2); si (num2=0) llavors escriure("No divisible por 0"); sino escriure("Respuesta:",num1/num2); falgorisme</p>	<pre>//P8. Pedir la clave 3 veces con while //----- #include <stdio.h> main () { char clave[10]; int x; x=3; while (x>0) { printf("Escribe la palabra de acceso: "); scanf("%s",&clave); if(strcmp(clave, "ofimega") == 0) //strcmp -> compara texto { printf("correcto"); x=0; } else { x--; //decrementa una unidad x=x-1 printf ("incorrecto, te quedan %d oportunidades\n",x); } } scanf("%d"); }</pre> <div style="text-align: right;"> </div>
--	---

Problema 8:

Escribir un programa que solicite la carga de un valor positivo y nos muestre desde 1 hasta el valor ingresado de uno en uno. Ejemplo: Si introducimos un 30 se debe mostrar en pantalla los números del 1 al 30.



```

#include <stdio.h>
main ()
{
  int n,x;

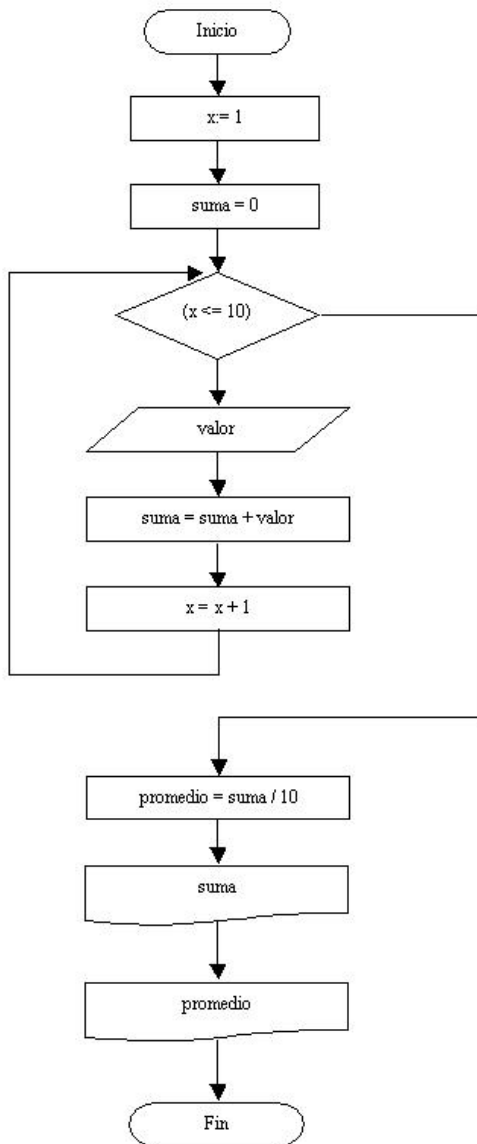
  printf("Ingrese numero final:");
  scanf("%d",&n);
  x=1;
  while (x<=n)
  {
    printf("%d",x);
    printf(" - ");
    x = x + 1;
  }
  scanf("%d");
}
  
```

En Pseudocodi
 Algorisme Contar es
 Var
 n, x: enter;
 Fvar
 Inici
 Escriure("Ingrese numero final:");
 Llegir(n);
 Mentre (x > n) fer
 Escriure(x);
 X:=x+1;
 Fmentre
 FAlgorisme

Los nombres de las variables n y x pueden ser palabras o letras (como en este caso) La variable x recibe el nombre de CONTADOR. Un contador es un tipo especial de variable que se incrementa o decremanta con valores constantes durante la ejecución del programa. El contador x nos indica en cada momento la cantidad de valores impresos en pantalla.

Problema 9:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.



```

#include <stdio.h>
main ()
{
  int x,suma,valor,promedio;
  x=1;
  suma=0;
  while (x<=10)
  {
    printf("Ingrese un valor:");
    scanf("%d",&valor);
    suma=suma+valor;
    x=x+1;
  }
  promedio=suma/10;
  printf("La suma de los 10 valores es: %d",suma);
  printf("El promedio es: %d",promedio);
}
  
```

En este problema, a semejanza de los anteriores, llevamos un CONTADOR llamado x que nos sirve para contar las veces que debe repetir el while. También aparece el concepto de ACUMULADOR (un acumulador es un tipo de variable que aumenta o disminuye con valores variables) Hemos dado el nombre de suma a nuestro acumulador. Cada ciclo que se repita la estructura repetitiva, la variable suma se incrementa con el contenido ingresado en la variable valor.

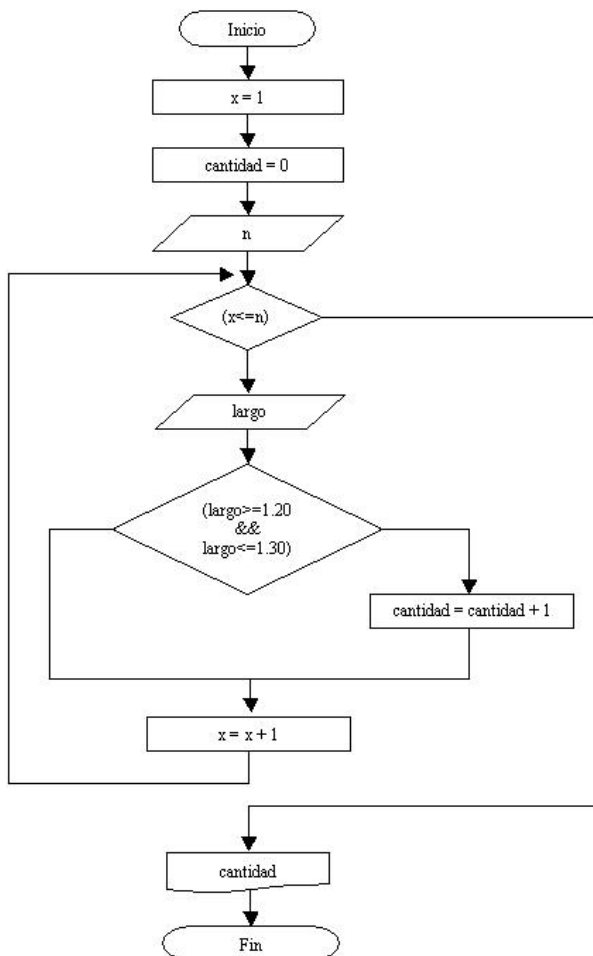
Ejercicio **Factorial** (versión básica). Multiplicar por el número decrementando. Ejemplo: $4! = 4 \cdot 3 \cdot 2 \cdot 1$

Flujograma factorial	pseudocodi factorial	llenguatge C factorial
<pre> graph TD inicio([inicio]) --> num[num, temp, dec] num --> dec[dec=dec-1] dec --> temp[temp=temp*dec] temp --> dec{dec>1} dec -- sí --> temp dec -- no --> mostar[Mostar temp] mostar --> fin([fin]) </pre>	<pre> var num: enter; temp: enter; dec: enter; fvar inici escriure("Donam el num"); llegir(num); dec:=num; temp:=num; mentre (dec>1) fer dec:=dec-1; temp:=temp*dec; fmentre escriure (temp); falgorisme </pre>	<pre> #include <stdio.h> int main() { int num,dec,temp; printf("Donam el num"); scanf("%d", &num); dec=num; temp=num; while (dec>1) { dec--; temp=temp*dec; } printf("Factorial %d",temp); return 0; } </pre>

Problema 10:

Una planta que fabrica perfiles de hierro posee un lote de n piezas.

Confeccionar un programa que pida ingresar por teclado la cantidad de piezas a procesar y luego ingrese la longitud de cada perfil; sabiendo que la pieza cuya longitud esté comprendida en el rango de 1,20 y 1,30 son aptas. Imprimir por pantalla la cantidad de piezas aptas que hay en el lote.



```

#include <stdio.h>
int main ()
{
  int x,cantidad,n;
  float largo;
  x=1;
  cantidad=0;
  printf("Cuantas piezas procesará:");
  scanf("%d",&n);
  while (x<=n)
  {
    printf("Ingrese la medida de la pieza:");
    scanf("%f",&largo);
    if (largo>=1.20 && largo<=1.30)
    {
      cantidad = cantidad +1;
    }
    x=x + 1;
  }
  printf("La cantidad de piezas aptas son: %d",cantidad);
}

```

Ejercicio para crear un menú:

```

#include <stdio.h> //ok edu

main () //crear un menu con bucle
{
  int x = 0;
  while (x != 3)
  {
    printf ("1.-Opcion menu1\n");
    printf ("2.-Opcion menu2\n");
    printf ("3.-Salir\n");
    printf ("Escoja opcion: ");
    scanf("%d",&x);
    switch (x)
    {
      case 1:printf ("Elegido opcion 1\n\n");
      break;
      case 2:printf ("Elegido Opcion 2\n\n");
    }
  }
}

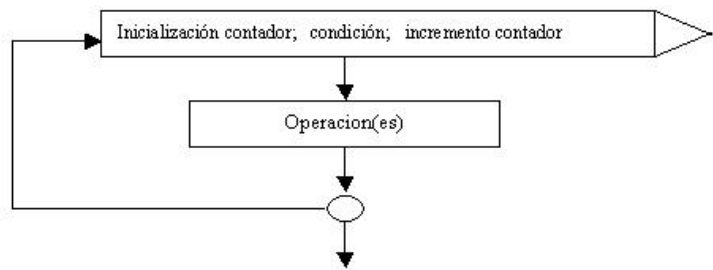
```

Estructura repetitiva for

Es un bucle finito donde CONOCEREMOS de antemano la cantidad de veces que queremos que el bloque se repita.

La estructura *for* puede usarse en cualquier situación repetitiva, porque en última instancia no es otra cosa que una estructura *while* generalizada.

Argumentos: Una variable que hace la función de *contador*. La "inicialización del contador". Una "condición" final y un "incremento del contador".



Sintaxis en C

```
for ( i=inicial ; i<final ; incrementa i )
{ ... }
```

En Pseudocódigo español:

```
para i en ... hacer
...
fpara
```

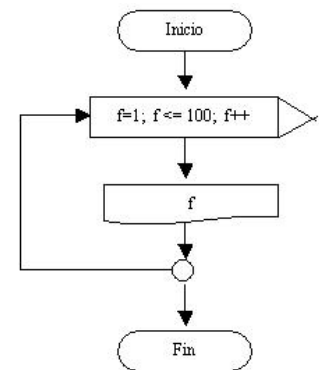
En Pseudocódigo català:

```
per i en ... fer o per( ... ) fer
...
fper
```

Problema 11:

Realizar un programa que imprima por pantalla los números del 1 al 100 utilizando la cláusula **For**

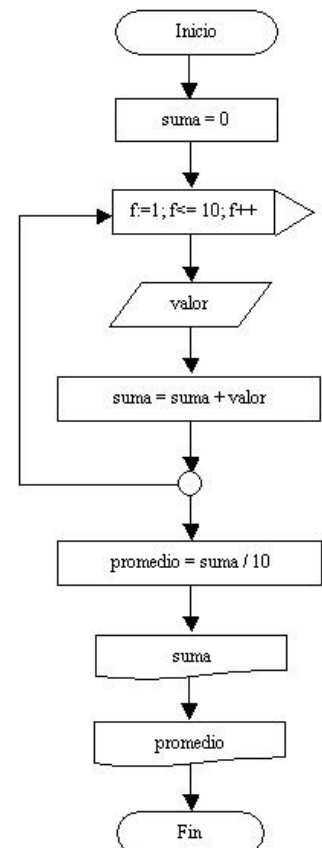
<pre> En Lenguaje C #include <stdio.h> main () { int f; for(f=1;f<=100;f++) { printf("\t %d",f); // \t:tab } scanf("%d"); } </pre>	<pre> En pseudocodi cat. algorisme els_cent_primers es var f:enter; fvar inici per(f:=1;f<=100;f++) fer escriure(f); fper falgorisme </pre>
---	--



Problema 12:

Desarrollar un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio. Utilizando la estructura **for**.

<pre> #include <stdio.h> main () { int suma,f,valor,promedio; suma=0; for(f=1;f<=10;f++) { printf("Valor %d/10: ",f); scanf("%d",&valor); suma=suma+valor; } printf("La suma es: %d", suma); printf("\n"); promedio=suma/10; printf("El promedio es: %d", promedio); scanf("%d"); } </pre>	<pre> algorisme demana_deu_vegades es var suma,f,valor,promedio:enter; fvar inici per(f:=1;f<=10;f++) fer escriure("Valor ",f); llegir(valor); suma=suma+valor; fper promedio=suma/10; escriure("La suma es ", suma); escriure("El promedio es ", promedio); falgorisme </pre>
---	---



Problema: Variante de hacer el factorial de un número mediante **WHILE** y mediante **FOR**

<pre> //----- Factorial con while ----- #include <stdio.h> main () { int x, veces; x = 1; printf ("Escribe el factorial:"); scanf("%d",&x); veces=x; while (veces >1) { veces--; x=x*veces; } printf("El factorial es: %d \n", x); system("pause"); // o scanf("%d") } </pre>	<pre> //----- Factorial con for ----- main () { int x, veces; x = 1; printf ("Escribe el factorial:"); scanf("%d",&x); veces=x; for (veces=x-1; veces>1; veces--) { x=x*veces; } printf("El factorial es: %d\n", x); system("pause"); // o scanf("%d") } </pre>
--	--

Problema: Pedir el número y hacer la tabla de multiplicar

En pseudocódigo	Diagrama de flujo: Tabla	En lenguaje C
<p>Pseudocódigo: Tabla</p> <p>Variables: tabla, contador, resultado : enteras</p> <ol style="list-style-type: none"> 1. Inicio 2. Escribir ("Número de la tabla") 3. Leer (tabla) 4. Hacer para contador = 1 hasta contador > 10 <ol style="list-style-type: none"> 4.1 resultado = tabla * contador 4.2 Escribir (tabla, "*", contador, "=", resultado) 4.3 contador = contador + 1 <p>Fin para Fin</p>	<pre> graph TD INICIO([INICIO]) --> Input[/Variables: tabla, contador, resultado/] Input --> Init[contador = 1] Init --> Loop[contador > 10] Loop -- F --> Calc[resultado = tabla * contador] Calc --> Show([Mostrar resultado]) Show --> Inc[contador = contador + 1] Inc --> Loop Loop -- V --> FIN([FIN]) </pre>	<pre> #include <stdio.h> int main() { int tabla, contador,resultado; printf("Dime el numero de la tabla: "); scanf("%d", &tabla); printf("\n La tabla del %d", tabla); for (contador=0;contador<11; contador++) { resultado=tabla*contador; printf("\n %d por %d = %d", tabla,contador,resultado) } } </pre>

Doble for: Permite un recorrido bidimensional utilizando dos contadores, uno para filas y otro para columnas. (ver más adelante)

Ejemplo: Mostrar las tablas de Multiplicar del 1 al 10:

*Nota: El formato %4d:
4 digitos numericos de derecha a izquierda*

```

#include <stdio.h>
int main(){
    int i,j;
    for(i=1;i<=10;i++)
    {
        for(j=1;j<=10;j++)
        {
            printf("%4d",i*j);
        }
        printf("\n"); //salto de linea
    }
    scanf("%d");}
}
            
```

Llamada recursiva a una función con for:

<pre> #include <stdio.h> long factorial(int numero); int main(int argc, char** argv) { int contador = 0; /* calcula el factorial de 0 a 10 */ for (contador = 0; contador <= 10; contador++) printf("%d! = %ld\n", contador, factorial(contador)); return 0; } /* función factorial <u>recursiva</u> */ long factorial(int numero) { if (numero <= 0) /* caso base */ return 1; /* casos bases: 0! = 1 y 1! = 1 */ else /* llamada recursiva */ return numero * factorial(numero - 1); /* llamada a la función factorial */ } </pre>	<pre> #include <stdio.h> long factorial(int numero); int main(int argc, char** argv) { int contador = 0; /* calcula el factorial de 0 a 10 */ for (contador = 0; contador <= 10; contador++) printf("%d! = %ld\n", contador, factorial(contador)); return 0; } /* función factorial <u>iterativa</u> */ long factorial(int numero) { long resultado = 1; int i = 0; /* declaracion de la función factorial iterativa */ for (i = numero; i >= 1; i--) resultado *= i; return resultado; } </pre>
--	--

Repaso y ampliación

Repaso y ampliación de operadores:

Operadores	Incrementales:	Otros operadores:
Aritméticos: + mas - menos * por / división % resto	++ Incremento: i=i+1 → i++	& Dirección a un objeto * Direcccionamiento indirecto [] Direcccionamiento indexado
Relacionales: >= mayor o igual != diferente de... == igual que...	-- Decremento: i=i-1 → i-	sizeof() Tamaño de la variable , reunir varias expresiones en una
Lógicos: NO negación: !, Y AND lógico: && O OR lógico:	+= -= /= Asignación compleja	

Tipos de datos:

Tipos de datos	Palabra	Ejemplo
Entero	Int	Int numero=0;
Real	Float	Float numero=12.2;
Carácter	Char	Char letra = 'a';
Cadena de carácter	Char	Char palabra[10] = "HOLA";
Lógico <stdbool.h>	Bool	Bool permiso=false;

Pseudocódigo:

Definición: (falso lenguaje) es una descripción informal de un proceso informático o algoritmo.

Sintaxis: Dependiendo del escritor, el pseudocódigo puede variar mucho en su estilo.

Generalmente:

<u>Asignación:</u> x←3 x=3 x:=3	<u>Condicional:</u> Si <i>condición</i> entonces Instrucciones; Sino entonces Instrucciones; Fin Si	<u>Bucle while:</u> Mientras <i>condición</i> Hacer Instrucciones Fin Mientras
<u>Bucle do - while:</u> Repetir Instrucciones Hasta que <i>condición</i>	<u>Múltiples casos:</u> Según <i>variable</i> hacer Caso valor Instrucción Caso valor Instrucción De otro modo Instrucción Fin Según	<u>Para (For):</u> Para i←0 hasta n paso 1 X ← Li Instrucciones Fin para

<u>Pseudocódigo estilo Fortran:</u>	<u>Pseudocódigo estilo Pascal:</u>	<u>Pseudocódigo estilo C:</u>
<pre> programa bizzbuzz hacer i = 1 hasta 100 establecer print_number a verdadero si i es divisible por 3 escribir "Bizz" establecer print_number a falso si i es divisible por 5 escribir "Buzz" establecer print_number a falso si print_number, escribir i escribir una nueva línea fin del hacer </pre>	<pre> procedimiento bizzbuzz para i := 1 hasta 100 hacer establecer print_number a verdadero; Si i es divisible por 3 entonces escribir "Bizz"; establecer print_number a falso; Si i es divisible por 5 entonces escribir "Buzz"; establecer print_number a falso; Si print_number, escribir i; escribir una nueva línea; fin </pre>	<pre> subproceso funcion bizzbuzz para (i <- 1; i<=100; i++) { establecer print_number a verdadero; Si i es divisible por 3 escribir "Bizz"; establecer print_number a falso; Si i es divisible por 5 escribir "Buzz"; establecer print_number a falso; Si print_number, escribir i; escribir una nueva línea; } </pre>

La función aleatoria:

En C, la función *Rand*, incluida en la librería *stdlib*, necesita de *Srand* (y la librería *time*) para iniciar la secuencia que genera un valor pseudo-aleatorio y que el inicio de este sea siempre diferente: `srand (time(NULL));`

Ejemplo: Genera un valor aleatorio dentro de un rango acotado, pedido al usuario.

<pre> #include <stdio.h> #include <stdlib.h> #include <time.h> int main() { int min, max, numero; srand(time(0)); // o srand (time(NULL)) printf("Introdueix minim:"); scanf("%d", &min); printf("Introdueix maxim:"); scanf("%d", &max); numero=rand()%(max-min+1)+min; printf("El valor creat es: %d ", numero); return (0); } </pre>	<pre> algorisme aleatoris es var min, max, numero: enter ; fvar inici num_aleatori Escriure("Introdueix minim:"); llegir(minim); Escriure("Introdueix maxim:"); llegir(maxim); Numero:=num_aleatori%(max-min+1)+min; Escriure("El valor creat es:", numero); falogrisme </pre>
---	--

Estructura repetitiva do while

Do and while... : Hazlo primero y luego repite si...

Se utiliza cuando, por lo menos una vez, debe ejecutar el bloque repetitivo.

Por eso la condición **while** está abajo del ciclo a repetir y el **do** arriba, a diferencia del while o del for que está en la parte superior.

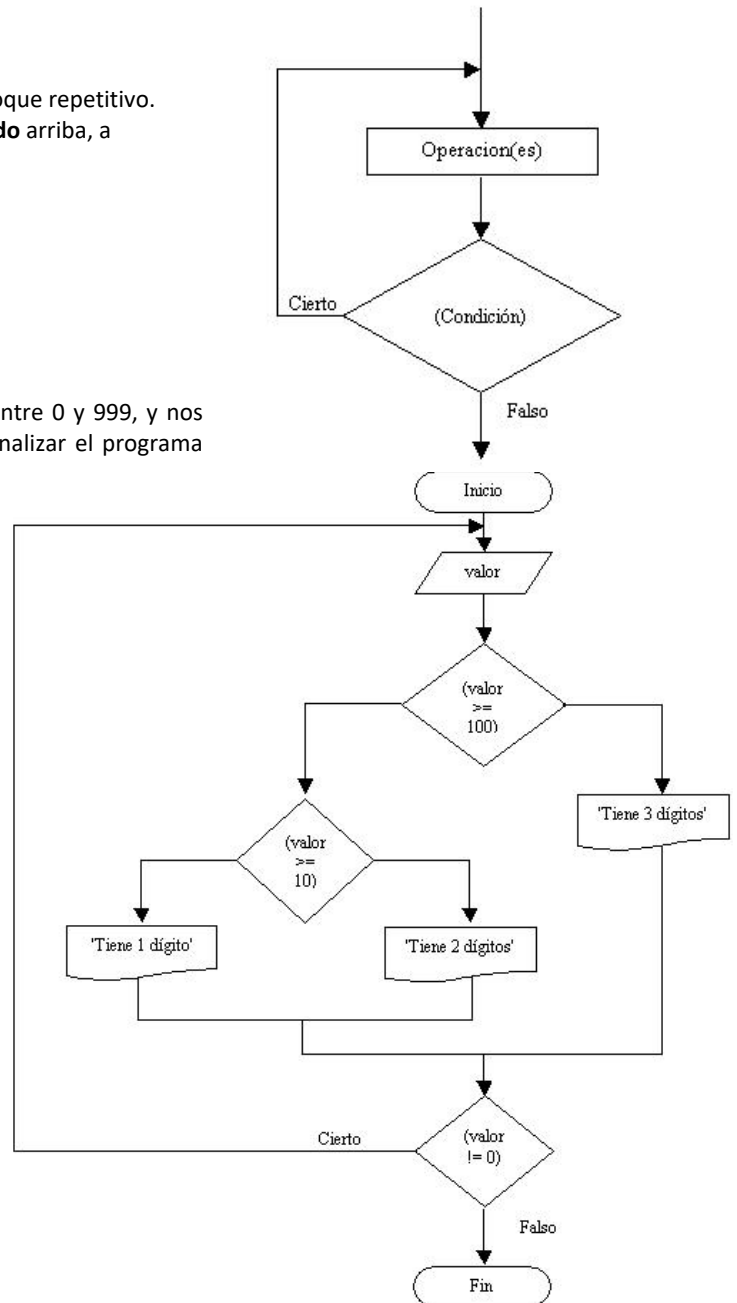
Sintaxis:

```
do
{
... instrucciones ...
}
while (condición);
```

Problema 13 do - while:

Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se escriba el valor 0.

```
#include <stdio.h>
main ()
{
int valor;
do {
printf("Ingrese un valor entre 0 y 999
(0 finaliza):");
scanf("%d",&valor);
if (valor>=100)
{
printf("Tiene 3 dígitos.");
}
else
{
if (valor>=10)
{
printf("Tiene 2 dígitos.");
}
else
{
printf("Tiene 1 dígito.");
}
}
} while (valor!=0);
}
```



Doble bucle: Podemos utilizar bucles anidados

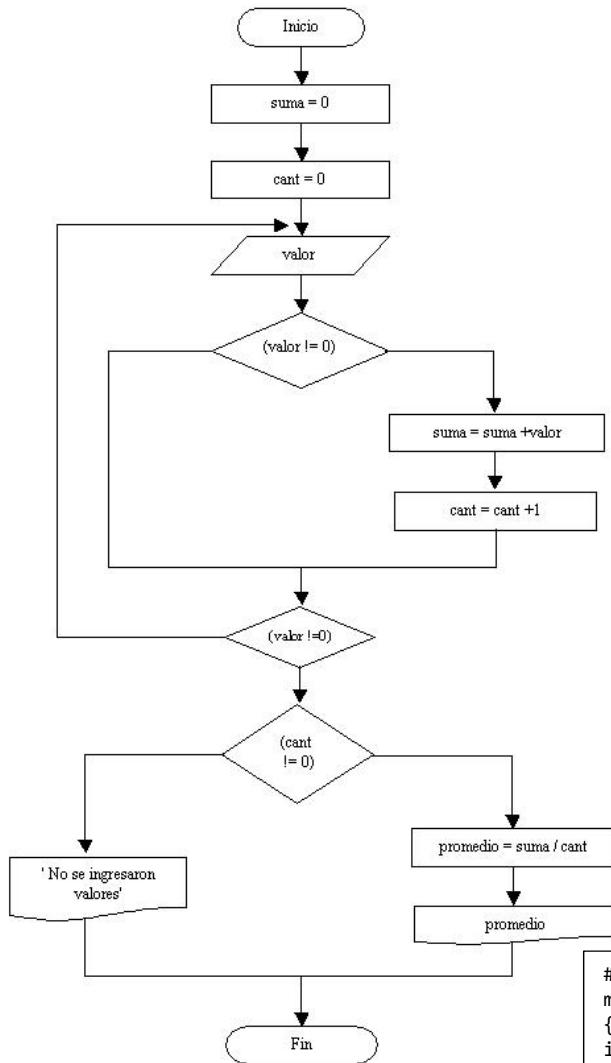
Ejemplo: Repetir 10 veces la cuenta del 1 al 10 con doble do – while y con doble for.

```
// -- Ejercicio en C don doble do-while --
#include <stdio.h>
main(void)
{
int i=1;
int j=1;
do
{
do
{
printf("%i ", j);
j = j+1;
}
while (j<=10);
j=1;
printf("\n");
i = i+1;
}
while (i<=10);
printf("\n");
system("PAUSE");
}
```

```
// -- Ejercicio en C con doble for --
#include <stdio.h>
main(void)
{
int i=1;
int j=1;
for (i=1;i<=10;i++)
{
for (j=1;j<=10;j++){
printf("%d ", j);
}
printf("\n");
}
system("PAUSE");
}
```

Problema 14 a:

Escribir un programa que solicite la carga de números por teclado, obtener su promedio. Finalizar la carga de valores cuando se cargue el valor 0.



Problema 14b:

Programa que solicite el nombre. Luego la carga de números por teclado, obtener su suma, promedio y suma de cuadrados. Finalizar la carga de valores cuando se cargue el valor 0.

Cuando la finalización depende de algún valor ingresado por el operador conviene el empleo de la estructura **do - while**, por lo menos se cargará un valor.

Definimos el contador *cant* que cuenta la cantidad de valores ingresados por el operador (no lo incrementa si ingresamos 0) El acumulador *suma* que almacena los valores ingresados.

```
#include <stdio.h>
main ()
{
    int suma,cant,valor,promedio;
    suma=0;
    cant=0;
    do {
        printf("Ingrese un valor (0 para finalizar):");
        scanf("%d",&valor);
        if (valor!=0)
        {
            suma=suma+valor;
            cant++;
        }
    } while (valor!=0);
    if (cant!=0)
    {
        promedio=suma/cant;
        printf("El promedio de los valores ingresados es:%d",promedio);
    }
    else {
        printf("No se ingresaron valores.");
    }
    scanf("%d");
}
```

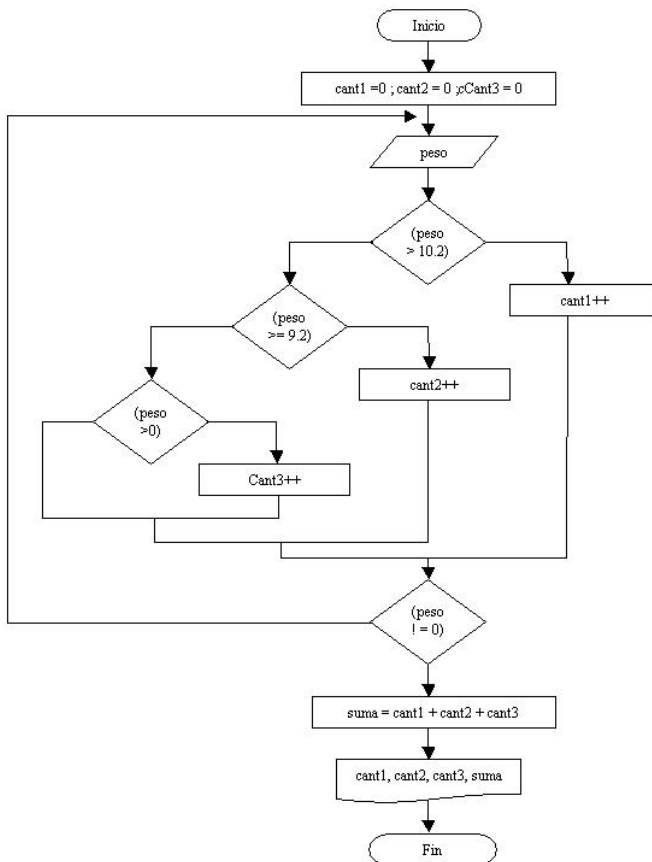
```
#include <stdio.h>
main ()
{
    int suma,veces,valor;
    float promedio,sumacuadrado;
    char nombre[20];
    suma=0; //inicio la variable
    veces=0; //inicio la variable
    printf("=====\n");
    printf("SUMA, PROMEDIO Y CUADRADO DE VALORES \n");
    printf("=====\n");
    printf("\n");
    printf("Introduce tu nombre: "); //FUERA DEL BUCLE
    scanf ("%s",&nombre);

    do { //empieza aqui el bucle
        printf("Hola %s, introduce un valor (escribe 0 para finalizar):",nombre);
        scanf("%d",&valor);
        if (valor!=0)
        {
            suma=suma+valor;
            sumacuadrado=sumacuadrado+(valor*valor);
            veces++;
        }
    } while (valor!=0); //si es cero se acaba el bucle
    if (veces!=0) //si es diferente de cero no han introducido nada
    {
        promedio=suma/veces;
        printf("La suma de los valores es: %d\n",suma);
        printf("El promedio de los valores es: %0.1f\n",promedio);
        printf("La suma de sus cuadrados es: %0.1f\n",sumacuadrado);
        printf("Adios, %s, recuerdos a Edu \n", nombre);
    }
    else {
        printf("No se ingresaron valores.");
    }
    scanf("%d");
}
```

Problema 15:

Realizar un programa que permita ingresar el peso (en kilogramos) de piezas. El proceso termina cuando ingresamos el valor 0. Se debe informar:

- ¿Cuántas piezas tienen un peso entre 9.8 Kg. y 10.2 Kg.?, cuántas con más de 10.2 Kg.? y cuántas con menos de 9.8 Kg.?
- La cantidad total de piezas procesadas.



```

#include <stdio.h>
main ()
{
int cant1,cant2,cant3,suma;
float peso;
cant1=0;
cant2=0;
cant3=0;
do {
printf("Peso de la pieza (0 finalizar):");
scanf("%d",&peso);
if (peso>10.2)
{
cant1++;
}
else
{
if (peso>=9.8)
{
cant2++;
}
}
else
{
if (peso>0)
{
cant3++;
}
} } }
while(peso!=0);
suma=cant1+cant2+cant3;
printf("Piezas aptas: %d\n",cant2);
printf("Piezas con un peso superior a 10.2:
%d\n",cant1);
printf("Piezas con un peso inferior a 9.8:
%d\n",cant3);
scanf("%d");
}
  
```

Problema 16

En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.

Se pide confeccionar un programa que lea los datos de las cuentas corrientes e informe:

- De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:

Estado de la cuenta

- 'Acreedor' si el saldo es >0.
- 'Deudor' si el saldo es <0.
- 'Nulo' si el saldo es =0.

- La suma total de los saldos acreedores.

```

int cuenta;
float saldo,suma;
sum_acum=0;
do {
printf("Ingrese número de cuenta:");
scanf("%d",&cuenta);
if (cuenta>=0)
{
printf("Ingrese saldo:");
scanf("%d",&saldo);
if (saldo>0)
{
printf("Saldo Acreedor.");
sum_acum=sum_acum+saldo;
}
else
{
printf("Saldo Deudor.");
}
}
} while(cuenta>=0);
printf("Total de saldos Acreedores: %d\n", sum_acum);
scanf("%d");
}
  
```

Tratamiento de secuencias de búsqueda

Secuencia de búsqueda mediante centinela:

Un centinela en un bucle de búsqueda es normalmente el primer valor no válido para que finalice el bucle.

Recorrido: Mismo tratamiento para todos los elementos de la secuencia. Termina al llegar al final de la secuencia
Ej.- Mostrar los elementos de una secuencia, sumar los números pares de una secuencia.

Búsqueda: Recorre sólo hasta encontrar un elemento buscado (o al llegar al final si no encontrado)

Ej.- Localizar el primer número que sea mayor que 1.000 -> Termina al encontrar la condición

Esquemas:

Recorrido:	Con centinela:	Búsqueda con centinela:
Si no sabemos cuántos elementos hay no podemos implementar con for <u>Esquema:</u> <ul style="list-style-type: none">▸ Inicialización▸ <i>Mientras</i> no se llegue al final de la secuencia:▸ Obtener el siguiente elemento▸ Procesar el elemento▸ Finalización	<ul style="list-style-type: none">▸ Inicialización▸ Obtener el primer elemento▸ <i>Mientras</i> no sea el centinela:▸ Procesar el elemento▸ Obtener el siguiente elemento▸ Finalización	<ul style="list-style-type: none">▸ Inicialización▸ Obtener el primer elemento▸ <i>Mientras</i> ni encontrado ni el centinela:▸ Obtener el siguiente elemento▸ Finalización (¿encontrado?)

Ejercicios

Ejercicio1. Cuenta los mayores de edad en una secuencia de edades introducidas por teclado.

En Pseudocodi

Algorisme majors es

```
Var
    Nume, majors: enter;
Fvar
Inici
    Nume:=100;
    Majors:=0;
    Mentre (nume>0) fer
        Escriure("Introduir edad (0 terminar):");
        Llegir (nume);
        Si (nume>18) llavors
            majors:=majors+1;
        fsi
    Fmentre
    Escriure("Hay %d majors de edad\n", majors);
FAlgorisme
```

En Lenquaje C:

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

main()
{
    int nume=100; //numero a pedir
    int majors=0; //contador par
    while (nume>0) //inicio bucle
    {printf("Introduir edad (0 o menor terminar):");
      scanf("%d",&nume);
      if (nume>18) majors++;
    }
    printf("Hay %d majors de edad\n", majors);
}
```

Ejercicio2. Cuenta el porcentaje de números pares e impares introducidos por teclado hasta poner un 0.

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

main()
{
    int nume=-1; //numero a pedir
    int par=0; //contador par
    int impar=0; //contador impar
    int i=0; //contador de nume
    while (nume!=0) //inicio bucle
    {printf("Introduir num. natural: ");
      scanf("%d",&nume);
      if (nume>=0) // si es positivo
      {
          i++;
          if (nume%2==0) par++; //resto de la división
          else impar++;
      }
      else printf("no es natural\n");//si es negativo
    } //fin del bucle while

    par=par*100/i; //pasa a %
    impar=impar*100/i; //pasa a %
    printf("El %d %% de nombres es par\n", par);
    printf("El %d %% de nombres es impar", impar);
}
```

(URV) Problemas de series con FOR en pseudocódigo y en C:

1.- Calcular el resultat de la següent sèrie on el valor de n el dóna l'usuari pel teclat. El valor d'n ha de ser positiu i major que 0.

$$sèrie = \sum_{(i=0)}^n \left(\frac{1}{2^i}\right) = \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \dots + \frac{1}{2^n}$$

Algoritme en pseudocodi	Programa en C
<pre> algorisme CalculIteracions es var base:=2, exponent, i: enters; suma:=1, sol:=1:real; \$Incloem el valor minim de la iteracio (i=0)-->1/(2^0)=1. \$ fvar inici escriure("Calcul de la suma d'(1/2^i)."); escriure("Introdueix el nombre d'n iteracions: "); llegir(exponent); si((exponent>0) o (exponent==0)) llavors per (i:=1; i<exponent; i++) fer sol:=sol*base; suma:=suma + (1/sol); fper escriure("La serie [n=]", exponent, "val= ", suma); sino escriure("ERROR"); fsi falgorisme </pre>	<pre> #include <stdio.h> #include <stdlib.h> int main() { int base=2, exponent, i; float sol=1, suma=1; // Incloem el valor minim per a i=0 --> 1/(2^0)=1 printf("Calcul de la suma d'(1/2^i).\n\n"); printf("\n\nIntrodueix el nombre d'n iteracions: "); scanf("%d",&exponent); if ((exponent>0) (exponent=0)){ for (i=1; i<exponent; i++){ sol=sol*base; suma = suma+(1/sol); } printf("La serie [n=%d] val = %f\n",exponent, suma); } else{ printf("ERROR"); } return 0; } </pre>

2.- Calcular el resultat de la següent sèrie on el valor de n el dóna l'usuari pel teclat:

$$\sum_{(i=0)}^n (-1)^{(i)} \left(\frac{1}{(i+1)}\right)$$

Algoritme en pseudocodi	Programa en C
<pre> algorisme serie1 es var signe, n, i: enters; suma: real; fvar inici escriure("Calcul de la serie 1"); escriure("Introdueix el nombre de vegades n: "); llegir(n); si (n>0) llavors per (i:=0; i<n; i++) fer si (i mod 2==0) llavors signe:=1 sino signe:=-1; suma:=suma + signe*(1/(i+1)); fper escriure("La suma de ", n, "interaccions val= ", suma); sino escriure("Nombre n incorrecte"); fsi falgorisme </pre>	<pre> #include <stdio.h> #include <stdlib.h> int main() { int signe, n, i; float suma=1; printf("Calcul de la serie 1\n"); printf("Introdueix el nombre de vegades n: "); scanf("%d",&n); if (n>0) { for (i=0; i<n; i++){ if (i % 2==0) signe=1; //->signe positiu si es par else signe=-1; //-> signe negatiu si es impar suma = suma + signe*(1/(i+1)); } printf("La suma de %d interaccions val %f", n,suma); } else{ printf("Nombre n=%d incorrecte ",n); } scanf("%d"); return 0; } </pre>

Ejercicios For – While en C++.

Escribir un programa que muestre una lista de números del 1 al 20, indicando a la derecha de cada uno si es divisible por 3 o n.

```
#include <iostream> // biblioteca para uso de cout
using namespace std;
int main() // función principal
{
    int i; // variable para bucle
    for(i = 1; i <= 20; i++) // bucle for de 1 a 20 {
        cout << i; // muestra el número
        if(i % 3 == 0)
            cout << " es múltiplo de 3"; // resto==0
        else cout << " no es múltiplo de 3"; //resto ≠ 0
        cout << endl; // cambio de línea
    }
    return 0;
}
```

Escribir el programa anterior, pero usando una **función** para verificar si el número es divisible por tres, y un bucle de tipo while.

// Este programa muestra una lista de números,
// indicando para cada uno si es o no múltiplo de 3.

```
#include <iostream> // biblioteca para uso de cout
using namespace std;

// Prototipos:
bool MultiploDeTres(int n);

int main() // función principal
{
    int i = 1; // variable para bucle

    while(i <= 20) // bucle hasta i igual a 20
    {
        cout << i; // muestra el número
        if(MultiploDeTres(i)) cout << " es múltiplo
de 3";
        else cout << " no es múltiplo de 3";
        cout << endl; // cambio de línea
        i++;
    }

    return 0;
}
```

// Función que devuelve verdadero si el parámetro 'n' es
// múltiplo de tres y falso si no lo es

```
bool MultiploDeTres(int n)
{
    if(n % 3) return false; else return true;
}
```

Escribir un programa que pida varios números, hasta que el usuario quiera terminar, y los descomponga en factores primos.

// Programa que descompone números en factores primos

```
#include <iostream> // biblioteca para uso de cout
using namespace std;

int main()
{
    int numero;
    int factor;
    char resp[12];

    do {
        cout << "Introduce un número entero: ";
        cin >> numero;
        factor = 2;
        while(numero >= factor*factor) {
            if(!(numero % factor)) {
                cout << factor << " * ";
                numero = numero / factor;
                continue;
            }
            if(factor == 2) factor++;
            else factor += 2;
        }
        cout << numero << endl;
        cout << "Descomponer otro número?: ";
        cin >> resp;
    } while(resp[0] == 's' || resp[0] == 'S');
    return 0;
}
```

OFIMEGA

Listas y cadenas de caracteres (arrays de una dimensión)

Un vector o arreglo es una lista de datos del mismo tipo.

Las cadenas son un tipo de vector, o arreglo que contienen una lista de caracteres.

Ejemplos: `int vector[3] = {1,2,3}` `int notas[3] = {5,7,9}` `char nombre[7] = {'O','F','I','M','E','G','A'}`

Caracteres y cadenas de caracteres

Para almacenar un texto, "string" o cadena de caracteres (por ejemplo un nombre de una persona) debemos definir un array o lista de una dimensión del tipo carácter, especificando como logitud de sus elementos el número de letras o caracteres máximo:

- En C y C++ se utiliza la variable `char`: `char nombre [longitud_máxima];` ejemplo: `char Saludo[5];` y se introduce el texto con `strcpy(nombre,"hola");`
- En C# o Cbuilder ya existe una variable del tipo `string`. Ejemplo: `string nombre1;` y se introduce en texto con `readline`: `nombre1=Console.ReadLine();`
- Una cadena de texto es un array cuyo último carácter es 0 o null (`\0`)
- Tamaño de una cadena: Si la cadena contiene espacios, se lee sólo hasta el primer espacio.
- En C no podemos asignar ni comparar un array `char` con el signo igual a una cadena de texto, → texto = "Hola"; -> incorrecto sino que utilizamos para asignar: `strcpy(texto,"hola")` o para comparar: `strcmp(texto,"hola")` de la librería `string.h`

h	o	l	a	\0
---	---	---	---	----

Problema 16:

Pedir el nombre y edad de dos personas y muestra el nombre de la persona con mayor edad.

```
Programa en C++
#include <stdio.h>
#include <string.h>
main ()
{
char nombre1[20],nombre2[20];
int edad1,edad2;
printf("Ingrese el nombre 1:");
scanf("%s",&nombre1);
printf("Edad de %s:", nombre1);
scanf("%d",&edad1);
printf("Ingrese el nombre 2:");
scanf("%s",&nombre2);
printf("Edad de %s:", nombre2);
scanf("%d",&edad2);
printf("La persona de mayor edad es:");
if (edad1>edad2)
{
printf("%s", nombre1);
}
else
{
printf("%s",nombre2);
}
scanf("%d");
}
```

Problema 17: Solicitar contraseña.

```
Programa en C
#include <string.h>
#include <stdio.h>
main()
{
char s[80];
puts("Clave :");
gets(s);
if((strcmp(s, "pepe")) == 0) //strcmp -> compara dos cadenas de texto
puts("Ok...");
else {
puts("Password equivocado");
}
}
```

```
#include <stdio.h>

main()
{
char nombre[] = "Ofimega";
printf( "Texto: %s\n", nombre );
printf( "Tamaño de la cadena: %i bytes\n", sizeof nombre );
}
```

Cadenas de caracteres string

Podemos leer una cadena entera (string) desde teclado con `gets` o mostrar con `puts` de la librería `stdio.h`

Estas funciones están en desuso

```
#include "stdio.h"
main()
{
char s[TM];
puts("Escribe hola");
gets(s);
printf("%s\n", s);
}
```

Podemos almacenar una variable de carácter sin limitación utilizando *

```
#include <stdio.h>
main ()
{
char *string = "OFIMEGA";
printf("%s\n", string);
scanf("%d");
}
```

Texto en C++ : Streams – Cin - Cout:

En C++ a diferencia de C las entradas y salidas se leen desde streams (canal, flujo o corriente)

Para su uso hay que incluir la librería `#include <iostream.h>` entonces se pueden introducir y sacar datos con `cin` y `cout`:

cin: Lee datos (por defecto del teclado)
cout: Imprime el texto entre comillas
>> : operador de inserción, acompaña a cin (entra texto)
<< : operador de extracción, acompaña a cout (sale texto)

```
#include <stdio.h>
#include <iostream.h>
main ()
{
    cout << "muestro un texto sin formato" << endl;
    scanf("%d");
}
```

Funciones de carácter en la librería "stdio.h":

- Función `getchar()` Lee un carácter desde el teclado; espera por el retorno de carro (enter).
- Función `getche()` Lee un carácter del teclado; no espera por retorno de carro.
- Función `getch()` Lee un carácter del teclado, no hace eco y no espera por retorno de carro (enter).
- Función `putchar()` Escribe un carácter en la pantalla.
- Función `gets()` Lee una cadena desde el teclado. Get string
- Función `puts()` Escribe una cadena en la pantalla. Put string
- Función `cprintf()` Escribe carácter formateado.
- Función `kbhit()` Espera por la pulsación de una tecla.
- Función `fgets()` Lee una cadena de tamaño específico y la almacena en una variable.
- Función `fputs()` Función contraria a `fgets()`.

Nota: El uso de "gets" está desaconsejado a partir del estándar C99 (de 1999), por tratarse de una orden poco segura. Esta orden ni siquiera existirá en los compiladores que siguen el estándar C11 (a partir de diciembre de 2011).

```
#include <stdio.h>
main()
{
    char carac;
    printf("Digita un caracter :");
    carac=getchar();
    printf("%c\n", putchar(carac));
    scanf("%d");
}
```

Funciones de carácter de la biblioteca "ctype.h"

- `isalnum()` Devuelve diferente de cero (0) si el carácter es una letra o un dígito.
- `isctrl()` :Devuelve diferente de cero (0) si el carácter es un carácter de control. Entre 0 ... 0x1F y 0x7F (DEL).
- `isdigit()` :Devuelve diferente de cero (0) si el carácter es un dígito.
- `isalpha()` : Devuelve diferente de cero si es un character alfanumérico.
- `isgraph()` : Devuelve diferente de cero si el carácter es imprimible y distinto de espacio.
- `islower()` :Devuelve diferente de cero si se trata de un carácter minúscula.
- `isprint()` : Devuelve diferente de cero si el carácter es imprimible incluyendo el espacio.
- `ispunct()` :Devuelve diferente de cero si el carácter es un carácter de puntuación (diferente a los alfabéticos y numéricos).
- `isspace()` :Devuelve diferente de cero si el carácter es un espacio.
- `isxdigit()` : Devuelve diferente de cero si el carácter es un dígito hexadecimal.
- `isupper()` :Devuelve diferente de cero si el carácter es mayúscula.

Convertir de mayúsculas a minúsculas o viceversa:

- `toupper()` -> convierte a mayúscula
- `tolower()` -> convierte a minúscula.

```
/** convierte un texto en mayúscula **/
#include <stdio.h>
#include <ctype.h>

main()
{
    char texto[80];
    int i;

    printf("Introduce una frase en minúscula :");
    gets(texto); //->permite varias palabras
    //scanf("%s",&texto); ->solo una palabra

    for(i=0; texto[i]; i++)
        printf("%c", toupper(texto[i]));

    scanf("%d");
}
```

En este ejercicio, restamos el nº de carácter de mayúsculas a minúscula en posiciones ASCII ya que las mayúsculas son los caracteres ASCII que van del 65(A) al 90(Z), las minúsculas van del 97(a) al 122(z): ejemplo: 'A' = 'a' + 64;

```
/** Devuelve el número de letra **/
#include <stdio.h>
#include <ctype.h>
int main()
{
    char letra; // variable letra del tipo caracter
    int num; // variable numérica entera
    printf("\n Lletra:");
    scanf("%c", &letra);
    // en código ASCII la A mayusc empieza en 64
    num=toupper(letra)-64; //-> orden ASCII
    printf("El numero de letra es la %d", num);
}
```

Funciones de cadena de la biblioteca <string.h>

Se hallan definidas en el archivo "string.h".

- › **strcpy()** :Copia el contenido de la cadena origen en la cadena destino. Formato : strcpy(destino, origen);
- › **strcat()** :Concatena dos cadenas. Formato : strcat(cadena1, cadena2); Concatena la cadena2 al final de la cadena1.
- › **strncat()**: Añade la cadena 2 al final de la cadena 1 a partir de una posición (cadena1,cadena2,sizeof n)
- › **strcmp()**: Compara dos cadenas y devuelve cero si son iguales. Formato : strcmp(cadena1, cadena2); Si cadena1 es mayor que cadena2 devuelve un valor mayor que cero. Si cadena1 es menor que cadena2 devuelve un valor negativo.
- › **strlen()** :Mide la longitud de una cadena en caracteres. Formato : strlen(cadena);
- › **strchr()** y **strstr()** : Permite comprobar si una cadena contiene un cierto texto.
- › **strtok()**: palabra = strtok(cadena, " "); Parte una cadena de texto por un carácter

Ejemplo strcpy:

```
#include "string.h"
main()
{
char a[80];
strcpy(a, "hola mundo");
printf("%s\n", a);
}
```

Ejemplo strcat:

```
#include <string.h>
#include <stdio.h>
main()
{
char a1[80], a2[40];
strcpy(a1, "ofimega");
strcpy(a2, " academias");
strcat(a1, a2);
puts(a1);
}
```

Ejemplo strcmp:

```
#include <string.h>
#include <stdio.h>
main()
{
char s[80];
puts("Clave :"); gets(s);
if((strcmp(s, "pepe")) == 0)
printf("Ok...");
else
{
printf("Password equivocado");
exit(0);
}
}
```

Ejemplo strlen:

```
#include <string.h>
#include <stdio.h>
main()
{
int s[80];
strcpy(s, "Hola");
printf("%d\n", strlen(s));
getch();
}
```

```
#include <stdio.h>
#include <string.h>
main()
{
char color[] = "rojo";
char grosor[] = "grosos";
char descripcion[1024];

strcpy(descripcion, "Lapiz color ");
strncat(descripcion, color, 1024);
strncat(descripcion, " de trazo ", 1024);
strncat(descripcion, grosor, 1024);
printf(descripcion);
// imprime "Lapiz color rojo de trazo grueso"
}
```

OFIMEGA

<p><i>Programa que muestre el alfabeto.</i> Para complicarlo un poco más, debe imprimir dos líneas, la primera en mayúsculas y la segunda en minúsculas.</p> <pre>// Muestra el alfabeto de mayúsculas y minúsculas #include <iostream> using namespace std; int main() { char a; // Variable auxiliar para los bucles // El bucle de las mayúsculas lo haremos con un while a = 'A'; while(a <= 'Z') cout << a++; cout << endl; // Cambio de línea // El bucle de las minúsculas lo haremos con un for for(a = 'a'; a <= 'z'; a++) cout << a; cout << endl; // Cambio de línea return 0; }</pre>	<p><i>Muestra el alfabeto mayúsculas/minúsculas alternativas:</i></p> <pre>// Muestra el alfabeto de mayúsculas y minúsculas: // AbCdEfGhIjKlMnOpQrStUvWxYz #include <iostream> #include <cctype> using namespace std; int main() { char alfabeto[27]; // Cadena que contendrá el alfabeto int i; // variable auxiliar para los bucles // Aunque podríamos haber iniciado alfabeto directamente, // lo haremos con un bucle for(i = 0; i < 26; i++) alfabeto[i] = 'a' + i; alfabeto[26] = 0; // No olvidemos poner el fin de cadena // Aplicamos el primer procedimiento si la posición es // par convertimos el carácter a minúscula, si es impar // a mayúscula for(i = 0; alfabeto[i]; i++) if(i % 2) alfabeto[i] = tolower(alfabeto[i]); else alfabeto[i] = toupper(alfabeto[i]); cout << alfabeto << endl; // Mostrar resultado // Aplicamos el segundo procedimiento si el carácter era // una mayúscula lo cambiamos a minúscula, y viceversa for(i = 0; alfabeto[i]; i++) if(isupper(alfabeto[i])) alfabeto[i] = tolower(alfabeto[i]); else alfabeto[i] = toupper(alfabeto[i]); cout << alfabeto << endl; // Mostrar resultado // Aplicamos el tercer procedimiento, pondremos los dos // primeros caracteres directamente a mayúsculas, y // recorreremos el resto de la cadena, si el carácter // dos posiciones a la izquierda es mayúscula cambiamos // el carácter actual a minúscula, y viceversa alfabeto[0] = 'A'; alfabeto[1] = 'B'; for(i = 2; alfabeto[i]; i++) if(isupper(alfabeto[i-2])) alfabeto[i] = tolower(alfabeto[i]); else alfabeto[i] = toupper(alfabeto[i]); // Mostrar resultado: cout << alfabeto << endl; // El último procedimiento, es tan simple como aplicar // el segundo de nuevo for(i = 0; alfabeto[i]; i++) if(isupper(alfabeto[i])) alfabeto[i] = tolower(alfabeto[i]); alfabeto[i] = toupper(alfabeto[i]); // Mostrar resultado: cout << alfabeto << endl; return 0; }</pre>
<p>Opción 2:</p> <pre>#include <iostream> using namespace std; // Código que genera el alfabeto alternando // Entre mayúsculas y minúsculas int main() { char letra = 'a'; bool sumar = false; while (letra <= 'z') { if (!sumar) { cout << char(letra-32); sumar = true; } else { cout << letra; sumar = false; } letra++; } cin.get(); }</pre>	

<p><i>Ejercicio:</i> Leer caracteres desde el teclado y contar cuántos hay de cada tipo.</p> <pre>// Cuenta letras #include <iostream> #include <cstdio> #include <cctype> using namespace std; int main() { int consonantes = 0; int vocales = 0; int digitos = 0; int mayusculas = 0; int minusculas = 0; int espacios = 0; int puntuacion = 0; char c; // caracteres leídos desde el teclado cout << "Contaremos caracteres hasta que se pulse '&'" << endl; while((c = getchar()) != '&') { if(isdigit(c)) digitos++; else if(isspace(c)) espacios++; else if(ispunct(c)) puntuacion++; else if(isalpha(c)) { if(isupper(c)) mayusculas++; else minusculas++; switch(tolower(c)) { case 'a': case 'e': case 'i': case 'o': case 'u': vocales++; break; default: consonantes++; } } } cout << "Resultados:" << endl; cout << "Dígitos: " << digitos << endl; cout << "Espacios: " << espacios << endl; cout << "Puntuación: " << puntuacion << endl; cout << "Alfabéticos: " << mayusculas+minusculas << endl; cout << "Mayúsculas: " << mayusculas << endl; cout << "Minúsculas: " << minusculas << endl; cout << "Vocales: " << vocales << endl; cout << "Consonantes: " << consonantes << endl; cout << "Total: " << digitos + espacios + vocales + consonantes + puntuacion << endl; return 0; }</pre>	<p><i>Ejercicio:</i> Pide la clave un máximo de tres veces y si es correcta, calcula la media de tres notas.</p> <pre>#include <stdio.h> #include <string.h> int main() { int i; float nota1, nota2, nota3, media; char nombre[20],clave[20]; nota1=0; nota2=0; nota3=0; i=0; printf("cual es tu nombre?? "); scanf("%s", &nombre); do { i=i+1; printf("Sr. %s introduzca su clave ", nombre); scanf("%s", &clave); } if(i>=3) { while (i=3) printf("Sr. %s su cuenta ha sido bloqueada", nombre); goto fin; } while (strcmp(clave, "ofimega") != 0); //comparo la clave con ofimega { printf("\n%s, cual es la nota de la primera evaluacion? ", nombre); scanf("%f", &nota1); printf("cual es la nota de la segunda evaluacion? ", nombre); scanf("%f", &nota2); printf("cual es la nota de la tercera evaluacion? ", nombre); scanf("%f", &nota3); media=(nota1+nota2+nota3)/3; //calculo la media if(media>=5) { printf("señor %s, has sacado un %2.1f y por lo tanto has aprobado", nombre, media); } else { printf("señor %s has suspendido el examen con un %2.1f", nombre, media); } } fin: scanf("%d"); //espera }</pre>
---	---

Matrices o arrays bidimensionales

Una matriz es una estructura de datos que permite almacenar un CONJUNTO de datos del MISMO tipo alineados.

Se nombra como ejemplo: `int mat [3][5]` para una matriz de 3 filas y 5 columnas.

Para indicar un elemento de la misma, se nombra la matriz seguida de DOS subíndices:

Por ejemplo: `mat [1][4]` primero se indica el nº de fila y luego el nº de columna (empezando a contar en 0).

En este ejemplo gráfico se ha almacenado el valor entero (int): 97

Todos los elementos de la matriz deben ser del mismo tipo (int, float, string).

Sus filas y cols comienzan a numerarse a partir de cero, similar a los vectores.

- › Sintaxis: `tipo nombre_array [nº_filas] [nº_cols];`
- › Ejemplo : `int matriz [2][4];`

		<i>mat</i>			
		Columnas			
		0	1	2	3
Filas	0	50	5	27	400
	1	0	67	90	6
	2	30	14	23	251

Arrays de cadenas

Ejemplo : `char lineas[30][80];`

Define un array de 30 cadenas cada una con un máximo de 80 caracteres.

Para acceder a una cadena individualmente, se especifica el índice de la fila.

Ejemplo : `gets(lineas[0]);`

Arrays de multiples dimensiones

Formato : `tipo nombre_array [<expN1>][<expN2>][<expN3>]...[<expNn>];`

Ejemplo : `int a[2][3][3];`

Inicialización

Una matriz o arreglo bidimensional se puede inicializar de este modo: `int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};`

En pseudocódigo:

Se suele utilizar el tipo `tMatriz` para tablas de varias dimensiones y `tTabla` para tablas de una dimensión.

Ejemplo:

```
tipo
    tMatriz=tabla[RMIN1...RMAX1, RMIN2...RMAX2] de tipo
ftipo
var
    m: tMatriz
fvar
```

Ordenar Tablas

Para ordenar datos en un array se suelen utilizar varios métodos:

Método de burbuja: Intercambiar cada pareja consecutiva que no esté ordenada:

```
Para i=1 hasta n-1
    Para j=i+1 hasta n
        Si A[i] > A[j]
            Intercambiar ( A[i], A[j])
```

Selección directa: En cada pasada busca el menor, y lo intercambia al final de la pasada.

```
Para i=1 hasta n-1
    menor = i
    Para j=i+1 hasta n
        Si A[j] < A[menor]
            menor = j
    Si menor <> i
        Intercambiar ( A[i], A[menor])
```

Inserción directa: Comparar cada elemento con los anteriores -que ya están ordenados- y desplazarlo hasta su posición correcta

```
Para i=2 hasta n
    j=i-1
    mientras (j>=1) y (A[j] > A[j+1])
        Intercambiar ( A[j], A[j+1])
        j = j - 1
```

Problema 17: Leer desde teclado un número y mostrar la tabla de multiplicar almacenada en una matriz de 3x10. →

```
//Tabla de multiplicar
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int tabla[3][10];
    int fil,col,num;
    printf("Numero de la tabla de multiplicar: ");
    scanf ("%d",&num);
    //--> pedir y guardar
    for (fil=0;fil<10;fil++)
    {
        tabla[0][fil]=num;
        tabla[1][fil]=fil+1;
        tabla[2][fil]=num*(fil+1);
    }
    //--> mostrar
    for (fil=0;fil<10;fil++)
    {
        printf("\n\t %d x %d = %d",
            tabla[0][fil],tabla[1][fil],tabla[2][fil]);
    }
    printf("\n\n");
    system("pause");
}
```

<p>Problema 19.- Escribir un programa que haga el producto de dos matrices 3x3. El programa debe incluir un procedimiento que lea las matrices, una función que haga el producto y otro procedimiento que escriba el resultado:</p> <pre> #include <cstdlib> #include <iostream> using namespace std; void leermatriz (float m[3][3]) { int i,j; for (i=1;i<=3;i++) { for (j=1;j<=3;j++) { cout<<"introducir el elemento "<<i<<","<<j<<endl; cin>>m[i][j]; } } } void producto_matrices (float a [3][3],float b[3][3],float p[3][3]) { int i,j,k; for (i=1;i<=3;i++) { p[i][j]=0; for (j=1;j<=3;j++) { p[i][j]=p[i][j]+a[i][k]*b[k][j]; } } } void escribir_matriz (float m[3][3]) { int i,j; for (i=1;i<=3;i++) { for (j=1;j<=3;j++) { cout<<m[i][j]<<" "; } cout<<endl; } } int main() { float a[3][3]; leermatriz (a); leermatriz (b); producto_matrices (a,b,p); escribir_matriz (p); system("PAUSE"); return EXIT_SUCCESS; } </pre>	<p>Búsqueda de un valor dentro de una matriz.</p> <pre> // Búsqueda binaria en un array #include <iostream> //o #include <stdio.h> en C using namespace std; //o #include <stdlib.h> en C int Busca(int*, int, int, int); // Declara función Busca //tabla, valor a buscar, inicio, tamaño o fin de la tabla int tabla[] = { 1, 3, 12, 33, 42, 43, 44, 45, 54, 55, 61, 63, 72, 73, 82, 83, 84, 85, 94, 95, 101, 103, 112, 133, 142, 143, 144, 145, 154, 155, 161, 163, 172, 173, 182, 183, 184, 185, 194, 195 }; int main() { int pos; int valor=141; pos = Busca(tabla, valor, 0, sizeof(tabla)/sizeof(tabla[0])- 1)+1; if(pos > 0) cout << "Valor " << valor << " encontrado en posicion: " << pos << endl; else cout << "Valor " << valor << " no encontrado" << endl; return 0; } int Busca(int* vec, int valor, int i, int s) /* Función de búsqueda binaria: Busca el "valor" dentro del vector "vec" entre los márgenes inferior "i" y superior "s" */ { int inferior = i; int superior = s; int central; do { central = inferior+(superior-inferior)/2; if(vec[central] == valor) return central; else if(vec[central] < valor) inferior = central+1; else superior = central-1; } while (superior >= inferior); return -1; } </pre>
---	---

Arrays Bidimensionales (Matrices)

Se necesita de un sistema que utiliza una **matriz** de 5 filas y cuatro columnas, para almacenar los 3 parciales y su promedio de 5 alumnos.

Pseudocódigo: alumnos

Arreglos:

calificaciones : real de [5] renglones [4] columnas

Variables:

num_alum, parcial : entero = 0

acum_cal : real = 0

inicio

para num_alum = 1 hasta num_alum > 5

 acum_cal = 0

 para parcial = 1 hasta parcial > 3

 Escribir "Calificación del alumno ", num_alum, "en parcial:", parcial

 Leer calificaciones[num_alum][parcial]

 acum_cal = acum_cal +

 calificaciones[num_alum][parcial]

 parcial = parcial + 1

 fin para

 calificaciones[num_alum][parcial] = acum_cal / 3

 num_alum = num_alum + 1

Fin para

Fin

// diagrama N-S : Alumnos

Inicio

Arreglos:

calificaciones : real de [5] renglones [4] columnas

Variables:

num_alum, parcial : entero = 0

acum_cal : real = 0

num_alum > 5

 acum_cal = 0

 parcial > 3

 Escribir "Calificación del alumno ", num_alum, "en parcial:", parcial

 Leer calificaciones[num_alum][parcial]

 acum_cal = acum_cal + calificaciones[num_alum][parcial]

 parcial = parcial + 1

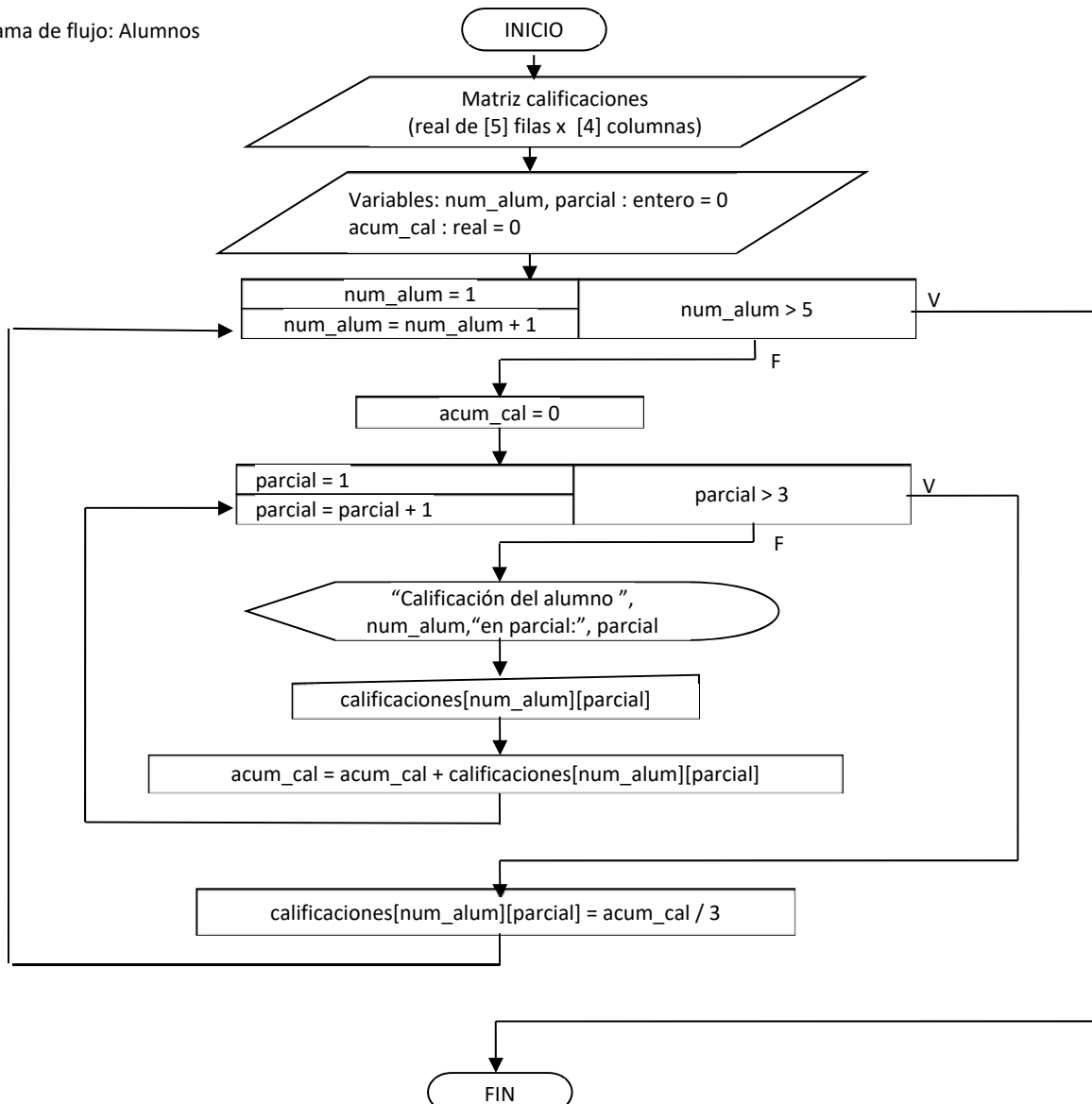
 calificaciones[num_alum][parcial] = acum_cal / 3

num_alum = 1

num_alum = num_alum + 1

Fin

// Diagrama de flujo: Alumnos



Ejercicio pedir barcos en el juego de hundir la flota

Tabla de multiplicar

```

//***** poner barcos ***** //
#include <stdio.h>
#include <string.h>

void mostrar_encabezado(); //acción dibuja
encabezado tablero
char barcos_juga[10][10]; // matriz de barcos del
jugador
char barcos_compu[10][10]; // matriz de barcos de
la computadora

int main ()
{
int col,fil,i;
char terminado;
do
{
mostrar_encabezado(); //lamada a la acción
printf("\n numero de columna:");
scanf("%d", &col);
printf("\n Numero de fila:");
scanf("%d", &fil);
col--; //bajamos la numeración para la matriz
fil--;
if (barcos_juga[col][fil]=='X')
{
printf("\n ya esta
ocupado\n");//comprobamos si ya hay uno...
system("pause");
}
else barcos_juga[col][fil]='X';

mostrar_encabezado();

for (fil = 0; fil < 10; fil++)
{
printf ("\n %d", fil+1);
for (col = 0; col < 10; col++)
printf (" %c ", barcos_juga[col][fil]);
}
printf("\n \n Has terminado(S/N)?: ");
scanf("%s", &terminado);
}
while (terminado!='S' && terminado!='s');
}

void mostrar_encabezado() //acción
{
system("cls"); //limpia
printf("
\n");
printf(" A B C D E F G H I J \n");
printf("
\n");
}

```

```

#include <stdio.h>
int main()
{
int tabla[3][10];
int fil,col,num;
printf("Numero de la tabla de multiplicar: ");
scanf ("%d",&num);
//pedir
for (fil=0;fil<10;fil++)
{
tabla[0][fil]=num;
tabla[1][fil]=fil+1;
tabla[2][fil]=num*(fil+1);
}
//mostrar
for (fil=0;fil<10;fil++)
{
printf("\n");
for(col=0;col<3;col++)
printf("%d",tabla[col][fil]);
}
}

```

Funciones y procedimientos

Acción: Un método, subprograma o subrutina a la que se llama sin parámetros y no retorna ningún valor, se denomina **acción**.

Procedimiento: Un método, subprograma o subrutina a la que se llama con parámetros se denomina **procedimiento**.

Función: Un método, subprograma o subrutina que además retorna un dato o valor se denomina **función**

Sintaxis de un procedimiento

```
void [nombre del método](parámetros)
{
    [algoritmo]
}
```

Sintaxis de una función

```
tipo de dato [nombre del método](parámetros)
{
    [algoritmo]
    return [tipo de dato]
}
```

Los parámetros pueden pasarse por valor o por referencia: Por defecto se pasan por valor y si lo queremos por referencia tener un puntero. Si ee parámetro es un array entonces se suele pasar por defecto por referencia.

El orden importa: En general, una función debe estar declarada antes de usarse o ponerse antes que la función main

Diseño descendente: Si se escriben los procedimientos antes que el programa principal-

Ejemplo 1. Acción.

Acción: Función sin argumentos y que no devuelve nada:

void: Si no va a devolver ningún valor, mejor se puede especificar incluyendo void (nulo o vacío):

```
void saludar()
{
    system("cls")
    printf("Bienvenido al programa\n");
    printf("de ejemplo\n");
}
```

```
#include <stdio.h>
#include <conio.h>
void prepara_pantalla() // -> Se escribe antes porque no se declara
{
    clrscr(); //o system("cls")
    printf( "La pantalla está limpia\n" );
    return; // No devuelve ningún valor, ni siquiera es necesario este return
}
int main()
{
    prepara_pantalla(); // Llamamos a la función escrita arriba
}
```

Nota: A partir del estándar C99, es obligatorio indicar el tipo devuelto, así que deberemos usar explícitamente "void" o int en la función main.

Ejemplo 2. Procedimiento.

Función con argumentos, no devuelve ningún valor:

En este ejemplo la función compara dos números y nos dice cual es mayor.

En este caso hemos declarado antes la función para poder escribirla al final

```
#include <stdio.h>
#include <conio.h>
void compara(int a, int b); // -> aquí declaramos la función (termina en ; )
int main()
{
    int num1, num2;

    printf( "Introduzca dos números: " );
    scanf( "%i %i", &num1, &num2 );
    compara(num1, num2); // -> Llamamos a la función con sus dos argumentos
    return 0;
}
void compara(int a, int b) // -> aquí escribimos la función (no poner ; )
{
    if ( a>b ) printf( "%i es mayor que %i\n" , a, b );
    else printf( "%i es mayor que %i\n", b, a );
}
```

Ejemplo 3. Función.

Función con argumentos que devuelve un valor.

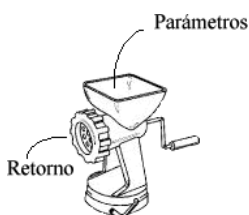
Este ejemplo es como el anterior pero devuelve como resultado el mayor de los dos números.

Return finaliza la función y devuelve el dato.

En este ejemplo podíamos haber hecho también:

```
printf( "El mayor de los dos es %i\n",
    compara( num1, num2 ) );
```

De esta forma nos ahorramos tener que definir la variable 'resultado'.



```
#include <stdio.h>
#include <conio.h>
int compara( int a, int b ) /* parámetros a y b*/
{
    int mayor
    if ( a>b )
        mayor = a;
    else mayor = b;
    return mayor;
}
int main()
{
    int num1, num2;
    int resultado;
    printf( "Introduzca dos números: " );
    scanf( "%i %i", num1, num2 );
    resultado = compara( num1, num2 ); /* Recogemos el valor que
    devuelve la función en resultado */
    printf( "El mayor de los dos es %i\n", resultado );
    return 0;
}
```

Ejercicio de funciones: Crear un programa que muestre un menú para sumar, restar o salir mediante los tres métodos: acciones, procedimientos y funciones.

- 1) Sumar
- 2) Restar
- 0) Salir

Tras mostrar el menú, el programa evalúa la opción para realizar la suma o la resta y volverá a mostrar el menú.

Variante con acción o subrutina

```
#include <stdio.h>
#include <stdlib.h>
//declaramos antes las acciones
void sumar();
void restar();
int main()
{
    int op=1;
    do
    {
        system("cls"); //limpia
        printf("tria una
        opcio:\n");
        printf("1-sumar\n");
        printf("2-restar\n");
        printf("0-salir\n");
        scanf("%d", &op);
        if (op==1)
        {
            sumar();
        }
        else if (op==2)
        {
            restar();
        }
    }
    while(op>0);
return 0;
}

void sumar()
{
    int n1, n2,resultat;
    printf("\n n1: ");
    scanf("%d",&n1);
    printf("\n n2: ");
    scanf("%d",&n2);
    resultat=n1+n2;
    printf("el resultat es %d
    \n",resultat);
    printf("Pulsa Intro para
    continuar...");
    scanf("%d",&resultat);
}

void restar()
{
    int n1, n2,resultat;
    printf("\n n1: ");
    scanf("%d",&n1);
    printf("\n n2: ");
    scanf("%d",&n2);
    resultat=n1-n2;
    printf("el resultat es
    %d\n",resultat);
    printf("Pulsa Intro para
    continuar...");
    scanf("%d",&resultat);
}
```

Variante con procedimientos

```
#include <stdio.h>
#include <stdlib.h> // system("cls");
#include <conio.h> // getch y clrscr

void pedirnumeros();
void sumar(int n1,int n2);
void restar(int n1,int n2);
int n1, n2,result; //variables
globales

int main()
{
    int op=1;
    do
    {
        system("cls"); //o clrscr();
        printf("Escoge opcion:\n");
        printf("1-sumar\n");
        printf("2-restar\n");
        printf("0-salir\n");
        scanf("%d", &op);

        if (op==1)
        {
            pedirnumeros();
            sumar(n1,n2);
        }
        else if (op==2)
        {
            pedirnumeros();
            restar(n1,n2);
        }
    }
    while(op>0);

    return 0;
}

void pedirnumeros()
{
    printf("\n Valor de num 1: ");
    scanf("%d",&n1);
    printf("\n Valor de num 2: ");
    scanf("%d",&n2);
}

void sumar(int n1,int n2)
{
    result=n1+n2;
    printf("Resultat de la suma es %d
    \n",result);
    printf("Pulse una tecla para
    continuar...");
    getch();
}

void restar(int n1,int n2)
{
    result=n1-n2;
    printf("Resultat de la resta es %d
    \n",result);
    printf("Pulse una tecla para
    continuar...");
    getch();
}}
```

Variante con funciones

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

void mostrarmenu();
void pedirnumeros();
void mostrarresult(int r);
int sumar(int n1,int n2);
int restar(int n1,int n2);

int n1, n2; //var globales

int main()
{
    int op=1,result;//var locales
    do
    {
        mostrarmenu();
        scanf("%d", &op);
        switch (op)
        {
            case 1:
                pedirnumeros();
                result=sumar(n1,n2);
                mostrarresult(result);
                break;
            case 2:
                pedirnumeros();
                restar(n1,n2);
                mostrarresult(result);
                break;
        }
    }
    while(op>0);

    return 0;
}

void mostrarmenu()
{
    system("cls"); //o clrscr();
    printf("Escoge opcion:\n");
    printf("1-sumar\n");
    printf("2-restar\n");
    printf("0-salir\n");
}

void pedirnumeros()
{
    printf("\n Valor de num 1: ");
    scanf("%d",&n1);
    printf("\n Valor de num 2: ");
    scanf("%d",&n2);
}

sumar(int n1,int n2)
{
    int r;
    r=n1+n2;
    return r;
}

restar(int n1,int n2)
{
    int r;
    r=n1-n2;
    return r;
}

void mostrarresult(int n)
{
    printf("El resultado es %d \n",n);
    printf("Pulse una tecla para
    continuar...");
    getch();
}
```

Añadir al ejercicio anterior funciones para *sumatorio*, *factorial*, *combinaciones* y *variaciones*

Ejercicio sumatorio diseño descendente

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

int calcular(int a) //procedimiento para sumatorio
{
    int n,aux;
    double res; //resultado
    printf("Has escogido la opcion %d\n",a);
    if (a==2 ) {
        printf("Escriba el numero sobre el que quiere el
sumatorio");
        scanf("%d",&n);
        res=0;
        for(aux=n;aux>=1;aux--) res+=aux;
    }
    else if (a==3 )
    {
        printf("Escriba el numero factorial");
        scanf("%d",&n);
        res=1;
        for(aux=n;aux>=1;aux--) res=res * aux; // o: res*=aux
    }
    return res;
}
//-----
int main(){
    int seleccion; //var opción escogida
    double res; //var resultado
    do{
        /* Mostrar el menú*/
        printf("1 Salir\n2 Sumatorio\n");
        printf("3 Factorial\n\nEscriba su opcion");
        scanf("%d",&seleccion);
        switch(seleccion)
        {
            case 1:
                printf("Pulse una tecla para continuar\n\n",res);
                getch();
                break;
            default: // para el caso2 y caso 3
                res=calcular(seleccion); //-> llamada a función
                printf("El resultado es:%.01f\n",res);
                printf("Pulse una tecla para continuar\n\n",res);
                getch();
        } //-> fin del case
    } // fin del do
    while(seleccion!=1);
} // fin del main
```

Ejercicio factorial y combinaciones

```
#include <stdio.h>
int combinacions(int n, int m);

int main ()
{
    int m;
    int n;
    int result;
    result=0;
    printf("Valor de m:");
    scanf("%d",&m);
    printf("Valor de n:");
    scanf("%d",&n);
    result=combinacions(n,m);
    printf("La combinacion de %d elementos
tomados de %d en %d es %d",m,n,n,result);
    scanf("%d"); //o getch() o return 0
}

int combinacions(int n, int m)
{
    int var,per,comb,i;
    per=n;
    var=m;
    if (n<=m)
    {
        for(i=n-1;i>=1;i--) per=per*i;
        for(i=m-1;i>=n;i--) var=var*i;
        comb=var/per;
    }
    else
        comb=0;
    return comb;
}
```

Parámetros de paso:

Pasar parámetros por referencia: Sintaxis: tipo Nombre (tipo E1, tipo E2, tipo *ES1, tipo *Es2)

Se pasan punteros por referencia para usarlos como parámetros de salida.

Pasar arrays a una función:

Por defecto, los arrays, arreglos o vectores que se pasan a una función son como referencia y no como valor. Esto significa que todas las modificaciones que hagamos dentro de la función, realmente se realizan en el array original.

La función suele también recibir un segundo parámetro que indica el tamaño del arreglo. Ejemplo:

▸ `int MiFuncion(int mi_arreglo[], int num_eLemetos);`

Para llamar a la función anterior, se pone como primer argumento el nombre del arreglo (sin corchetes) y como segundo argumento el número de elementos del arreglo. Ejemplo:

▸ `x = MiFuncion(numeros, 10);`

Para pasar a una función un arreglo de dos dimensiones, si debemos indicar el tamaño de la segunda dimensión del arreglo.

Ejemplo:

`int MiFuncion(int mi_arreglo[][5], int num_eLemetos);`

```
#include <stdio.h>
void ImprimeMatriz(int ma[][3], int filas)
{
    int i=0,j=0;
    for (i=0; i<filas; ++i)
        { for (j=0; j<3; ++j)
            { printf("%d ",ma[i][j]);
            }
        printf("\n");}
}

int main()
{
    int x=0,y=0;
    int matriz[3][3] = {{0},{0},{0}};
    printf("Introduzca valores para la matriz\n");
    for (x=0; x<3; ++x)
        { for (y=0; y<3; ++y)
            {printf("Valor elemento [%d][%d]: ", x, y);
            scanf("%d",&matriz[x][y]);
            }
        printf("\n");
    }
    printf("Matriz\n");
    ImprimeMatriz(matriz, 3);
    return 0;
}
```

Parámetros de paso a funciones:

- › **Por valor:** envía una copia de los parámetros a la función por lo tanto los cambios que se hagan en ella no son tomados en cuenta dentro de la función main().
- › **Por referencia:** se hace utilizando apuntadores. Se envía la dirección de memoria de la variable, por lo tanto los cambios que haga la función si afectan el valor de la variable.

Ejemplo de paso por valor:

```
#include <stdio.h>
void sumar_valor(int numero);

int main(void)
{
    int numero = 57;
    sumar_valor(numero);
    printf("Valor de numero dentro de main() es:
%d\n", numero);
    return 0;
}

void sumar_valor(int numero) //-> paso por valor
{
    numero++;          //-> aumentamos 1
    printf("Valor de numero dentro sumar_valor() es:
%d\n", numero);
    return;
}
```

Ejemplo de paso por referencia:

```
#include <stdio.h>

void sumar_referencia(int *numero);
int main(void)
{
    int numero = 57;
    sumar_referencia(&numero); printf("\nValor de
numero dentro de main() es: %d ", numero);
    return 0;
}

void sumar_referencia(int *numero) //por referencia
{
    *numero += 1;          //-> aumentamos 1
    printf("\nValor de numero dentro de
sumar_referencia() es: %d", *numero);
    return;
}
```

Punteros como parámetros de funciones

```
#include <iostream>
using namespace std;

void funcion(int *q); //<- parámetro puntero

int main() {
    int a;
    int *p;

    a = 100;
    p = &a;
    // Llamamos a funcion con un puntero
    funcion(p); // (1)
    cout << "Variable a: " << a << endl;
    cout << "Variable *p: " << *p << endl;
    // Llamada a funcion con la dirección de "a"
    (constante)
    funcion(&a); // (2)
    cout << "Variable a: " << a << endl;
    cout << "Variable *p: " << *p << endl;

    return 0;
}

void funcion(int *q) {
    // Cambiamos el valor de la variable apuntada por el
    puntero
    *q += 50;
    q++;
}
```

Arrays como parámetros de funciones

```
#include <stdio.h>

#define F 10 //-> constantes del preprocesador
#define C 20

void funcion(int tab[][C]); //<- parámetro array

int main() {
    int tabla[F][C];
    funcion(tabla); //<- llamada a función sin corchetes
    printf("%d", tabla[2][4]);
}

void funcion(int tab[][C]) //<- no se pone primer corchete
{
    tab[2][4] = 6;
}
```

Recursividad de funciones: Es la llamada repetitiva a sí misma. La recursividad es más lenta que la interacción

Ejemplo: Factorial mediante llamada recursiva a la función factorial.

```
#include <stdio.h>

long factorial(int n)
{
    if (n > 1) // caso base
        return n * factorial (n - 1); //-> llamada a sí misma
}

int main(void)
{
    printf("%ld ", factorial(5)); //-> primera llamada
    return 0;
}
```

TRES EN RAYA

```
#include <stdio.h> //-> printf ), scanf()
#include <stdlib.h> //-> exit()
#include <conio.h> //-> getch()
#define ESPACIO ' '
char matriz[3][3] =
{
    ESPACIO, ESPACIO, ESPACIO,
    ESPACIO, ESPACIO, ESPACIO,
    ESPACIO, ESPACIO, ESPACIO
};
//-> declaramos las acciones
void obtener_movimiento_de_jugador (void);
void obtener_movimiento_de_computadora (void);
void mostrar_matriz (void);
char comprobar (void); //-> función retorna char
```

```
int main (void)
```

```
{
    char hecho = ESPACIO;

    printf ("JUEGO DE LAS TRES EN RAYA.\n\n");
    do
    {
        mostrar_matriz ();
        obtener_movimiento_de_jugador ();
        hecho = comprobar (); //ver si gana el jugador
        if (hecho != ESPACIO)
            break;
        obtener_movimiento_de_computadora ();
        hecho = comprobar (); //ver si gana
    }
    while (hecho == ESPACIO);

    if (hecho == 'X')
        printf ("\n *** ¡HAS GANADO! ***\n");
    else
        printf ("\n *** ¡YO GANO! ***\n");

    mostrar_matriz (); //mostrar posiciones finales
    printf ("\nPulsa una tecla para finalizar");
    getch ();
}
```

```
void obtener_movimiento_de_jugador (void)
```

```
{
    int x, y;

    printf ("\nIntroduzca num de fila y columna");
    scanf ("%d %d", &x, &y);
    x--; //fila
    y--; //columna
    if (matriz[x][y] != ESPACIO)
    {
        printf ("Inválido, prueba de nuevo.\n");
        obtener_movimiento_de_jugador ();
    }
    else
        matriz[x][y] = 'X';
}
```

```
void obtener_movimiento_de_computadora (void)
```

```
{
    int encontrado = 0;
    int i, j;
    for (i = 0; ! encontrado && i < 3; i++)
        for (j = 0; ! encontrado && j < 3; j++)
            if (matriz[i][j] == ESPACIO)
                encontrado = 1;
    if (encontrado)
        matriz[i-1][j-1] = 'O';
    else
    {
        printf ("Tablero completo.\n");
        exit (0);
    }
}
```

```
void mostrar_matriz (void)
```

```
{
    int i;
    printf ("\n    1  2  3");
    printf ("\n +-----+");
    for (i = 0; i < 3; i++)
    {
        printf ("\nd | %c | %c | %c |", i+1,
            matriz[i][0], matriz[i][1], matriz[i][2]);

        if (i != 2)
            printf ("\n +-----+!");
    }
    printf ("\n +-----+");
}
```

```
char comprobar (void)
```

```
{
    int t;
    /* comprobar filas */
    for (t = 0; t < 3; t++)
        if (matriz[t][0] == matriz[t][1] &&
            matriz[t][1] == matriz[t][2])
            return matriz[t][0];

    /* comprobar columnas */
    for (t = 0; t < 3; t++)
        if (matriz[0][t] == matriz[1][t] &&
            matriz[1][t] == matriz[2][t])
            return matriz[0][t];

    /* comprobar diagonal principal */
    if (matriz[0][0] == matriz[1][1] &&
        matriz[1][1] == matriz[2][2])
        return matriz[0][0];

    /* comprobar diagonal inversa */
    if (matriz[0][2] == matriz[1][1] &&
        matriz[1][1] == matriz[2][0])
        return matriz[0][2];

    return ESPACIO;
}
```

Ficheros: Lectura y escritura.

FILE *: Puntero que indica al nombre del archivo. Todas las funciones de entrada/salida estándar usan este puntero para conseguir información sobre el fichero abierto. Este puntero no apunta al archivo sino a una estructura que contiene información sobre él: nombre del archivo, la dirección de memoria donde se almacena, tamaño del buffer...

Funciones para abrir y cerrar el fichero:

- **fopen**: Abre el fichero – Sintaxis: **FILE * = fopen**(nombre_fichero, modo);
- **fclose**: Cierra el fichero – Sintaxis: **fclose**(fichero)

Funciones para leer/guardar en ficheros: Dependerán del dato que queramos escribir y/o leer y del tipo de fichero. Pero casi siempre empiezan por f: Ejemplos: fscanf/ fprintf, fgets/fputs, fread/fwrite (luego se ven).

Tipos de ficheros/archivos:

- Ficheros de **texto**: Almacenan únicamente caracteres de texto. Se suele utilizar las funciones *Fscanf*, *fprintf* o *getc*, *putc*
- Ficheros **binarios**: Permiten almacenar varios tipos de datos como structs o imágenes. Se tratan con *fread* y *fwrite*

Variantes de modo:

- r read**: Abre un fichero existente para lectura.
- w write**: Crea un fichero nuevo (o borra su contenido si existe) y lo abre para escritura.
- a append**: Abre un fichero para escritura en modo añadir datos, si no existe lo crea.

Modificadores siguiendo a los modos anteriores:

- b** Abre el fichero en modo binario.
- t** Abre el fichero en modo texto.
- +** Abre el fichero para lectura y escritura.

Ejemplos de combinaciones:

- rb+** Abre el fichero en modo binario para lectura y escritura.
- w+** Crea (o lo borra si existe) un fichero para lectura y escritura.
- Rt** Abre un archivo existente en modo texto para lectura.

Ejemplos:

Ejemplo 1. Grabación de un fichero de texto

En este ejemplo, abrimos el fichero "prueba.txt" en modo grabación de texto: **fopen**("prueba.txt", "wt")

Grabamos 2 líneas enteras de texto utilizando la función *fputs*. La segunda línea se guarda en dos partes.

Por último, cerramos el fichero (*Fclose*)

```
#include <stdio.h>
int main()
{
    FILE* fichero; //variable del tipo fichero
    fichero = fopen("prueba.txt", "wt");
    fputs("Esto es una línea a guardar\n", fichero);
    fputs("Esto es otra línea a guardar", fichero);
    fputs(" y esto es continuación de la anterior\n", fichero);
    fclose(fichero);
    return 0;
}
```

Ejemplo 2. Lectura de un fichero de texto.

Explicación

- **Abrir el fichero**: fichero = **fopen**(nombre_fichero, modo);
- **Comprobar** si se ha abierto:
 - If (fichero==NULL) → { printf("No se puede leer\n"); exit(1); }
- **Leer** el fichero – **getc** → Con *getc*, *fgetc* o *fscanf*: lee los caracteres uno a uno. También existen *fgets* o *fread* que leen más de un carácter.
 - El formato de la función *getc* (y de *fgetc*) es:


```
int getc(FILE *fichero);
```

 En el ejercicio lo usaremos como: `letra = getc(fichero);`
- **Comprobar fin de fichero** – **feof** → Cuando entramos en el bucle *while*, la lectura se realiza hasta que se encuentre el final del fichero. Para detectar el final del fichero se pueden usar dos formas:
 - con la función *feof*()
 - comprobando si el valor de la letra es EOF (la última del fichero)

En el ejemplo hemos usado la función *feof*: `int feof(FILE *fichero);`

si se ha llegado al final de fichero devuelve un valor distinto de 0. Si no se ha llegado, devuelve un cero:

`while (feof(fichero)==0)` otro ejemplo: `while (letra!=EOF)` `while (!feof(fichero))`

- **Cerrar el fichero** con *fclose*(fichero); **fclose**: → Importante no olvidar cerrar el fichero pues podría corromperse. Al cerrarlo se vacían los buffers y se guarda el fichero en disco. Al cerrarse con la función *fclose*(fichero), si todo va bien *fclose* devuelve un cero, si hay problemas devuelve otro valor. Por ejemplo, si el disco está lleno:

`if (fclose(fichero)!=0) printf("Problemas al cerrar el fichero\n");`

```
#include <stdio.h>

int main()
{
    FILE *fichero; //variable del tipo fichero
    char letra; //variable del tipo caracter

    fichero = fopen("prueba.txt", "r");
    if (fichero==NULL)
    {
        printf( "No se puede abrir el fichero.\n" );
        exit( 1 );
    }
    //muestra el fichero letra por letra...
    letra=getc(fichero);
    while (feof(fichero)==0)
    {
        printf( "%c",letra );
        letra=getc(fichero); //lee el caracter
    }
    if (fclose(fichero)!=0)
        printf( "Problemas al cerrar el fichero\n" );
}
```

Ficheros de texto:

las funciones utilizaremos para tratar ficheros dependerán del dato que queramos escribir y/o leer en el fichero.

Lectura de ficheros de texto

- **fscanf**— Lee de un fichero datos con formato. Ejemplo: **fscanf**(pFile, "%f", &f); -> Ver recuadro
- **getc**: Lee caracteres uno a uno (*obsoleto*). Ejemplo: **c = getc**(pFile) **c** es un carácter y pFile el fichero
- **fgets**: Lee cadenas de texto (string) de una longitud por líneas. Sintaxis: **fgets**(buffervariable, longitud_max, fichero);

Escritura de ficheros de texto

- **fprintf** — Escribe en un fichero datos con formato. Ejemplo: **fscanf**(pFile, "%f", &f)
- **putc**: Escribe un carácter en el fichero. Ejemplo: **putc**(c, pFile); **c** es un carácter y pFile el fichero
- **fputs** — Escribe líneas enteras: trabaja junto con la función fgets. Sintaxis: **fputs**(cadena, fichero);
Lee desde el fichero hasta que encuentra un carácter '\n' o hasta que lee longitud_max-1 caracteres y añade '\0' al final de la cadena. La cadena leída la almacena en buffer. Si se encuentra EOF antes o se produce un error devuelve NULL.

<pre>#include <stdio.h> int main() { FILE *fichero; char letra; fichero = fopen("origen.txt","r"); // r=lectura if (fichero==NULL) { printf("No se puede abrir el fichero.\n"); exit(1); } printf("Contenido del fichero:\n"); letra=getc(fichero); //coge un caracter while (feof(fichero)==0) { printf("%c",letra); letra=getc(fichero); } if (fclose(fichero)!=0) printf("Problemas al cerrar el fichero\n"); }</pre>	<pre>#include <stdio.h> int main() { FILE *fichero; char texto[100]; fichero=fopen("origen.txt","r"); if (fichero==NULL) { printf("No se puede abrir el fichero.\n"); exit(1); } printf("Contenido del fichero:\n"); fgets(texto,100,fichero); //coge 100 caracteres while (feof(fichero)==0) { printf("%s",texto); fgets(texto,100,fichero); } if (fclose(fichero)!=0) printf("Problemas al cerrar el fichero\n"); }</pre>
--	---

Ejercicio 1

Escribe un programa que lea un fichero, le suprima todas las vocales y lo guarde en otro.

Ejemplo fichero origen.txt:

```
El alegre campesino pasea por el
campo ajeno a que el toro se
acerca por detrás
```

El fichero destino.txt:

```
l lgr cmpsn ps pr l cmp
jn q l tr s crc pr dtrás
```

Ejercicio 2

La solución propuesta no elimina las vocales acentuadas, modifica el programa para conseguirlo.

Lo único que hay que hacer es añadir "áéíóúÁÉÍÓÚ" a la línea:

```
if (!strchr("AEIOUaeiouáéíóúÁÉÍÓÚ",letra)) putc( letra, destino );
```

<pre>#include <stdio.h> int main() { FILE *origen, *destino; char letra; origen=fopen("origen.txt","r"); destino=fopen("destino.txt","w"); if (origen==NULL destino==NULL) { printf("Error en fichero\n"); exit(1); } letra=getc(origen); while (feof(origen)==0) { putc(letra,destino); printf("%c",letra); letra=getc(origen); } if (fclose(origen)!=0) printf("Problemas al cerrar el fichero origen.txt\n"); if (fclose(destino)!=0) printf("Problemas al cerrar el fichero destino.txt\n"); }</pre>	<pre>Ejercicio 1 #include <stdio.h> int main() { FILE *origen, *destino; char letra; origen=fopen("origen.txt","r"); destino=fopen("destino.txt","w"); if (origen==NULL destino==NULL) { printf("Error en fichero\n"); exit(1); } letra=getc(origen); while (feof(origen)==0) { if (!strchr("AEIOUaeiou",letra)) putc(letra, destino); letra=getc(origen); } if (fclose(origen)!=0) printf("Problemas al cerrar el fichero origen.txt\n"); if (fclose(destino)!=0) printf("Problemas al cerrar el fichero destino.txt\n"); }</pre>
---	---

fprintf y fscanf

Si queremos leer un valor numérico de un fichero de texto podemos usar de la librería stdio:

sscanf — Lee un string de entrada de acuerdo con un formato

ejemplo: sscanf(texto,"%f",&valordecimal);

fscanf — Lee de un fichero datos con formato

ejemplo: fscanf(pFile,"%f",&f); -> Ver recuadro

fprintf — Escribe en un fichero datos con formato

Trabajan igual que sus equivalentes printf y scanf. La única diferencia es que podemos especificar el fichero sobre el que operar (si se desea puede ser la pantalla para fprintf o el teclado para fscanf).

Los formatos de estas dos funciones son:

```
int fprintf(var fichero, char formato, valor_a_grabar);
```

```
int fscanf(var fichero, char formato, valor_a_leer);
```

```
//-- guarda el numero y la letra pi -- //
#include <stdio.h>
int main ()
{
    char str [80];
    float f;
    FILE * pFile;
    pFile = fopen ("fichero.txt","w+");
    fprintf (pFile, "%f %s", 3.1416, "PI");
    rewind (pFile); // volver al principio
    fscanf (pFile, "%f", &f);
    fscanf (pFile, "%s", str);
    fclose (pFile);
    printf ("lectura: %f y %s \n",f,str);
    return 0;
}
```

Movimientos por el fichero: fseek, ftell, rewind

fseek

Salta a una cierta posición de un fichero. Desplaza la posición actual del cursor a otro punto. El desplazamiento puede ser positivo (avanzar), cero o negativo (retroceder). Fseek devuelve valor 0 si no ha habido problemas.

Sintaxis: `fseek(*pfichero, desplazamiento, modo); -> int`

Ejemplo: `fseek(fichero, -10, SEEK_CUR);`

- **Pfichero**: puntero de tipo FILE que apunta al fichero a usar.
- **Desplazamiento**: Posiciones (o bytes) que queremos desplazar el cursor.
- **Modo**: posición de origen. Hay tres modos: SEEK_SET (0, comienzo), SEEK_CUR (1, actual), SEEK_END (2, final)
 - SEEK_SET : El puntero se desplaza desde el principio del fichero.
 - SEEK_CUR : El puntero se desplaza desde la posición actual del fichero.
 - SEEK_END : El puntero se desplaza desde el final del fichero.

```
// -- lectura de una letra en una posición del archivo -- //
#include <stdio.h>
int main()
{
    FILE *fichero; //-> puntero que indica el archivo
    long posicion; //-> entero largo que indica el desplazamiento del cursor
    int resultado; //-> valor de retorno
    fichero = fopen( "origen.txt", "r" );
    printf( "¿Qué posición quieres leer? " ); fflush(stdout);
    scanf( "%D", &posicion );
    resultado = fseek(fichero, posicion, SEEK_SET);
    if (!resultado) // si no hay problemas resultado es 0
        printf( "En la posición %D está la letra %c.\n", posicion, getc(fichero) );
    else
        printf( "Problemas posicionando el cursor.\n" );
    fclose( fichero );
}
```

ftell

Devuelve la posición actual en un fichero (-1 en caso de error). Esta función es complementaria a fseek

Sintaxis: `ftell(FILE *pfichero); -> devuelve long`

Ejemplo de uso: `posActual = ftell(ficheroDatos);`

El valor que nos da ftell puede ser usado por fseek para volver a la posición actual.

rewind (puntero_fichero): Sitúa el puntero al principio del archivo.

remove(nombre_fichero): Borra el fichero.

Imprimir un fichero: Dirigir el texto a una impresora.

Un fichero lo mandamos a impresora si redirigimos la salida hacia el dispositivo llamado PRN por ejemplo impresora:

En Windows: `impresora = fopen("prn:", "wt");`

En Linux: `impresora = fopen("/dev/lp", "wt");`

```
#include <stdio.h>
int main()
{
    FILE* impresora;
    impresora = fopen("prn:", "wt");
    fputs("Esto va a la impresora\n", impresora);
    fclose(impresora);
    return 0;
}
```

Lectura y escritura de ficheros binarios.

Los ficheros tratados en modo texto son adecuados para trabajar con caracteres. Pero los ficheros tratados en modo binarios nos permiten guardar o leer datos de cualquier tipo, como fichero de imagen bmp o incluso con estructuras. (→ véase el tema structs) ya que, si se abre en modo texto, habría que escribir cada campo como una línea de texto con `fprintf` para que convirtiera los campos de tipo numéricos en tipo de texto.

fread y **fwrite** Son las funciones para leer y guardar en ficheros modo binario.

- **Fwrite:** Permite escribir en un fichero. *Sintaxis:* `fwrite (buffer, tamaño, numero_veces, *pfichero);`
- **Fread:** Para sacar información de un fichero. *Sintaxis:* `fread (buffer, tamaño, numero_veces, *pfichero);`
 - *buffer* : variable que contiene los datos que vamos a escribir en el fichero.
 - *tamaño* : el tamaño del tipo de dato a escribir. Puede ser un int, un float, una estructura, ... Para conocer su tamaño usamos el operador `sizeof`. Si en el fichero hay menos elementos, `fread` devolverá sólo los que tiene.
 - *numero* : el número de datos a escribir.
 - *pfichero* : El puntero al fichero sobre el que trabajamos.

Para guardar una estructura: `fwrite (*buffer, tamaño, nº de veces, puntero_fichero)`. Para calcular el tamaño en bytes de un dato o un tipo de dato se suele utilizar la función `sizeof (dato)` o `sizeof (tipo-de-dato)`;

Ejemplo agenda de teléfonos: Guardar y leer un struct en un fichero binario

Guardar en fichero:

Guarda el nombre, apellido y teléfono de una estructura en un archivo.

El bucle termina cuando el 'nombre' se deja en blanco.

Este programa guarda los datos personales mediante `fwrite` usando la estructura `registro`. Abrimos el fichero en modo 'a' (append, añadir), para que los datos que introducimos se añadan al final del fichero.

Una vez abierto entramos en un bucle `do-while` mediante el cual introducimos los datos. Los datos se van almacenando en la variable `registro` (que es una estructura). Una vez tenemos todos los datos de la persona los metemos en el fichero con `fwrite`:

`fwrite(registro, sizeof(registro), 1, fichero);`

- `registro` - es la variable (en este caso una estructura) que contiene la información a meter al fichero.
- `sizeof(registro)` - lo utilizamos para saber cuál es el número de bytes que vamos a guardar, el tamaño en bytes que ocupa la estructura.
- `1` - indica que sólo vamos a guardar un elemento. Cada vez que se recorre el bucle guardamos sólo un elemento.
- `fichero` - el puntero `FILE` al fichero donde vamos a escribir.

```
#include <stdio.h>
struct {
    char nombre[20];
    char apellido[20];
    char telefono[15];
} registro;

int main()
{
    FILE *fichero;
    fichero = fopen( "nombres.txt", "a" );
    do { //inicio bucle do-while
        printf( "Nombre: " );
        gets(registro.nombre);
        if (strcmp(registro.nombre,""))
        {
            printf( "Apellido: " );
            gets(registro.apellido);
            printf( "Teléfono: " );
            gets(registro.telefono);
            fwrite(registro, sizeof(registro), 1, fichero );
        }
    } while (strcmp(registro.nombre,"")!=0);
    fclose( fichero );
}
```

Leer en fichero:

Siguiendo con el ejemplo anterior ahora vamos a leer los datos que habíamos introducido en "nombres.txt".

Abrimos el fichero `nombres.txt` en modo lectura.

Con el bucle `while` nos aseguramos que recorremos el fichero hasta el final (y que no nos pasamos).

La función `fread` lee un registro (`numero=1`) del tamaño de la estructura `registro`. Si realmente ha conseguido leer un registro la función devolverá un `1`, en cuyo caso la condición del 'if' será verdadera y se imprimirá el registro en la pantalla. En caso de que no queden más registros en el fichero, `fread` devolverá `0` y no se mostrará nada en la pantalla.

```
#include <stdio.h>
struct {
    char nombre[20];
    char apellido[20];
    char telefono[15];
} registro;

int main()
{
    FILE *fichero;

    fichero = fopen( "nombres.txt", "r" );
    while (!feof(fichero)) {
        if (fread(registro, sizeof(registro), 1, fichero )) {
            printf( "Nombre: %s\n", registro.nombre );
            printf( "Apellido: %s\n", registro.apellido);
            printf( "Teléfono: %s\n", registro.telefono);
        }
    }
    fclose( fichero );
}
```

Ejercicios con ficheros de texto: Calculo del área de una esfera leyendo los valores de un fichero.

<p>Objetivo: Lee los radios en un fichero de texto llamado radi_esfera.txt, calcula sus áreas y las guarda en otro fichero llamado area_esfera.txt</p> <p>Planteamiento base: Abriremos 2 archivos con fopen: uno en modo Read y el otro en modo Write</p> <pre> /**cálculo de área de una esfera desde un archivo ** #include <stdio.h> #include <math.h> // para la constante M_PI int main() { float radi, area; // declara 2 variables floatantes FILE *fitxer_radi, *fitxer_area; fitxer_radi = fopen("radi_esfera.txt", "r"); fitxer_area = fopen("area_esfera.txt", "w"); if (fitxer_radi==NULL) { printf("No se puede abrir el archivo de origen\n"); exit(1); } while (feof(fitxer_radi)==0) { fscanf (fitxer_radi, "%f", &radi); area = 4*M_PI * radi*radi; printf ("area de %f es %f\n ", radi, area); fprintf (fitxer_area, "area de %f es %f\n ", radi, area); } fclose (fitxer_radi); fclose (fitxer_area); printf ("Grabado en el archivo: area_esfera.txt"); scanf("%d"); } </pre>	<p>Ejercicio CambiarClave Pide al usuario la clave actual (clave) y la compara con una clave guardada en un fichero (claveOld). Si la clave es correcta permite guardar en el fichero la clave nueva (claveNew)</p> <pre> /***/ Cambiar y guardar clave en un fichero **** #include <stdio.h> #include <string.h> main() { char clave[10],claveNew[10],claveOld[10]; FILE *archivo; archivo=fopen("clave.txt","r"); //abre clave.txt modo lectura if (archivo==NULL) { printf("No se puede abrir el archivo de datos\n"); } else { fscanf (archivo, "%s", &claveOld); //lee la clave printf("Ingrese la clave actual: "); scanf("%s",&clave); if (strcmp(clave, claveOld) != 0) //compara texto { printf("Clave anterior incorrecta"); } else { printf("Ingrese la nueva clave"); scanf("%s",&claveNew); fprintf (archivo, "%s ", claveNew); //guarda la clave printf ("Grabada la nueva clave %s",claveNew); fclose(archivo); scanf("%d"); } } } </pre>
--	---

Ejemplos con ficheros binarios y structs

<pre> FILE *f; int v[6], elem_escritos, num; f = fopen ("datos.cht", "wb"); /* Para escribir los 3 últimos elementos de v (el 2, el 3 y el 4) */ elem_escritos = fwrite (&v[2], sizeof(int), 3, f); /* Para escribir el primer elemento de v, valen las 2 instrucciones siguientes */ fwrite (v, sizeof (int), 1, f); fwrite (&v[0], sizeof(int), 1, f); </pre>	
---	--

```

//***** MENU JUEGO BARQUITOS ***** //
#include <stdio.h>
#include <stdlib.h> // para aleatorio
#include <time.h>
#include <string.h>

void mostrar_menu(); //presenta y muestra menu
void instrucciones(char nom_archivo[20]); //muestra
archivo
void mostrar_tablero(char nom[20]); //dibuja tablero
con nombre
void poner_barcos(); //pide barcos al jugador
char barcos_juga[10][10]; // matriz barcos jugador
char barcos_compu[10][10]; // matriz barcos computadora
int main ()
{
    int opci=1;
    do
    {
        mostrar_menu();
        scanf("%d", &opci);
        switch (opci)
        {
            case 1:
                instrucciones("help.txt");//muestra archivo
                break;
            case 2:
                mostrar_tablero("TABLERO DEL JUGADOR");
                poner_barcos();
                break;
        }
    }
    while(opci>0);
    return 0;
}

void mostrar_menu()
{
    system("cls"); //limpia
    printf("\n");
    printf("//***** BARQUITOS *****\n");
    printf("-----> juego de estrategia <-----\n");
    printf("\n");
    printf("Escoge opcion:\n");
    printf(" 1- Instrucciones\n");
    printf(" 2- Establecer barcos\n");
    printf(" 3- Jugar\n");
    printf(" 0- salir\n");
    printf("\n");
}

void instrucciones(char nom_archivo[20])
{
    FILE *fichero;
    char letra;
    system("cls"); //limpia
    fichero =
    fopen(nom_archivo,"r");//fopen("help.txt","r");
    if (fichero==NULL)
    {
        printf( "No se puede abrir el fichero %s\n",nom_archivo );
        exit( 1 );
    }
}

```

```

printf("Contenido del fichero %s n",nom_archivo);
letra=getc(fichero);
while (feof(fichero)==0)
{
    printf( "%c",letra );
    letra=getc(fichero);
}

if (fclose(fichero)!=0)
printf( "Problemas al cerrar el fichero
%s\n",nom_archivo);
system("pause");
}
void mostrar_tablero(char nom[20])
{
    //int i;
    system("cls"); //limpia

    printf("\n **** %s ****\n",nom);
    printf(" _____\n");
    printf("  A B C D E F G H I J \n");
    printf(" _____\n");
    /*for (i = 0; i < 10; i++)
    {
        if (i==9) {printf ("%d| \n", i+1);}
        else
        {printf (" %d| \n", i+1);}
    }
    printf(" _____\n");
    printf("\n");*/
}

void poner_barcos()
{
    int col,fil,i;
    bool ocupado;
    char terminado;
    do
    {
        ocupado=false;
        printf("\n numero de columna:");
        scanf("%d", &col);
        printf("\n Numero de fila:");
        scanf("%d", &fil);
        col--; //bajamos la numeración para la matriz
        fil--;
        if (barcos_juga[col][fil]=='X')
        {
            printf("\n ya esta ocupado\n");//comprobamos si
            ya hay uno...
            system("pause");
        }
        else barcos_juga[col][fil]='X';
    }

    mostrar_tablero("TABLERO DEL JUGADOR");
    for (fil = 0; fil < 10; fil++)
    {
        printf ("\n %d", fil+1);
        for (col = 0; col < 10; col++)
            printf (" %c ", barcos_juga[col][fil]);
        printf("\n \n Has terminado(S/N)? : ");
        scanf("%s", &terminado);
    }
    while (terminado!='S' && terminado!='s');
}

```

Punteros

Los punteros nos permiten acceder directamente a cualquier parte de la memoria. Esto da a los programas C una gran potencia. Sin embargo son una fuente ilimitada de errores.

Direcciones de variables

Al declarar una variable estamos diciendo al ordenador que nos reserve una parte de la memoria para almacenarla. Cada vez que ejecutemos el programa la variable se almacenará en un sitio diferente, eso no lo podemos controlar, depende de la memoria disponible y otros factores misteriosos. Puede que se almacene en el mismo sitio, pero es mejor no fiarse. Dependiendo del tipo de variable que declaremos el ordenador nos reservará más o menos memoria. Como vimos en el capítulo de tipos de datos cada tipo de variable ocupa más o menos bytes. Por ejemplo si declaramos un char, el ordenador nos reserva 1 byte (8 bits). Cuando finaliza el programa todo el espacio reservado queda libre.

Existe una forma de saber qué direcciones nos ha reservado el ordenador. Se trata de usar el operador **&** (operador de dirección). Vamos a ver un ejemplo: Declaramos la variable 'a' y obtenemos su valor y dirección.

```
#include <stdio.h>
int main()
{
    int a;
    a = 10;
    printf( "Dirección de a = %p, valor de a = %i\n", &a, a );
}
```

Para mostrar la dirección de la variable usamos **%p** (puntero) en lugar de **%i**, sirve para escribir direcciones de punteros y variables. El valor se muestra en hexadecimal.

No hay que confundir el valor de la variable con la dirección donde está almacenada la variable. La variable 'a' está almacenada en un lugar determinado de la memoria, ese lugar no cambia mientras se ejecuta el programa. El valor de la variable puede cambiar a lo largo del programa, lo cambiamos nosotros. Ese valor está almacenado en la dirección de la variable. El nombre de la variable es equivalente a poner un nombre a una zona de la memoria. Cuando en el programa escribimos 'a', en realidad estamos diciendo, "el valor que está almacenado en la dirección de memoria a la que llamamos 'a'"

Qué son los punteros

Un puntero es una variable para almacenar direcciones de memoria de otra variable.

El resultado: Dirección de numero = 00003, valor de numero = 43

```
#include <stdio.h>
int main()
{
    int numero;

    numero = 43;
    printf( "Dirección de numero = %p, valor de
numero = %i\n", &numero, numero );
}
```

```
#include <stdio.h>
int main()
{
    int numero;
    int *punt;

    numero = 43;
    punt = &numero;
    printf( "Dirección de numero = %p, valor de
numero = %i\n", *punt, numero );
}
```

Declaración de un puntero: depende del tipo de dato al que queramos apuntar. En general la declaración es:
tipo_de_dato *nombre_del_puntero;

Si queremos que almacene la dirección de una variable char: char *punt;

Ejemplo para declarar un puntero que apuntará a una variable del tipo int: int *punt;

El * (asterisco) sirve para indicar que se trata de un puntero, debe ir justo antes del nombre de la variable, sin espacios. En la variable punt sólo se pueden guardar direcciones de memoria, no se pueden guardar datos.

Utilidad de los punteros: Pueden servir para argumentos (o parámetros) a una función y modificarlos. También permiten el manejo de cadenas y de arrays. Otro uso importante es que nos permiten acceder directamente a la pantalla, al teclado y a todos los componentes del ordenador.

Comprobar si dos variables son iguales usando punteros:

Punteros como argumentos de funciones:

```
#include <stdio.h>

int main()
{
    int a, b;
    int *punt1, *punt2;

    a = 5; b = 5;
    punt1 = &a; punt2 = &b;

    if ( *punt1 == *punt2 )
        printf( "Son iguales\n" );
}
```

```
#include <stdio.h>

int suma( int *a, int b )
{
    int c;
    c = *a + b;
    *a = 0;
    return c;
}

int main()
{
    int var1, var2, resultado;

    var1 = 5; var2 = 8;
    resultado = suma(&var1, var2);
    printf( "La suma es: %i y a vale: %i\n", resultado , var1 );
}
```

invertir el orden de los elementos de un vector de enteros, usando sólo punteros, sin variables auxiliares.

// Programa que invierte el orden de un vector

```
#include <iostream>
using namespace std;
void Mostrar(int*, int);
void Intercambia(int*, int*);

int main() {
    int vector[10] = { 2,5,6,7,9,12,35,67,88,99 };
    int *p, *q;
    Mostrar(vector, 10); // Mostrar estado inicial
    p = vector; // Primer elemento
    q = &vector[9]; // Ultimo elemento

    while(p < q) { // Bucle de intercambio
        Intercambia(p++, q--);
    }
    Mostrar(vector, 10); // Mostrar estado final
    return 0;
}

void Mostrar(int *v, int n) {
    int *f = &v[n];
    // Puntero a posición siguiente al último elemento
    while(v < f) {
        cout << *v << " ";
        v++;
    }
    cout << endl;
}

void Intercambia(int *p, int *q) {
    // Intercambiar sin usar variables auxiliares:
    *p += *q;
    *q = *p-*q;
    *p -= *q;
}
```

Estructuras (registros)

- ▶ Un registro es una agrupación de datos, no necesariamente del mismo tipo. Se definen con la palabra "struct" y son accesibles a través de un identificador o variable del tipo struct. Ejemplo: *struct personas cliente;*
- ▶ Una estructura es como un registro de una base de datos y un array de estructura como la tabla completa de registros.
- ▶ A cada campo dentro de la estructura la llamamos miembro y a cada registro o estructura, elemento.
- ▶ Las estructuras pueden pasarse como argumentos de funciones. Las estructuras no pueden compararse entre si.
- ▶ Un vector o array contiene datos del mismo tipo, mientras que una estructura puede agrupar datos de diferente tipo
- ▶ La estructura es un tipo de dato definido por el usuario, que se debe declarar antes de que se pueda utilizar.

Inicializar una estructura: Existen dos formas de inicializar una estructura: Dentro de la sección de código de su programa declarar la variable indicado primero el tipo de datos (struct { ... }) y después el nombre que tendrá esa variable (personas) o también podemos declarar primero cómo van a ser nuestros registros, y más adelante definir variables de ese tipo. *struct mystruct variable; struct personas cliente;*

Se puede asignar un alias usando la palabra clave **typedef**: *typedef struct { ... } mystruct;*

Acceso a los campos: Utilizando el operador punto entre la variable y el miembro o campo: *variable.campo;*
ejemplos: *amigo.edad=27; printf("%s", cliente.nombre); if (strcmp(cliente.nombre,"Juan")==0)*

Arrays de Estructura: Se puede crear un array de estructura tal como se crea un array de otros tipos. Los arrays de estructuras son idóneos para almacenar un archivo completo de empleados o cualquier otro conjunto de datos adaptados a la estructura. La declaración de un array de estructuras es similar a cualquier array.

```
#include <stdio.h>
struct estructura_amigo {
    char nombre[30];
    char apellido[40];
    char telefono[10];
    int edad;
}; //<- poner al final el ;
struct estructura_amigo amigo = {
    "Juanjo",
    "Lopez",
    "592-0483",
    30
};
int main()
{
    printf( "%s tiene ", amigo.apellido );
    printf( "%i años ", amigo.edad );
    printf( "y su telefono es el %s.\n" , amigo.telefono );
}
```

Se puede crear la estructura y luego el array de estructuras de la forma siguiente:

```
struct estructura_amigo amigo[] =
{
    "Juanjo", "Lopez", "504-4342", 30,
    "Marcos", "Gamendez", "405-4823", 42,
    "Ana", "Martinez", "533-5694", 20
};
```

NOTA: En algunos compiladores es posible que se exija poner antes de **struct** la palabra **static**

- ▶ A continuación, en el primer ejemplo tenemos una estructura con un solo registro.
- ▶ En el segundo ejemplo, para tener varios registros es necesario tener un *Array* de ELEMENTOS. Es decir, para guardar la información de varios amigos, necesitamos declarar un array de estructura.

Estructura con un solo registro

```
#include <stdio.h>

struct estructura_amigo // Definimos la estructura
{ char nombre[30]; //Definimos los campos
  char apellido[40];
  char telefono[10];
  char edad; // no sobrepasará los 256 años
};

//declaramos la variable tipo struct llamada amigo
struct estructura_amigo amigo;

//o también todo junto estructura y variable:
struct
{ char nombre[30];
  char apellido[40];
  char telefono[10];
  char edad; }
amigo;

int main()
{
    printf( "Escribe el nombre del amigo: " );
    scanf( "%s", &amigo.nombre );
    printf( "Escribe el apellido del amigo: " );
    scanf( "%s", &amigo.apellido );
    printf( "Escribe su numero de telefono: " );
    scanf( "%s", &amigo.telefono );
    fflush( stdout );
    printf( "El amigo %s %s tiene el numero de tel:
    %s.\n",
    amigo.nombre,amigo.apellido, amigo.telefono );
}
```

Estructura con 3 registros

```
#include <stdio.h>
#define ELEMENTOS 3 //pondremos 3 amigos

struct estructura_amigo {
    char nombre[30];
    char apellido[40];
    char telefono[10];
    int edad;
};

struct estructura_amigo amigo[ELEMENTOS];
int main()
{
    for(int i=0; i<ELEMENTOS; i++ )
    {
        printf("\nDatos del amigo número %i:\n",i+1);
        printf( "Nombre: " );
        scanf("%s", &amigo[i].nombre);
        printf( "Apellido: " );
        scanf("%s", &amigo[i].apellido);
        printf( "Teléfono: " );
        scanf("%s", &amigo[i].telefono);
        printf( "Edad: " );
        scanf("%d", &amigo[i].edad );
        while(getchar()!='\n'); //→ vacia buffer
    }
    /*** Ahora imprimimos sus datos ***/
    for( i=0; i<ELEMENTOS; i++ )
    {
        printf( "El amigo %s ", amigo[i].nombre );
        printf( "%s tiene ", amigo[i].apellido );
        printf( "%i años ", amigo[i].edad );
        printf( "y telefono %s.\n" , amigo[i].telefono );
    }
}
```

fflush(stdout): El bufer de la salida se vacía (se envían los datos al momento)

<pre> Estructuras con punteros. #include <stdio.h> struct estructura_amigo { char nombre[30]; char apellido[40]; int edad; }; struct estructura_amigo amigo, *p_amigo; int main() { p_amigo = &amigo; /* Introducimos los datos mediante punteros */ printf("Nombre: "); gets(p_amigo->nombre); printf("Apellido: "); gets(p_amigo->apellido); printf("Edad: "); scanf("%i", &p_amigo->edad); /* Mostramos los datos */ printf("El amigo %s ", p_amigo->nombre); printf("%s tiene ", p_amigo->apellido); printf("%i años.\n", p_amigo->edad); } </pre>	<pre> Pasar estructuras a una función #include <stdio.h> struct estructura_amigo { char nombre[30]; char apellido[40]; char telefono[10]; int edad; }; int suma(struct estructura_amigo arg_amigo) { return arg_amigo.edad+20; } int main() { struct estructura_amigo amigo = { "Juanjo", "López", "592-0483", 30 }; printf("%s tiene ", amigo.apellido); printf("%i años ", amigo.edad); printf("y dentro de 20 años tendrá %i.\n", suma(amigo)); system("PAUSE"); } </pre>
--	--

Estructuras dentro de estructuras (Anidadas):

Permite tener la estructura de datos más ordenada

Para acceder al número de cd de cierto título usaríamos lo siguiente:
inventario.existencias.cd

y para acceder al nombre del proveedor:
inventario.proveedor.nombre

```

struct estruc_existencias {
    int discos;
    int cintas;
    int cd;
};

struct estruc_proveedor {
    char nombre_proveedor[40];
    char telefono_proveedor[10];
    char direccion_proveedor[100];
};

struct estruc_inventario {
    char titulo[30];
    char autor[40];
    estruc_existencias existencias;
    estruc_proveedor proveedor;
} inventario;

```

Funciones complementarias a struct:

- **Typedef:**
 Nos permite la creación de nuevos tipos de datos. Definición de tipos. Permite utilizar el nombre del struct como nuevo nombre de tipo al declarar las variables o los campos de otros structs. Asignar un nombre alternativo a tipos existentes.
Sintaxis: **typedef** <tipo> <identificador>
 (nombre de la variable y el campo separados por un punto)
 Un uso típico es la redefinición de tipos estructurados.

```

typedef struct // estructura anónima
{ char nombre[80];
  char sexo;
  int edad;
} Persona; // se declara el tipo Persona
...
Persona p; //estructura p del tipo persona
...
p.edad = 44;

```

- **Enum:**
 Permite numerar los campos (0, 1, 2, ...) para poder utilizarlos en cases, por ejemplo. Aunque podemos inicializar nuestra primer constante.

```

enum DiasSemanas
{
    Domingo = 1,
    Lunes,
    Martes,
    Miercoles,
    Jueves,
    Viernes,
    Sabado
};

```

- **Union:**
 Es similar a la "estructura". Para ahorrar memoria, los miembros ocupan el mismo área de memoria. En el ejemplo de la figura, se puede acceder a "u.i" o a "u.d", pero no a ambos al mismo tiempo.

```

union {
    int i;
    double d;
} u;

```

Guardar un Struct en un fichero.

Es preferible guardar los struct en ficheros binarios porque permiten guardar datos de distintos tipos. Si un struct se guarda en modo texto, habría que escribir cada campo como una línea de texto con `fprintf` para que convirtiera los campos de tipo numéricos en tipo de texto. (→ *Veáse ejemplos de grabación de structs en Ficheros binarios*)

Recordamos que para grabar y leer en ficheros binarios se utilizan las funciones:

- **Fwrite:** Permite escribir en un fichero. *Sintaxis:* `fwrite(buffer, tamaño, número, *pfichero);`
- **Fread:** Para sacar información de un fichero. *Sintaxis:* `fread(buffer, tamaño, número, *pfichero);`

Ejemplo:

```

/** Escribir y leer cinco registros en un fichero


---


/** Escribir y leer varios registros en un fichero
#include <stdio.h>
#define NUM 3
struct estruct_notas {
    char nom[30];
    int nota;
};
struct estruct_notas notas;
void crear_fichero();
void leer_fichero();
//-----
int main(void)
{
    crear_fichero();
    leer_fichero();
}
//-----
void crear_fichero ()
{
    FILE *fich;
    int i= 0;
    if ((fich = fopen("f_notas.dat", "wb")) == NULL)
        printf ("Error en apertura para escritura\n");
    else {
        for(i = 0; i < NUM; i++)
        {
            printf("Nombre: ");
            scanf("%s", &notas.nom);
            printf("Nota: ");
            scanf("%d", &notas.nota);
            fwrite(&notas, sizeof(notas), 1, fich);
        }
        fclose (fich);
    }
}
void leer_fichero ()
{
    FILE *fich;
    if ((fich = fopen("f_notas.dat", "rb")) == NULL)
        printf ( "Error de lectura \n " );
    else
    {
        fread (&notas, sizeof(notas), 1, fich);
        while (!feof(fich))
        {
            printf ("%s: %d \n" , notas.nom, notas.nota);
            fread (&notas, sizeof(notas), 1, fich);
        }
        fclose (fich);
    }
}

```

// Notas - Ejercicio Array con struct y menu

```
#include <stdio.h>
#include <stdlib.h>
const MAX_FICHAS=100; //O #define MAX_FICHAS 100
int opcion;
int num_ficha=0;
int notaTemporal;
char textoTemporal[30];
int i=0;
void cargadatos(char modo[2]);
struct estructura_notas{
    char nombre[30];
    int nota;
};
struct estructura_notas notas[]; //variable del tipo struct

int main()
{
do
    { // Menu principal
    printf("\n");
    printf("Escoja una opción:\n");
    printf( "1.- Añadir datos\n" );
    printf( "2.- Mostrar todos\n" );
    printf( "3.- Mostrar los mayores\n" );
    printf( "4.- Buscar nombre\n" );
    printf( "5.- Guardar datos\n" );
    printf( "6.- Abrir datos\n" );
    printf( "7.- Salir\n" );
    scanf("%d",&opcion);
    switch(opcion)// Hacemos según la opción
    {
case 1: // Añadir un dato nuevo
        if (num_ficha < MAX_FICHAS) // Si queda hueco
        {
            printf("Introduce el nombre: \n");
            scanf("%s",& notas[num_ficha].nombre);
            printf("Introduce nota ");
            scanf("%d",&notas[num_ficha].nota);
            num_ficha++; // Y tenemos una ficha más
        }
        else // Si no hay hueco para más fichas
            printf("Máximo de fichas alcanzado");
        break;
case 2: // Mostrar todos
        for (i=0; i<num_ficha; i++)
        {printf("Nombre: %s ",notas[i].nombre);
        printf("Nota: %d \n" ,notas[i].nota);}
        break;
case 3: // Mostrar según nota
        printf("¿A partir de que nota? ");
        scanf("%d",&notaTemporal);
        for (i=0; i<num_ficha; i++)
        if (notas[i].nota >= notaTemporal)
        {printf("Nombre: %s \n",notas[i].nombre);
        printf("Nota: %d \n" ,notas[i].nota);}
        break;
case 4: // Ver un nombre
        printf("Nombre a buscar:");
        scanf("%s",&textoTemporal);
        for (i=0; i<num_ficha; i++)
        if (strcmp(notas[i].nombre,textoTemporal)==0)
            {printf("Nombre: %s \n",notas[i].nombre);
            printf("Nota: %d \n" ,notas[i].nota);}
            break;
case 5: // Guardar datos
            cargadatos("wb");
            break;
case 6: // Abrir datos
            cargadatos("rb");
            break;
case 7: // Salir: avisamos de que salimos
            printf("Fin del programa");
            break;
default: // Otra opcion: no válida
            printf("Opción desconocida!");
            break;
        }
    }
    while (opcion != 7); // Si es 5, terminamos
}

void cargadatos(char modo[2])
{
    FILE *fichero;
    //fichero = fopen( "nombres.dat", modo );
    if ((fichero = fopen("f_notas.dat", modo)) == NULL)
        printf ("Error en apertura\n");
    else {
        if (modo == "wb")//grabar
        {
            i=0;
            for (i=0; i<num_ficha;i++)
            {
                fwrite(&notas[i].nombre, sizeof(notas[i].nombre),
                num_ficha, fichero );
                fwrite(&notas[i].nota,
                sizeof(notas[i].nota),num_ficha, fichero );
                printf("%s ",notas[i].nombre);
                printf(" %d \n",notas[i].nota);
            }
        }
        if (modo=="rb")
        {
            printf( "Lectura del fichero\n" );
            fread(&notas[i].nombre, sizeof(notas[i].nombre), 1,
            fichero );
            fread(&notas[i].nota, sizeof(notas[i].nota), 1,
            fichero );
            while (!feof(fichero)) {
                fread(&notas[i].nombre, sizeof(notas[i].nombre), 1,
                fichero );
                fread(&notas[i].nota, sizeof(notas[i].nombre), 1,
                fichero );
                {
                    printf( "Nombre: %s ", notas[i].nombre );
                    printf( "Nota: %d \n", notas[i].nota);
                }
                i++;
            }
            num_ficha=i;
        }
        fclose( fichero );
    }
}
```

Librerías y complementos:

Creación de una librería en C.

- Agrupar varias funciones útiles en un mismo archivo de texto.
- Guardar el fichero con extensión .h, por ejemplo milibreria.h:

Se puede guardar en la carpeta **include** del compilador.

Otra opción es guardar el fichero en la misma carpeta del código que queramos compilar.

Dependiendo de este último paso tendremos...

Colocar en la cabecera del programa, junto a los llamamientos de otras bibliotecas:

#include <milibreria.h> → Cuando el fichero se encuentre en la carpeta include de nuestro compilador.

#include "milibreria.h" → Cuando el fichero esté en el mismo directorio que el archivo que queremos compilar.

```

/***** VAIXELLS *****/
// Constantes modificables:
#define TOCADO 'T' // simbolos
a representar en los tableros.
#define HUNDIDO 'H'
#define AGUA 'O'
// Librerias:
#include <stdio.h>
#include <stdlib.h> // per aleatori
#include <time.h>
#include <string.h>
#include <ctype.h>

//--> funcions
void mostrafitxer(char nom_fitxer[20]); //muestra
archivo
void mostrartauler(char nom[20]); //dibuja tablero con
nombre
void ficar_vaixelles(); //pide barcos al jugador
void jugar(); //jugar conta maquina
void opcions(); //ajustes
//char letra; //variable letra
//--> variables
char mat_juga[10][10]; // matriu vaixelles jugador
char mat_compu[10][10]; // matriu vaixelles
computadora
int lang=0; //variable codi idioma 0=esp 1=cat 2=eng
int num_barcos=3; //variable nº vaixelles per defecte
int main ()
{
    int opci=1;
    do
    {
        system("cls");
        if (lang==0) mostrafitxer("menu_esp.lng");
        else if (lang==1) mostrafitxer("menu_cat.lng");
        else if (lang==2) mostrafitxer("menu_eng.lng");
        else printf("No es troba el fitxer");
        scanf("%d", &opci);
        switch (opci)
        {
            case 1:
                system("cls");
                if (lang==0) mostrafitxer("help_esp.lng");
                else if (lang==1) mostrafitxer("help_cat.lng");
                else if (lang==2) mostrafitxer("help_eng.lng");
                else printf("No es troba el fitxer");
                system("pause");
                break;
            case 2:
                ficar_vaixelles();
                break;
            case 3:
                jugar();
                break;
            case 4:
                opcions();
                break;
        }
    } while(opci>0);
    return 0;
}

void mostrafitxer(char nom_fitxer[20])
{
    FILE *fitxer;
    char letra;
    system("cls"); //limpia
    fitxer = fopen(nom_fitxer,"r");//fopen("help.txt","r");
    if (fitxer==NULL)
    {
        printf( "No es pot obrir el fitxer %s\n",nom_fitxer );
        exit( 1 );
    }
    fscanf(fitxer, "%c",&letra);
    //letra=getc(fitxer);
    while (feof(fitxer)==0)
    {
        printf( "%c",letra );
        fscanf(fitxer, "%c",&letra);
        //letra=fscanf (fitxer, "%c");
    }
    fclose(fitxer);
}

void mostrartauler(char nom[20])//un texte de màxim
20 caracters
{
    //nom serà JUGADOR o ENEMIG
    printf("\n **** TAULER %s ****\n",nom);
    printf(" _____\n");
    printf(" A B C D E F G H I J \n");
    printf(" _____\n");
    for (int fil = 0; fil < 10; fil++)
    {
        printf ("\n %d", fil+1); //salto de renglón
        for (int col = 0; col < 10; col++)
        {
            if (strcmp(nom, "JUGADOR") == 0) printf (" %c ",
mat_juga[col][fil]); // si se llama desde jugador
compara string y muestra celda
            else if ((strcmp(nom, "ENEMIG") == 0) &&
(mat_compu[col][fil]!='X')) printf (" %c ",
mat_compu[col][fil]); //enemigo si no es barco
        }
        printf("\n _____\n");
    }
}

void ficar_vaixelles()
{
    int col,fil,barco;
    char c; // caracteres leídos desde el teclado
    system("cls"); //limpia antes de mostrar
    //----- muestra tablero vacio -----
    mostrartauler("JUGADOR");
    //----- pide barcos-----
    for (int barco = 1; barco < num_barcos+1; barco++)
    {
        printf("\n Pon el barco numero %d de
%d",barco,num_barcos);
        printf("\n lletra de la columna (A-J): ");
        scanf("%c", &c);
        col=toupper(c)-64; //caracteres ASCII del alfabeto
en mayusc. van del 65(A) al 90(Z)
        printf("Numero de la fila (1-10): ");
        scanf("%d", &fil);
        col--; //bajamos la numeración para la matriz
        fil--;
        if (col>9 || fil>9 || col<0 || fil<0) //si está
fuera del tablero mensaje de error
        {
            printf("\n casilla incorrecta. Prueba de
nuevo...\n");
            system("pause");
            barco--;
        }
        else if (mat_juga[col][fil]=='X') //comprobamos
si ya hay una X en el mismo sitio...
        {
            printf("\n ya esta ocupado. Prueba de
nuevo...\n");
            system("pause");
            barco--;
        }
        else mat_juga[col][fil]='X';
    }
    //----- muestra tablero jugador con los barcos puestos-
-----
    system("cls"); //limpia antes de mostrar
    mostrartauler("JUGADOR");
    //----- poner barcos enemigo/computadora -----
    --
    for (int barco = 1; barco < num_barcos+1; barco++)
    {
        col = rand()%9; //valor de fila y columna aleatoria
        (random)
        fil = rand()%9;
        if (mat_compu[col][fil]=='X') barco--; // si ya hay
una X en el mismo sitio repetimos...
        else mat_compu[col][fil]='X';
    }
}

```

```

}
printf ("\n Barcos listos.");
system("pause");
}

void opciones()
{
system("cls"); //limpia
printf(" |-----|\n");
printf(" |          AJUSTES          | \n");
printf(" |-----|\n");
printf("\n");
printf("\n Escoja numero de idioma (0:ESP 1:CAT
2:ENG): ");
scanf("%d", &lang);
printf("\n Escoja numero de barcos (1 a 10) ");
scanf("%d", &num_barcos);
}

void jugar()
{
int col,fil; //vars numeros de fila y columna a pedir
char c; // var para pedir letra de columna o salir
int tirada=0; //var numero de tirada
char terminado='N'; //variable para salir
char texto[20];
system("cls"); //limpia antes de mostrar
mostrartauler("JUGADOR");
mostrartauler("ENEMIG");
while (toupper(c)!='S') //o terminado!='S' &&
terminado!='s';
{
tirada++; //incrementa nº de tirada
printf("\nMoviment del jugador num. %d (S per
sortir): ",tirada);
printf("\nLletra de la columna (A-J): ");
scanf("%c", &c);
if (toupper(c)=='S') break;
col=toupper(c)-64; //caracteres ASCII del alfabeto
en mayusc. van del 65(A) al 90(Z)
printf("Numero de la fila (1-10): ");
scanf("%d", &fil);
col--;//bajamos la numeración para la matriz
fil--;
if (col>9 || fil>9 || col<0 || fil<0) //si está
fuera del tablero mensaje de error
{
printf("\n Casilla incorrecta. Prueba de
nuevo...\n");
system("pause");
tirada--;
}
else if (mat_compu[col][fil]=='X') //comprobamos
si ya hay una X en el mismo sitio...
{
mat_compu[col][fil]=HUNDIDO; //mostramos
símbolo 'H';
strcpy(texto, "HUNDIDO"); //introduce en texto
la cadena HUNDIDO
//system("pause");
}
}
}

```

```

else
{
mat_compu[col][fil]=AGUA; //mostramos
símbolo 'O';
strcpy(texto, "AGUA"); //introduce en texto la
cadena AGUA
}
system("cls");
mostrartauler("JUGADOR");
mostrartauler("ENEMIG");
printf("----> %s\n",texto);
system("pause");
//-----movimiento enemigo/computadora-----
-----
col = rand()%9; //valor de fila y columna
aleatoria (random)
fil = rand()%9;
if (mat_juga[col][fil]!='X') //comprobamos si ya
hay una X en el mismo sitio...
{
mat_juga[col][fil]='H';
strcpy(texto, "HUNDIDO");
}
else
{
mat_juga[col][fil]='O';
strcpy(texto, "AGUA");
}
system("cls"); //limpia antes de mostrar
mostrartauler("JUGADOR");
mostrartauler("ENEMIG");
printf("\nMoviment enemig num %d: \n",tirada);
col++; //sumamos 1 para empezar en 1
fil++;
c=toupper(col+64);
printf("Celda: %c%d ",c,fil);
printf("----> %s\n",texto);
system("pause");
}
printf("\n VAIXELLS ENEMIGS \n");
printf(" _____\n");
printf("  A B C D E F G H I J \n");
printf(" _____\n");
for (int fil = 0; fil < 10; fil++)
{
printf ("\n %d", fil+1); //salto de renglón
for (int col = 0; col < 10; col++)
{
// if (strcmp(nom, "JUGADOR") == 0) printf (" %c ",
mat_juga[col][fil]); // si se llama desde jugador
compara string y muestra celda
// else if ((strcmp(nom, "ENEMIG") == 0) &&
(mat_compu[col][fil]!='X')) printf (" %c ",
mat_compu[col][fil]); //enemigo si no es barco
printf (" %c ", mat_compu[col][fil]); //todo
}
}
printf("\n _____\n");
system("pause");
}
}

```

Tratamiento de Secuencias de búsqueda

Secuencia de búsqueda mediante centinela:

Un centinela en un bucle de búsqueda es normalmente el primer valor no válido para que finalice el bucle.

Recorrido:

Mismo tratamiento para todos los elementos de la secuencia. Termina al llegar al final de la secuencia

Ej.- Mostrar los elementos de una secuencia, sumar los números de una secuencia, ¿par o impar cada número de una secuencia?,

Búsqueda

Recorrido hasta encontrar un elemento buscado

Ej.- Localizar el primer número que sea mayor que 1.000

Termina al localizar el primer elemento que cumple la condición o al llegar al final de la secuencia (si no encontrado)

Esquemas:

Recorrido:

Un mismo tratamiento a todos los elementos. No sabemos cuántos elementos hay No podemos implementar con for

Esquema:

- › Inicialización
- › Mientras no se llegue al final de la secuencia:
- › Obtener el siguiente elemento
- › Procesar el elemento
- › Finalización

Recorrido con centinela:

- › Inicialización
- › Obtener el primer elemento
- › Mientras no sea el centinela:
- › Procesar el elemento
- › Obtener el siguiente elemento
- › Finalización

Búsqueda con centinela:

- › Inicialización
- › Obtener el primer elemento
- › Mientras ni encontrado ni el centinela:
- › Obtener el siguiente elemento
- › Finalización (¿encontrado?)

Algoritmos de ordenación

Algoritmos de ordenación simples:

- Selección / Selección directa: Intercambiar un elemento de la lista por el menor.

- Burbuja / Bombolla/Bubblesort: Intercambiar por parejas los elementos de una lista de menor a mayor.

- Inserción / Inserción directa: Ir añadiendo elementos en una tabla según su posición

- Ordenamiento Rápido (Quicksort): El más rápido. Se basa en ir dividiendo la lista en dos mitades recursivamente

- **Método de burbuja:** Intercambiar cada pareja consecutiva que no esté ordenada:

Para $i=1$ hasta $n-1$

Para $j=i+1$ hasta n

Si $A[i] > A[j]$

Intercambiar ($A[i], A[j]$)

- **Selección:** En cada pasada busca el menor, y lo intercambia al final de la pasada.

Para $i=1$ hasta $n-1$

menor = i

Para $j=i+1$ hasta n

Si $A[j] < A[\text{menor}]$

menor = j

Si menor $\neq i$

Intercambiar ($A[i], A[\text{menor}]$)

- **Inserción:** Comparar cada elemento con los anteriores -que ya están ordenados- y desplazarlo hasta su posición correcta

Para $i=2$ hasta n

$j=i-1$

mientras ($j \geq 1$) y ($A[j] > A[j+1]$)

Intercambiar ($A[j], A[j+1]$)

$j = j - 1$

Funciones más usuales según Librerías:

Stdio:

fclose	Cierra un fichero a través de su puntero.
feof	Comprueba el indicador de final de fichero. EOF contante de fin de fichero
ferror	Comprueba el indicador de errores.
fflush	envía el dato aún sin escribir al entorno local
fgetc	Devuelve un carácter de un fichero.
fgetpos	Devuelve la posición actual del fichero.
fgets	Consigue una cadena de caracteres de un fichero.
fopen	Abre un fichero para lectura, para escritura/reescritura o para adición.
fprintf	Introduce dato en el archivo o fichero
fputc	Escribe un carácter en un fichero.
fputs	Escribe una cadena de caracteres en un fichero.
fread	lee diferentes tamaños de datos de un fichero.
fscanf	Recoge dato del archivo o fichero
fseek	Sitúa el puntero de un fichero en una posición aleatoria.
fsetpos	Cambia la posición actual de un fichero.
ftell	Devuelve la posición actual del fichero como número de bytes.
fwrite	Envía, desde el array apuntado por puntero a un fichero, avanzado por el número de caracteres escritos.
getc	Devuelve un carácter desde un fichero.
getchar	Devuelve un carácter desde la entrada estándar
gets	Lee caracteres de entrada hasta que encuentra un salto de línea, y los almacena en un único argumento.
printf	Mostar o imprimir salidas de datos. Tambien: sprintf, snprintf, vprintf
putc	Escribe un carácter en un fichero.
putchar	Escribe un carácter en la salida estándar
puts	Escribe una cadena de caracteres en la salida estándar
remove	Elimina un fichero.
rename	Cambia al fichero de nombre.
rewind	Coloca el indicador de posición de fichero para el stream apuntado por stream al comienzo del fichero.
scanf	Utilizado para introducir entradas. También: sscanf, vfscanf, vscanf, vsscanf
tmpfile	Crea y abre un fichero temporal que es borrado cuando cerramos con la función fclose().

stdlib

abort	terminar ejecución anormalmente
abs	valor absoluto
atexit	registrar una función callback para la salida del programa
atof	(ascii to float) cadena de caracteres a coma flotante
atoi	(ascii to integer) cadena de caracteres a entero
atol	(ascii to long) cadena de caracteres a entero tamaño largo
bsearch	búsqueda binaria en un array
div	división entera o euclidiana
exit	(operating system)terminar ejecución del programa
free	Liberan memoria devolviéndola al heap
malloc	Reservan memoria dinámica del heap (montón o montículo) calloc, realloc
rand	Genera un número pseudo-aleatorio
srand	Inicio para el generador de números pseudo-aleatorios
strtod	(string to double) cadena de caracteres a coma flotante tamaño doble
strtol	(string to long) cadena de caracteres a entero largo
strtoul	(string to unsigned long) cadena de caracteres a entero largo sin signo (positivo)
system	ejecutar un comando externo

#include <string.h>

memchr	memcmp	memcpy	memmove	memset	strcat	strchr	strxfrm
strcmp	strcoll	strcpy	strcspn	strerror	strlen	strncat	
strncmp	strncpy	strpbrk	strrchr	strspn	strstr	strtok	

#include <ctype.h>

tolower	toupper	islower	isupper	isdigit	isalnum	isprint	isspace
---------	---------	---------	---------	---------	---------	---------	---------

#include <math.h>

Acos	Asin	atan	atan2	ceil	cos	cosh	tanh
Exp	Fabs	floor	fmod	frexp	ldexp	log	
log10	modf	pow	sin	sinh	sqrt	tan	

#include <time.h>

asctime	clock	ctime	difftime	Gmtime	localtime	mktime	time
strftime							

INDEX

Introducción a la Programación en C, C++ , C#	1
Estructuras básicas de control	6
Bucles o estructuras repetitivas	17
Matrices o arrays bidimensionales	34
Funciones y procedimientos	38
Ficheros: Lectura y escritura.....	43
Punteros	49
Estructuras (registros)	51
Librerías y complementos:	54
Algoritmos de ordenación	57