Tema 2: Fundamentos de C

Oscar Perpiñán Lamigueiro - David Álvarez

Hello World!

```c
#include <stdio.h>

int main()
{
  printf("Hello World!");

  return 0;
}
```

Hello World! (2)

```c
#include <stdio.h>

int main()
{
  printf("Hello World!\n");
  printf("¡Hola Mundo!\n");
  printf("Bonjour le Monde!\n");
  printf("Hallo Welt!\n");

  return 0;
}
```

Comentarios

```c
/* Este simple programa sirve para
   mostrar un mensaje en pantalla */
#include <stdio.h>

// Todo programa necesita una función main.
// Su contenido está delimitado entre llaves
int main()
{
  //La función printf muestra el mensaje en pantalla
  // Atención: el mensaje debe ir entre comillas
  printf("Hello World!\n");
  // La función main devuelve un entero con return.
  return 0;
} // Aquí acaba main y por tanto el programa
```

Palabras clave en C

Modificadores de tipo	Tipos primitivos	Tipos datos avanzados	Sentencias de control
`auto`	`char`	`enum`	`break`
`extern`	`double`	`struct`	`case`
`register`	`float`	`typedef`	`continue`
`static`	`int`	`union`	`do`
`const`	`long`		`else`
`volatile`	`short`		`if`
`signed`	`void`		`for`
`unsigned`	`sizeof`		`while`
			`return`
			~~`goto`~~
			`default`
			`switch`

bit

- Los ordenadores utilizan el sistema de numeración binario (dos dígitos, 0 y 1) para almacenar información.
- Un **dígito binario** (0 ó 1) se denomina *bit* (*binary digit*).
- Con **N bits** pueden representarse **2^N símbolos o 2^N números**
  - Ejemplo: con N = 8 bits se pueden representar los números positivos del 0 al 255 (2^8 - 1).

Representación de la información: binario y decimal

Ejemplo en decimal: 3452

10^3	10^2	10^1	10^0
1000	100	10	1
3	4	5	2

$$3452 = 3 \cdot 1000 + 4 \cdot 100 + 5 \cdot 10 + 2 \cdot 1$$

Ejemplo en binario: 10001111

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	**8**	**4**	**2**	**1**
1	0	0	0	**1**	**1**	**1**	**1**

$$128 + 8 + 4 + 2 + 1 = 143$$

Unidades de almacenamiento

- Byte: 8 bits ($2^8 = 256$)
- Kilobyte (KB): 1024 bytes ($2^{10} = 1024$)
- Megabyte (MB): 1024 KB (2^{20} bytes = 2^{10} KB)
- Gigabyte (GB): 1024 MB (2^{30} bytes = 2^{10} MB = 2^{20} KB)
- ...

Tipos de datos

`int` números enteros. Pueden ser **signed** o **unsigned**.

```
100 -41 0 12345
```

`float` y `double` números reales

```
3.0 101.2345 -0.0001 2.25e-3
```

`char` caracteres

```
's' '4' ';'
```

`_Bool` *booleanos*, 0 y 1

Nombres de constantes y variables

- Primer carácter: letra o carácter de subrayado (_) (**nunca un número**).
- Una o más letras (A-Z, a-z, *ñ excluida*), dígitos (0-9) o caracteres de subrayado.
- Tienen que ser distintos de las palabras clave.
- Las mayúsculas y las minúsculas son diferentes para el compilador.
- Es aconsejable que los nombres sean representativos

Constantes y Variables: declaración

- Son espacios de memoria que contienen datos. Se declaran para reservar la memoria en el ordenador.
- Declarar significa definir el tipo, asignarle un nombre y darle, si es necesario, un valor inicial.

tipo nombre=valor_inicial;

Constantes datos cuyo valor NO SE PUEDE MODIFICAR (**const**).

Variables datos cuyo valor se puede modificar mediante el operador *asignación* (=)

Definición de constantes y variables enteras

```c
#include <stdio.h>
int main()
{
  // declara una variable ENTERA con el identificador v1
  int v1;
  // declara una constante ENTERA con el identificador c1
  const int c1 = 4;
  // declara una variable ENTERA v2, y le asigna el valor 2
  int v2 = 2;
  // asigna el valor de la constante c1 a la variable v1 (cambia su valor previo)
  v1 = c1;
  // asigna el valor de la constante c1 a la variable v2
  v2 = c1;
  // error: c1 es una constante
  c1 = 3;

  return 0;
}
```

Definición de constantes y variables decimales

```c
#include <stdio.h>
int main()
{
  // declara una variable DECIMAL con el identificador v1
  float v1;
  // declara una constante DECIMAL con el identificador c1
  const float c1 = 4;
  // declara una variable DECIMAL v2, y le asigna el valor 2
  float v2 = 2;
  // asigna el valor de la constante c1 a la variable v1 (cambia su valor previo)
  v1 = c1;
  // asigna el valor de la constante c1 a la variable v2
  v2 = c1;
  // es esto válido?
  c1 = 3;

  return 0;
}
```

No sólo números - caracteres

- Cualquier información puede representarse con un conjunto de bits.
- ASCII (American Standard Code for Information Interchange): Estándar de 7 bits (128 caracteres), 95 caracteres imprimibles (del 32 al 126).

01001000	01101111	01101100	01100001
72	111	108	97
H	o	l	a

Tabla ASCII (no está entera)

32		44	,	56	8	68	D	80	P	92	\	104	h	116	t
33	!	45	-	57	9	69	E	81	Q	93	]	105	i	117	u
34	"	46	.	58	:	70	F	82	R	94	^	106	j	118	v
35	#	47	/	59	;	71	G	83	S	95	_	107	k	119	w
36	$	48	0	60	<	72	H	84	T	96	`	108	l	120	x
37	%	49	1	61	=	73	I	85	U	97	a	109	m	121	y
38	&	50	2	62	>	74	J	86	V	98	b	110	n	122	z
39	'	51	3	63	?	75	K	87	W	99	c	111	o	123	{
40	(	52	4	64	@	76	L	88	X	100	d	112	p	124	|
41	)	53	5	65	A	77	M	89	Y	101	e	113	q	125	}
42	*	54	6	66	B	78	N	90	Z	102	f	114	r	126	~
43	+	55	7	67	C	79	O	91	[	103	g	115	s		

Definición de constantes y variables de tipo caracter

```c
#include <stdio.h>

int main()
{
  //declara una variable de tipo CHAR con el identificador letra
  char letra = 'z';
  // declara una constante CHAR con el identificador c1
  const char c1 = '8';
  // asigna el valor de la constante c1 a la variable letra
  letra = c1;
  // Y char con un número? SI. Usando la Tabla ASCII. Qué caracter es?
  letra = 122;
  // es esto válido?
  c1 = 122;

  return 0;
}
```

Especificadores de formato y seciencias de escape

Formato	Significado
%c	Carácter simple
%d	Entero decimal
%i	Entero decimal, octal o hexadecimal
%e	Coma flotante con exponente
%f	Coma flotante sin exponente
%g	Usa %e o %f, el más corto
%lf	Usar con tipo double
%o	Entero octal, sin el cero inicial
%s	Cadena de caracteres
%u	Entero decimal sin signo
%x	Entero hexadecimal sin 0x.

Secuencia	Significado
\n	Salto de línea
\t	Tabulador horizontal
\"	Comilla doble
\\	Barra invertida

Imprimir frases en el terminal

```c
#include <stdio.h>

int main()
{
  // printf para imprimir una frase
  printf("El símbolo del hidrógeno es H, su número atómico es 1, y su masa atómica
       es 1.00794");

  // Y si lo queremos hacer con variables?
  // símbolo
  // número atómico
  // masa atómica

  return 0;
}
```

Tabla periódica

Imprimir datos en el terminal

```c
#include <stdio.h>

int main()
{
  // símbolo
  char sim = 'H';
  // número atómico
  int num_at = 1;
  // masa atómica
  float masa_at = 1.00794;

  // printf para imprimir una frase
  printf("El símbolo del hidrógeno es %c, su número atómico es %i, y su masa ató
      mica es %f", sim, num_at, masa_at);

  return 0;
}
```

Imprimir datos en el terminal

```c
#include <stdio.h>

int main()
{
  // símbolo
  char sim = 'H';
  // número atómico
  int num_at = 1;
  // masa atómica
  float masa_at = 1.00794;

  // printf para imprimir una frase
  printf("El símbolo del hidrógeno es %c, su número atómico es %i, y su masa ató
      mica es %f", sim, num_at, masa_at);

  return 0;
}
```

Imprimir datos en el terminal, números y caracteres

```
int main()
{

  char sim = 'H'; // símbolo
  int num_at = 1; // número atómico
  float masa_at = 1.00794; // masa atómica

  // printf para imprimir una frase
  printf("El símbolo del hidrógeno es %c, su número atómico es %i, y su masa ató
      mica es %f\n", sim, num_at, masa_at);

  // OJO!! Controlamos cuantos decimales aparecen
  printf("El símbolo del yodo es %c, su número atómico es %i, y su masa atómica es
       %.3f", sim+1, num_at+52, 126.90447);

  return 0;
}
```

Imprimir en el terminal tamaño de variables

sizeof

```c
#include <stdio.h>

int main()
{
  int a = 0;
  printf("Un int ocupa %i bytes", sizeof(a));

  // cual es el tamaño de las otras variables?

  // para que sirven las palabras de C long y short?

  return 0;
}
```

Imprimir en el terminal tamaño de variables, sizeof

```c
#include <stdio.h>

int main()
{
  // cual es el tamaño de las otras variables?
  // para que sirven las palabras de C long y short?
  // cambia el tamaño si uso unsigned y signed?

  printf("Un int ocupa %i bytes\n", sizeof(int) );
  printf("Un float ocupa %i bytes\n", sizeof (float) );
  printf("Un short int ocupa %i bytes\n", sizeof(short int) );

  return 0;
}
```

Ejercicio: combinar en un programa, todos los tipos (char, int, float, double), con los modificadores (unsigned, signed, long, short)

Tipos de datos especiales

```c
#include <stdio.h>
int main() {
  char a; // caracteres, guardados como enteros sin signo
  int b; // números enteros, 4 Bytes (B)
  float c; // números decimales, 4 B, hasta 6 cifras decimales

  double d; // números decimales, 8 B, hasta 14 cifras decimales

  // tipos compuestos, ejemplos
  unsigned int e; // solo numeros naturales y el 0
  short int f; // números enteros, 2 B

  long double g; // números decimales, 8 B, hasta 14 cifras decimales

  printf("int ocupa: %i B \t short int: %i B\n", sizeof(int), sizeof(short int));
  printf("double ocupa: %i B \t long double: %i B\n", sizeof(double), sizeof(long
      double));
  return 0;
}
```

Lectura de datos con scanf

scanf("formato", &variable);

```c
#include <stdio.h>
int main()
{
  // Primero debo reservar la memoria donde guardar la información
  int num;

  // Pedir el dato por escrito es OPCIONAL, pero deseable
  printf("Escribe un número \t");

  //Atención: SCANF, la variable debe ir precedida de &
  scanf("%i", &num);

  printf("Has escrito el número %i\n", num);

  return 0;
}
```

Lectura distintos tipos de datos con scanf

Ejercicio

```
#include <stdio.h>
int main()
{
	int num_ent;
	float num_dec;
	char caracter;
	double num_double;

	printf("Escribe un numero \t");
	scanf("%i", &num_ent);
	...
	// Rellenar el resto
	...
	return 0;
}
```

Errores comunes con `scanf`

- Escribir dentro de la cadena de control mensajes y secuencias de escape (p.ej. `\n`).
- Olvidar poner el operador **&** delante de los argumentos cuando son variables de los tipos básicos (**int**, **float**, **double**, `char`)
- Poner un especificador de formato no compatible con el tipo del argumento.

Lectura de caracteres, scanf y getchar

scanf(" %c", &variable);

variable = **getchar**();

- **scanf** vale para cualquier tipo de datos, mientras que **getchar** solo sirve para leer caracteres.
- ¿Qué pasa cuando queremos leer 2 caracteres de manera consecutiva?

Ejercicio: leer los caracteres de tu nombre para luego imprimirlos por pantalla. También leerlos como **int** e imprimirlos como **char**, y al revés.

Lectura de varios caracteres seguidos, fflush

```
#include <stdio.h>
int main()
{
  // Primero debo reservar la memoria donde guardar la información
  char ejemplo1, ejemplo2;

  // Pedir el dato por escrito es OPCIONAL, pero deseable
  printf("Escribe un caracter \t");
  ejemplo1 = getchar();

  printf("Escribe otro caracter \t");
  fflush(stdin);
  scanf("%c", &ejemplo2);

  printf("Has los caracteres %c y %c\n", ejemplo1, ejemplo2);

  return 0;
}
```

Aritméticos

```
x + y
x - y
x / y // OJO con el tipo de datos
x * y
x % y // módulo o resto de división de entera
```

Aritméticos con enteros

```c
#include <stdio.h>

int main()
{
  int x, y, sum;
  x = 10;
  y = 15;
  sum = x + y;

  printf("La suma de %i con %i es %i\n", x, y, sum);
  printf("La resta de %i con %i es %i\n", x, y, x-y);
  // los demás?
  // y con decimales?

  return 0;
}
```

Ejercicios

- Escriba un programa que lea del teclado una letra minúscula e imprima esa misma letra en mayúscula.

- Escriba un programa que pida un peso y altura de una persona y calcule su índice de masa corporal (imc). El imc de calcula dividiendo el peso en (kg) por la altura (m) al cuadrado.

Asignaciones especiales

- Recomiendo no usar mientras se aprende

```
x = y
x += y // x = x + y
x -= y // x = x - y
x *= y // x = x * y
x /= y // x = x / y
x %= y // x = x % y
```

```
// ERROR: en el lado izquierdo no puede ir una expresión
x + y = 1
```

Ejercicio operaciones de asignación

```c
#include <stdio.h>
int main() {
    int a, b = 3;

    a = 5;
    printf("a = %i\n", a);
    a *= 4; // a = a * 4
    printf("a = %i\n", a);
    a += b; // a = a + b
    printf("a = %i\n", a);
    a /= (b + 1); // a = a / (b+1)
    printf("a = %i\n", a);
    a = b = 1;
    printf("a = %i, b = %i\n", a, b);

    return 0;
}
```

Incrementales - solo usar en bucles

```
y = ++x // x = x + 1; y = x (preincremento)
y = x++ // y = x; x = x + 1 (postincremento)

y = --x // x = x - 1; y = x (predecremento)
y = x-- // y = x; x = x - 1 (postdecremento)
```

Ejercicio operaciones de incremento

```c
#include <stdio.h>

int main()
{
    int b = 2, r;
    //Preincremento
    r = ++b;
    printf("b = %i, r = %i\n", b, r);
    //Postincremento
    r = b++;
    printf("b = %i, r = %i\n", b, r);

    return 0;
}
```

Ejercicio operaciones de incremento 2

```c
#include <stdio.h>

int main()
{
    int a = 0;
    printf("a = %i\n", ++a);
    printf("a = %i\n", a++);
    printf("a = %i\n", a);
    printf("a = %i\n", --a);
    printf("a = %i\n", a--);
    printf("a = %i\n", a);

    return 0;
}
```

Ejercicio precedencia y asociatividad

```c
#include <stdio.h>

int main() { // Y si son enteros?
    float a = 4, b = 7, c = 3, g = 9, result;

    result = a + b * c;
    printf( "resultado = %f\n", result);

    result = (a + b) * c;
    printf( "resultado = %f\n", result);

    result = a * b / c * g;
    printf("resultado = %f\n", result);

    result = (a * b) / (c * g);
    printf("resultado = %f\n", result);

    return 0;
}
```

Relacionales

- El resultado es verdadero o falso.
- En programación esto significa 1 ó 0 respectivamente.

```
x == y
x != y
x > y
x >= y
x < y
x <= y
```

Ejercicio operaciones comparadores

```c
#include <stdio.h>
int main()
{
    int x = 10, y = 3;

    printf("x igual a y = %i\n", (x == y));
    printf("x distinto a y = %i\n", (x != y));
    printf("x mayor que y = %i\n", (x > y));
    printf("x menor o igual a y = %i\n", (x <= y));
    printf("x mayor o igual que y = %i\n", (x >= y));

    return 0;
}
```

Lógicos

- AND, OR, NOT

```
x && y //AND
x || y //OR
!x //NOT, operador unario
```

- Operador condicional ? (ternario)

```
// expresión boleana ? valor si cierto : valor si falso
x > y ? "cierto" : "falso"
x == y ? "true" : "false"
```

Ejercicio operaciones lógicas

```c
#include <stdio.h>
int main()
{
  int x, resto;

  printf("Escribe un número entero: ");

  scanf("%i", &x);

  // Calcula el resto de dividir por 2
  resto = x % 2;

  // Si el resto es 0, x es par.
  printf("Es un número %s\n", (resto == 0) ? "par" : "impar");

  return 0;
}
```

Conversión implícita

Asignaciones el valor de la derecha se convierte al tipo de la variable de la izquierda (posible aviso o error). NO RECOMENDABLE.

```c
#include <stdio.h>

int main()
{
  float f1 = 3.7, f2;
  int i1 = 2, i2;
  // Real a entero: pierde decimales
  i2 = f1;
  printf("Un real %f convertido a entero %i\n", f1, i2);
  // Entero a real: no cambia valor
  f2 = i1;
  printf("Un entero %i convertido a real %f\n", i1, f2);

  return 0;
}
```

Conversión explícita

Conversión explícita o forzada `(tipo) expresión`

```c
#include <stdio.h>

int main()
{
  int i1 = 5, i2=2;
  float f1;

  printf("La división entre enteros: %i/%i = %f\n", i1, i2, (float)(i1/i2));

  // Completar el último campo
  //printf("El resto de %i entre %i es: %i", i1, f1, ...);

  return 0;
}
```

Conversión en expresiones

Expresiones los valores de los operandos se convierten al tipo del operando que tenga la precisión más alta.

```
#include <stdio.h>

int main()
{
  double f1 = 100;
  int i1 = 150, i2 = 100;
  printf("Un entero, %i, dividido por un real, %f,", i1, f1);
  printf(" produce un real, %f\n", i1 / f1);
  printf("Un entero, %i, por un entero, %i: %i\n", i1, i2, i1 / i2);

  return 0;
}
```