

**Informática**  
*Ingeniería en Electrónica y Automática Industrial*  
*Ingeniería en Tecnologías Industriales*  
**Introducción (rápida) al lenguaje C**

V1.3 © Autores

**Introducción al lenguaje C**

- Introducción
- Características del lenguaje C
- Variables
- Funciones en C
- Identificadores o etiquetas
- Sentencias
- Ejemplos

V1.3 © Autores

**Introducción al lenguaje C**

- Fue creado en los años 70 por Dennis Ritchie sobre una máquina PDP-11 bajo el sistema operativo Unix
- Se desarrolló bajo el sistema operativo Unix pero no está ligado a él ni a ningún otros sistema operativo (algunos sistemas operativos están escritos en C)
- La proliferación de implementaciones obligaron a la creación de un estándar: el **ANSI** (American National Standard Institute)

V1.3 © Autores

**Características del lenguaje C (I)**

- Tuvo mucho éxito desde el principio por ser
  - Compacto
  - Estructurado
  - Portátil
  - Flexible
  - De tipo medio
  - Muy difundido

V1.3 © Autores

**Características del lenguaje C (II)**

**COMPACTO**

- Sólo hay 32 palabras reservadas en el estándar ANSI:
 

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while
- Permite el uso de todas las operaciones algebraicas, lógicas y relacionales de las matemáticas convencionales
- Cualquier programa podría escribirse sólo con las palabras reservadas y los operadores definidos (muy laborioso)

V1.3 © Autores

**Características del lenguaje C (III)**

**ESTRUCTURADO**

- La componente estructural es la **función**
- Cada función permite partes de código con datos privados.
- Una función puede ser llamada desde otra función y le presta un servicio.

V1.3 © Autores

## Características del lenguaje C (IV)

### PORTÁTIL

- Los programas resultantes son independientes del hardware en el que se diseñan si se usan librerías y funciones del estándar
- Un mismo código es compilable en diferentes sistemas cambiando sólo el compilador
- El compilador de C es sencillo si se compara con otros compiladores, por lo que hay compiladores para todos los entornos

### FLEXIBLE

- Fue creado, influenciado y probado por programadores profesionales por lo que tiene pocas restricciones y poco control sobre las decisiones del programador
  - Inconveniente para los principiantes, ventaja para los avanzados
- Permite múltiples tipos de datos y gran facilidad para conversiones entre esos tipos

V1.3

© Autores

7

## Características del lenguaje C (V)

### DE TIPO MEDIO

- Combina elementos de lenguajes de alto nivel con otros de lenguajes de bajo nivel:
  - Potentes sentencias (alto nivel)
  - Operaciones a nivel de bits, registros del procesador, puertos y memoria (bajo nivel)

### MUY DIFUNDIDO

- Al ser sencillos, los compiladores de C son de los primeros que se crean cuando aparece un sistema nuevo
- Muy popular entre programadores profesionales y aficionados
- Muy utilizado para programar sistemas operativos, intérpretes, compiladores, ensambladores, drivers y controladores de red, etc.

V1.3

© Autores

8

## Variables en C

- Las **variables** en C son *porciones de memoria con un nombre*
- Se utilizan para almacenar valores que pueden ser modificados por el programa
- Deben ser *declaradas* antes de ser utilizadas
  - La declaración establece el tipo de dato que va a contener
- El C soporta todos los tipos básicos de variables (carácter, entero, decimal, etc.) y permite:
  - Modificar los tipos definidos
  - Crear tipos nuevos

V1.3

© Autores

9

## Funciones en C

- La **función** es la unidad primaria de programación en C: es donde se desarrolla la actividad del programa
- Cada función determina un bloque de código independiente y portable
- Forma genérica:

```
tipodevuelto nombrefunción(listaparámetros)
{
    declaraciones          /* Inicio de la función */
    ...                    /* Cuerpo de la función */
    proposiciones          /* Final de la función */
}
```

V1.3

© Autores

10

## Identificadores o etiquetas

- Son los nombres con los que se identifican
  - Las variables
  - Las constantes
  - Las funciones
- Características:
  - Deben empezar con carácter alfabético o el guión bajo «\_» y pueden contener caracteres alfanuméricos
  - No son válidas las palabras reservadas
  - El C distingue entre mayúsculas y minúsculas
- Recomendaciones
  - Las funciones creadas por el programador comienzan con mayúscula
  - Las etiquetas o identificadores de constantes definidas o constantes simbólicas se escriben en mayúsculas

V1.3

© Autores

11

## Sentencias en C

- Pueden ir en cualquier posición de la línea, no existe el concepto de *campo* (columnas en la línea)
- Finalizan con punto y coma «;»
- Sangrado
  - Sangrado de líneas atendiendo a criterios de subordinación
  - Facilita la lectura y escritura del programa
  - Es ignorada por el compilador
- Bloques de sentencias
  - Conjuntos agrupados entre llaves «{}» formando una unidad indivisible
- Comentarios
  - Textos aclaratorios imprescindibles (no abusar)
  - Precedidos de «//» hasta el final de la línea (no es estándar ANSI)
  - Entre los símbolos «/\* \*/» cualquier número de líneas
- Directivas del preprocesador
  - Sentencias que comienzan siempre con el símbolo almohadilla «#»
  - Determinan el modo de actuar del compilador sobre el archivo fuente
  - No forman parte del lenguaje C en un sentido estricto. Las incluyen todos los compiladores y facilitan mucho la programación

V1.3

© Autores

12

## Ejemplos (I)

- El programa más sencillo
- Obsérvese

- La directiva del preprocesador
- La función principal
- El comentario
- La llamada a una función
- La cadena de caracteres

```
#include <stdio.h>

main()      /*Función Principal */
{
    printf("Primer programa en C. \n");
}
```

V1.3

© Autores

13

## Ejemplos (II)

- Programa para convertir una temperatura en grados Fahrenheit a grados Celsius

- Obsérvese

- Las directivas del preprocesador
- La declaración de variables previa a su utilización
- Las sentencias de asignación y las operaciones aritméticas
- El tipo de los datos
- Los comentarios
- La función `printf()`
- La sentencia de salida `return`

V1.3

© Autores

14

## Ejemplos (III)

```
/* Conversión de una temperatura en grados Fahrenheit
a grados Celsius. */

#include <stdio.h>

int main()
{
    int fahren, celsius; /* Variables enteras */

    printf("Conversión de °F a °C:\n");

    fahren = 100;          /* Temperatura a convertir */
    celsius = 5*(fahren-32)/9; /* Fórmula de conversión */
    printf("%d °F = %d °C\n", fahren, celsius); /* Resultados */
    return 0;
}
```

V1.3

© Autores

15

## Ejemplos (IV)

- Programa para convertir cualquier temperatura en grados Celsius a grados Fahrenheit

- Obsérvese:

- El modo de lectura de datos del teclado

V1.3

© Autores

16

## Ejemplos (V)

```
/* Conversión de temperaturas Fahrenheit-Celsius empleando números
reales. */

#include <stdio.h>

int main()
{
    float fahren, celsius; /* Variables reales*/

    printf("Conversión de °F a °C:\n");
    printf("Introduce la temperatura Fahrenheit: ");

    scanf("%f", &fahren); /* Toma de datos reales */

    celsius = (5.0/9.0)*(fahren-32); /* Fórmula */
    printf("%f °F = %f °C\n", fahren, celsius); /* Resultados */

    return 0;
}
```

V1.3

© Autores

17

## Ejemplos (VI)

- Programa para convertir una temperatura en grados Celsius a grados Fahrenheit utilizando números reales

- Obsérvese

- La declaración y utilización de variables no enteras
- Las operaciones aritméticas con números reales
- Los cambios en las funciones de entrada y salida de datos

V1.3

© Autores

18

## Ejemplos (VII)

```

/* Conversión de temperaturas Fahrenheit-Celsius empleando números
reales. */

#include <stdio.h>

int main()
{
float fahren, celsius; /* Variables reales*/

printf("Conversión de °F a °C:\n");
printf("Introduce la temperatura Fahrenheit: ");
scanf("%f", &fahren); /* Toma de datos reales */

celsius = (5.0/9.0)*(fahren-32); /* Fórmula */
printf("%f °F = %f °C\n",fahren, celsius); /* Resultados */

return 0;
}

```

V1.3

© Autores

19

## Ejemplos (VIII)

- Programa que muestra una tabla de equivalencia entre temperaturas Celsius y Fahrenheit (con bucle de tipo mientras)
- Obsérvese:
  - La utilización de constantes simbólicas
  - La sentencia `while`, su sintaxis y significado
  - La comparación *menor o igual que* «<=»

V1.3

© Autores

20

## Ejemplos (IX)

```

/* Tabla de conversión de temperaturas Fahrenheit-Celsius. Versión con bucle
while y constantes simbólicas. */

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

#define LIMINFE 0 /* Limite inferior */
#define LIMSUP 100 /* Limite superior */
#define INCREM 10 /* Incrementos */

int main()
{
float fahren, celsius; /* Variables */
fahren = LIMINFE; /* Origen de la tabla */
system("clear"); /* Borra la pantalla (en Linux) */
printf("°F\t °C\n"); /* Cabecera de la tabla */
printf("=====n");
while (fahren <= LIMSUP)
{
celsius = (5.0/9.0)*(fahren-32.0);
printf("%3.0f\t%6.1f\n", fahren, celsius);
fahren = fahren + INCREM;
}
return 0;
}

```

V1.3

© Autores

21

## Ejemplos (X)

- Programa que muestra una tabla de equivalencia entre temperaturas Celsius y Fahrenheit (con bucle de tipo *para*)
- Obsérvese:
  - La utilización de distintos tipos de variables
  - El sangrado
  - La sentencia `for`, su sintaxis y significado
  - Los modificadores de formato en la función `printf()`

V1.3

© Autores

22

## Ejemplos (XI)

```

/* Tabla de conversión de temperaturas Fahrenheit-Celsius. */
/* Versión con bucle "for" */

#include <stdio.h>

int main()
{
float fahren, celsius; /* Variables */
int liminfe, limsup, increm;

liminfe = 0; /* Limite inferior */
limsup = 100; /* Limite superior */
increm = 10; /* Incrementos */

printf("°F\t °C\n"); /* Cabecera de la tabla */
printf("=====n");

for (fahren=liminfe ; fahren<=limsup ; fahren=fahren+increm)
{
celsius = (5.0/9.0)*(fahren-32.0);
printf("%3.0f\t%6.1f\n",fahren, celsius);
}
return 0;
}

```

V1.3

© Autores

23

## Ejemplos (XII)

- La palabra clave `if` introduce bloques de sentencias que pueden ejecutarse o no, en función del valor (booleano) que tome la expresión entre paréntesis:

```

if (expresion)
{
/* Bloque de sentencias 1 si la expresión es cierta */
}
else
{
/* Bloque de sentencias 2 si la expresión es falsa */
}

```
- Las llaves pueden permitir el agrupamiento de sentencias `if-else` de forma adaptada a las necesidades de cada caso

V1.3

© Autores

24

Ejemplos (XIII) Uso de `if`

```
#include <stdio.h>

int main()
{
    float a, b, max;           /* Variables */

    a = 0.5;                  /* Valor para a */
    b = 100.2;                /* Valor para b */

    if (a < b)
    {
        max = b;
    }
    else
    {
        max = a;
    }
    printf("a = %f, b = %f, max = %f\n", a, b, max);
    return 0;
}
```

V1.3

© Autores

25

## Ejemplos (XIV) Uso de una función

```
#include <stdio.h>

float maximo(float a, float b) /* Función que devuelve el máximo de dos números */
{
    float max;

    if (a < b)
    {
        max = b;
    }
    else
    {
        max = a;
    }
    return max;
}

int main()
{
    float a, b, max;          /* Declaración de variables */

    a = 0.5;                 /* Valor para a */
    b = 100.2;               /* Valor para b */

    max = maximo(a, b);
    printf("a = %f, b = %f, max = %f\n", a, b, max);
    return 0;
}
```

V1.3

© Autores

26