

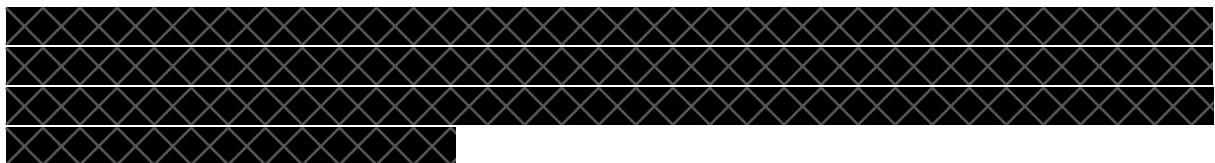
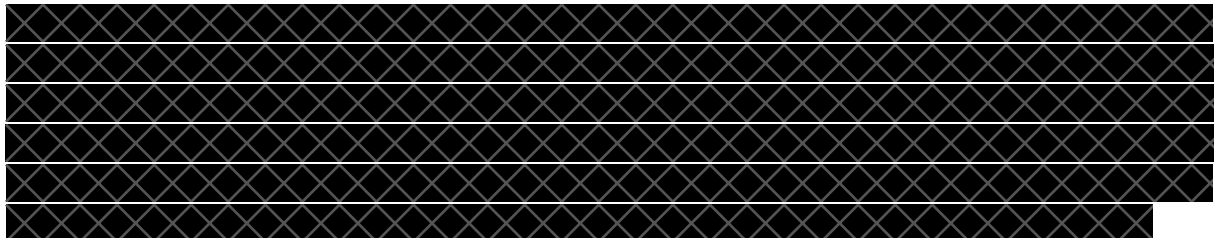
Tema 31. Lenguaje C: Características generales. Elementos del lenguaje. Estructura de un programa. Funciones de librería y usuario. Entorno de compilación. Herramientas para la elaboración y depuración de programas en lenguaje C.

Introducción: el lenguaje C	2
Características generales	2
Elementos del lenguaje	3
Tipos de datos	3
Variables	4
	5
Entorno de compilación	6
Estructura de un programa	6
Directivas de preprocesador	6
	7
Funciones de librería y usuario	7
Herramientas para la elaboración y depuración de programas	8
Conclusión	8
Bibliografía y webgrafía	9

Páginas	9
Palabras	2800
Esquemas/dibujos	0
Tablas	5
Nota: máx. 2700 palabras. Cada tabla/esquema unas 100-150 palabras	

1. Introducción: el lenguaje C

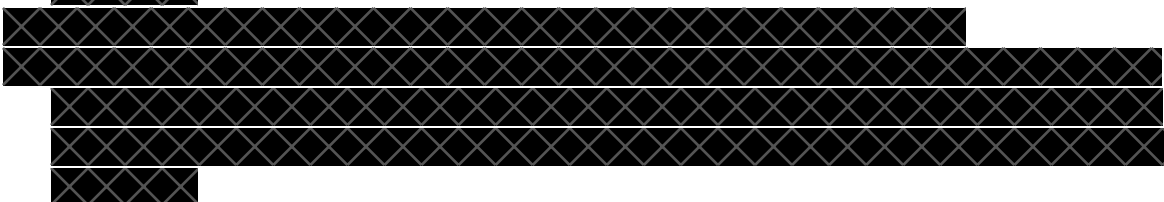
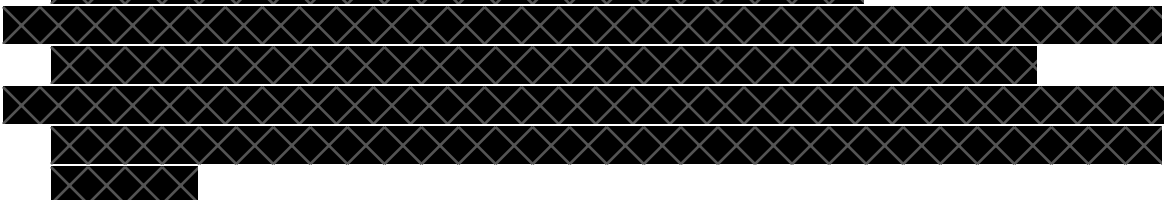
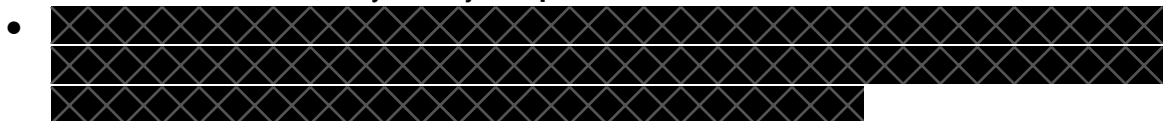
Uno de los lenguajes de programación más importantes a lo largo de la historia es, sin duda alguna, el lenguaje C. Su creación está íntimamente relacionada con el desarrollo del sistema operativo **Unix**, que fue implementado originalmente en el centro de investigación Bell Labs por Ken Thompson y Dennis Ritchie en lenguaje ensamblador. **Hoy en día sigue siendo uno de los lenguajes más populares y utilizados** en el mundo.



2. Características generales

Las características generales del lenguaje C son las siguientes:

- Es de **propósito general**; no está orientado a ninguna área en concreto ya que se pueden desarrollar desde núcleos de sistemas operativos, pasando por controladores de dispositivos hardware hasta aplicaciones software de usuario.
- Es un **lenguaje de alto nivel**, aunque algunos autores lo consideran de nivel medio ya que permite la manipulación de los elementos básicos del computador, como bits, bytes y direcciones de memoria. De hecho, una de sus características más “reconocibles” es el uso y manejo de **punteros**.



- Dispone de una amplia **biblioteca estándar**, con muchas funciones que le dotan de funcionalidad extra y, además, permite definir bibliotecas propias de usuario.

3. Elementos del lenguaje

A continuación se muestra un pequeño programa escrito en C, que calcula la longitud de una circunferencia de radio 50. Se explicarán los principales elementos del lenguaje a partir de este fragmento de código.

```
#include <stdio.h>
#define PI 3.1416

int MULT = 2;
float longitudCircunferencia( int radio );

int main() {
    int radio = 50;
    float longitud = longitudCircunferencia( radio );
    printf("La longitud de una circunferencia de radio %d es: %f", radio,
longitud );
}

float longitudCircunferencia( int radio ) {
    float longCir;
    longCir = MULT * PI * radio;
    return longCir;
}
```

3.1. Tipos de datos

En C existen cuatro tipos de datos elementales, que son:

char	Se utiliza para definir un carácter. Ej: 'a', 'b', 'c'...
int	Es el tipo de datos numérico entero (sin decimales). En el ejemplo, el radio (int radio = 50).
float	Define un número real (con decimales) de simple precisión (hasta 32 bits). En el ejemplo, la longitud de la circunferencia (float longCir)
double	Define un número real de doble precisión (hasta 64 bits)
void	Tipo vacío. Se suele utilizar para especificar que una función no devuelve ningún dato.

También existen los tipos de datos compuestos (estructuras), que se estudian en el tema 32, además de los vectores de datos (*arrays*). Por ejemplo, para construir una cadena de caracteres, en C se emplea un array de *chars*:

```
char hola1[] = "Hola";
char hola2[5] = { 'H', 'o', 'l', 'a', '\0' };
```

A los tipos de datos elementales numéricos se les pueden incluir diversos modificadores, como **signed y unsigned** (con signo o sin signo) o **long y short** (aumentar o disminuir el tamaño que ocupa el tipo de dato).

4. Entorno de compilación

El **proceso de compilado del lenguaje C** consta principalmente de tres fases:

- **Preprocesado.** Durante esta fase se modifica el código fuente desarrollado según una serie de directivas (`#define`, `#include...`) que se explicarán más adelante.
- **Compilación.** Se genera el programa objeto a partir del código fuente preprocesado. En un paso intermedio, desde el código fuente se genera el programa en ensamblador, que será después compilado a código máquina.
- **Enlazado.** Si existen códigos objeto de distintos módulos separados, como bibliotecas externas o de sistema, se unen para generar el programa ejecutable final.

En sistemas Unix podemos compilar un programa en C **haciendo uso del programa gcc** (*GNU Compiler Collection*). Este programa ya incluye todas las partes del proceso: llamada al preprocesador, compilado y enlazado de librerías para generar el ejecutable. Otro compilador muy popular es `clang`.

El uso básico de `gcc` es indicarle el programa fuente que recibirá y el nombre del fichero donde volcará el programa objeto compilado y listo para ejecutarse. Por ejemplo:

```
gcc hola.c -o hola
```

Otro ejemplo, esta vez sin llamar al enlazador.

```
gcc -c hola.c
```

El comando anterior preprocesará y compilará “hola.c”, y creará un programa objeto “hola.o” que no se podrá ejecutar ya que tendrá que ser enlazado primero, con el siguiente comando:

```
gcc hola.o -o hola
```

5. Estructura de un programa

La estructura general de un programa en C se compone de las 4 partes que se explican a continuación, y que deberían aparecer, además, en el orden indicado.

5.1. Directivas de preprocesador

Son instrucciones que se procesan por el preprocesador **antes de comenzar la fase de compilación**. El preprocesador las lee y **altera el contenido de los ficheros a compilar**.

Las principales son:

- **#define identificador valor.** Se conocen como macros, y permiten definir, como ya se ha comentado, constantes globales que se podrán utilizar en el programa. En el ejemplo encontramos `#define PI 3.1416`
- **#include <librería>.** Permite incluir el contenido de ficheros de cabecera (`.h`), para poder usar las funciones, constantes y definiciones incluidas en dichos ficheros. En el ejemplo encontramos `#include <stdio.h>`
- **#error mensaje.** Se para el proceso de compilación y se muestra el mensaje.