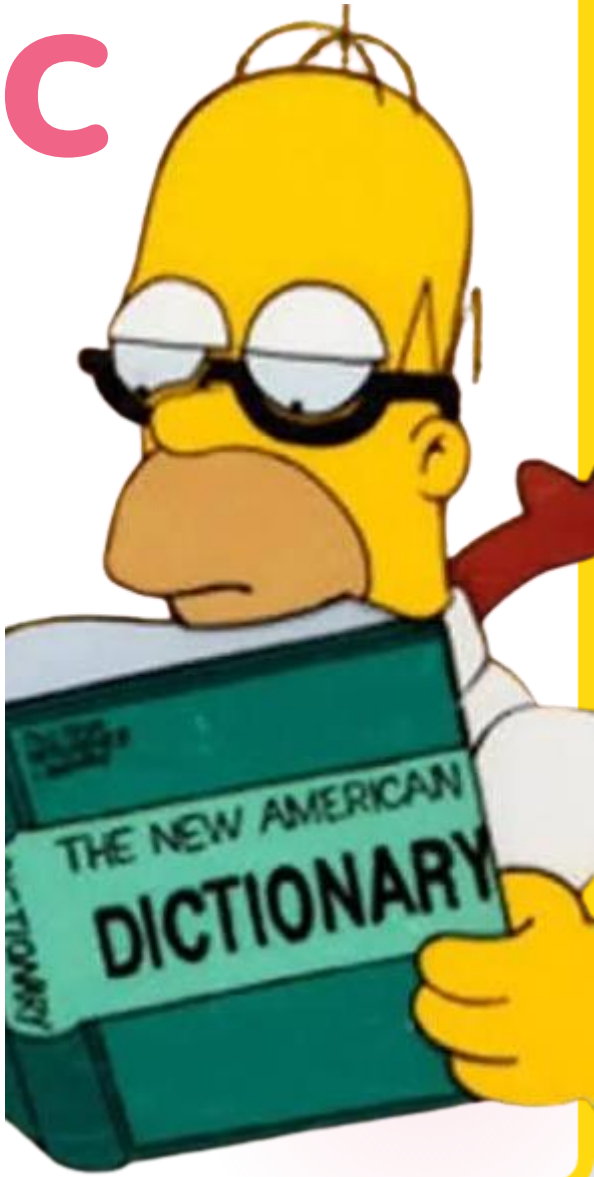
BIBLIOTECAS EN C

¿Qué son y para qué sirven?



¿QUÉ VEMOS HOY?

- **01.** ¿Qué es una biblioteca?
- **02.** Relación con lo que venimos viendo
- **03.** ¿Cómo armamos una biblioteca?
- **04.** Funciones privadas y públicas
- **05.** Ejemplo práctico

CON LO QUE SABEMOS...

Supónganse que quieren **vender su código** del Trabajo Práctico a alguien.

¿Cómo lo harían? ¿Cómo entregarían su código de forma segura y estructurada?



¿QUÉ ES UNA BIBLIOTECA?

Una biblioteca en C es un **contrato entre 2 partes**.

Por un lado, una parte proporciona funciones con ciertas condiciones preestablecidas.

Y la otra parte utiliza esas funciones sabiendo **qué hacen, pero no cómo lo hacen**.



¿LES SUENA A ALGO?



UNA REFLEXIÓN SOBRE ABSTRACCIÓN...

Pensemos en cómo venimos trabajando con nuestro código hasta ahora.

¿Qué pasa cuando empezamos a ocultar los detalles complejos?

CONDICIONES Y ABSTRACCIÓN

CONDICIONES PARA UNA FUNCIÓN

Ya sabemos hacer que una función tenga reglas.

Las **pre y post condiciones** nos dictan estrictamente qué necesita nuestra función para operar y qué garantiza que devuelve.

SABER EL QUÉ, PERO NO EL CÓMO

Este concepto suena bastante parecido a **caja negra y caja blanca**.

Sabemos que con ciertas entradas nos van a devolver ciertas salidas, pero ignoramos por completo cómo funciona internamente.

¿CÓMO SE ARMAN?



EL ARCHIVO .H

Contiene las **firmas** de las funciones. Es el índice o contrato público.



EL ARCHIVO .C

Contiene el **cuerpo** de estas funciones. Aquí vive la lógica real.



DISTRIBUCIÓN

Normalmente cuando damos el código, **solo se puede ver el .h** junto con el código compilado (.o).

TIPOS DE FUNCIONES

FUNCIONES PÚBLICAS

Teniendo en cuenta que los usuarios no siempre van a ver nuestro archivo .c, podemos pensar que las funciones declaradas en el .h son **públicas**.

Están diseñadas para ser consumidas desde el exterior.

FUNCIONES PRIVADAS

Dentro de nuestro archivo .c podemos tener funciones auxiliares que sirvan para la implementación de la biblioteca.

Como no se declaran en el .h, permanecen **ocultas** al usuario final.

VEAMOS UN EJEMPLO

MANOS A LA OBRA

Vamos a armar una biblioteca desde cero.



EL ARCHIVO .H (FIRMAS)

```h
#ifndef MI_BIBLIO_H
#define MI_BIBLIO_H

#include <stdbool.h>

//PRE: - .
//POST: Devuelve la suma del numero1 y el numero2.
int suma(int numero1, int numero2);

//PRE: - .
//POST: Devuelve la resta del numero1 y el numero2 en ese orden.
int resta(int numero1, int numero2);

//PRE: numero2 no puede ser 0.
//POST: Devuelve la división truncada del numero1 y el numero2.
int division(int numero1, int numero2);

#endif // MI_BIBLIO_H
```

HEADER GUARDS

```
#ifndef MI_BIBLIO_H
#define MI_BIBLIO_H
...
#endif
```

Sirven para no tener múltiples declaraciones de lo que está dentro de la biblioteca, protegiendo al compilador de redefiniciones.

EL ARCHIVO .C (IMPLEMENTACIÓN PÚBLICA)

```c
#include "mi_biblio.h"
#include <stdio.h>

const int ERROR = -1;

int suma(int numero1, int numero2){
    return numero1 + numero2;
}

int resta(int numero1, int numero2){
    return numero1 - numero2;
}
```

```c
bool es_divisor_0(int numero2){
    if (numero2 == 0){
        return false;
    }
    return true;
}

int division(int numero1, int numero2){
    if (!es_divisor_0(numero2)){
        return numero1 / numero2;
    }
    printf("Error: No se puede dividir por 0.");
    return ERROR;
}
```

CÓMO COMPILAR

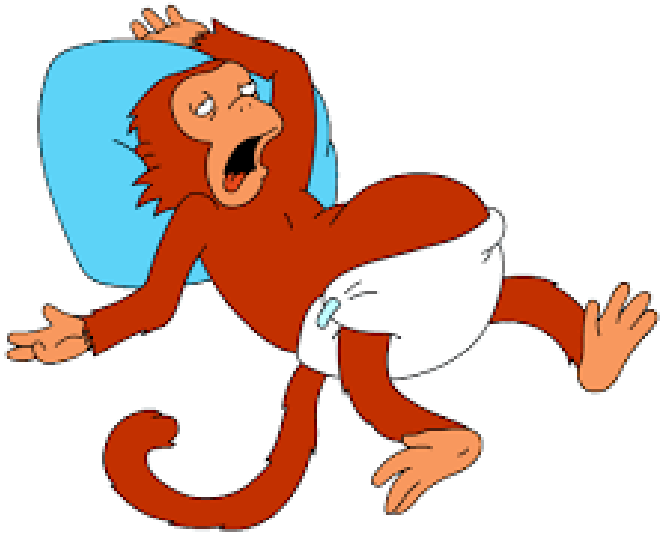
Para unir nuestro programa principal con nuestra biblioteca, usamos el compilador.

“cuentas.c" sería el archivo con el main que consume la biblioteca:

```
$ gcc cuentas.c mi_biblio.c mi_biblio.h -o ejemplo_clase
$ ./ejemplo_clase
```



¿PREGUNTAS?



FIN.