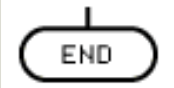
Conceptos básicos de programación en C

Flowchart Symbols	
	Start of a sequence (programme or sub -routine)
	Switching or controlling outputs, e.g. LED's, motors, sounders etc.
	Process commands, e.g. sub routines, delays etc.
	Checking inputs, e.g switches (digital) or sensors (analogue/digital)
	End of a sequence (programme or routine)

Sistemas embebidos

Embedded systems

Sistemas empotrados, embarcados.

Sistemas embebidos	Computadores convencionales
Realizan una tarea muy específica. Ejemplos: control de frenos ABS y climatizador en vehículos.	Equipo de propósito general. Ejemplos: edición de documentos, comunicación en red.
Forman parte de un sistema mayor.	Diseñados para trabajar de forma independiente
Los programas se almacenan en unidades de estado sólido (memorias ROM, FLASH), frecuentemente grabadas en fábrica	Sistema operativo de propósito general configurable en el que pueden instalarse multitud de aplicaciones en unidades de disco
Normalmente no interactúan directamente con operarios ni usuarios, y si lo hacen, se utilizan elementos de panel específicos. Ejemplo: horno microondas	Disponen de varios elementos de interfaz: pantalla, teclado, ratón, etc.

Identificadores

- Combinación de letras `A B … Z a b … z`, guión bajo _ y dígitos numéricos `0 … 9`.

- No puede comenzar por un dígito numérico.

- Se distinguen minúsculas de mayúsculas.

- Ejemplos:

```
temperatura
Temperatura
TEMPERATURA

TemperaturaHorno
temperatura_horno

TemperaturaHorno5
temperatura_horno_5
```

Números enteros con signo

Representación en complemento a 2

Tipo	Número N mínimo de bits	Rango -2^{N-1} ... $2^{N-1}-1$
char	8	-128 ... 127
short short int	16	-32768 ... 32767
int	16	-32768 ... 32767
long long int	32	-2147483647 ... 2147483648
long long long long int	64	-9223372036854775808 ... 9223372036854775807

Números enteros sin signo

Todos los bits son valor absoluto

Tipo	Número N mínimo de bits	Rango $0 \ldots 2^{N-1}$
`unsigned char`	8	0 ... 255
`unsigned short` `unsigned short int`	16	0 ... 65535
`unsigned` `unsigned int`	16	0 ... 65535
`unsigned long` `unsigned long int`	32	0 ... 4294967295
`unsigned long long` `unsigned long long int`	64	0 ... 18446744073709551615

Valores para números enteros

Ejemplo: $174446_{10} = 10\ 1010\ 1001\ 0110\ 1110_2 = 524556_8 = 2A96E_{16}$

- **Decimal**: dígitos de 0-9 — `174446`
- **Binario**: no se utiliza en lenguaje C estándar ..... `b00101010100101101110`
- **Octal**: dígitos 0-7 anteponiendo un 0 — `0524556`
- **Hexadecimal**: dígitos 0-9 y A-F anteponiendo 0x .... `0x2A96E`

Binario	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Binario	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binario	Hexadecimal
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Números reales

Formato IEEE 754 (simple y doble precisión)

Lenguaje C	Número bytes	Bits mantisa	Bits exponente	Rango (valor absoluto)
`float`	4	24	8	$1.17549435 \cdot 10^{-38}$ ... $3.40282347 \cdot 10^{38}$
`double`	8	53	11	$2.2250738585072014 \cdot 10^{-308}$ ... $1.7976931348623157 \cdot 10^{308}$
`long double`	12			$3.362103143112093506262677817321 8 \cdot 10^{-4932}$ ... $1.189731495357231765021263853031 \cdot 10^{4932}$

En decimal: $D * 2^E$

La mantisa `D` y el exponente `E` tienen bit de signo.

En 32 bits: `sEEEEEEE EDDDDDDD DDDDDDDD DDDDDDDD`

Los números reales son una representación aproximada

Ejemplos:

`a + (b + c)`	diferente de	`(a + b) + c`
`1.3 + 1.0`	diferente de	`2.3`
`a`	diferente de	`(a - b) + b`

Error frecuente en comparaciones:

```
1.   float x, y;
2.
3.   x = 10.0 / 2.0;
4.   y = 5.0;
5.   if (x == y) ...
```

En su lugar, considerad una precisión en los cálculos:

```
1.   float x, y, precision;
2.
3.   precision = 1E-5;
4.   x = 10.0 / 2.0;
5.   y = 5.0;
6.   if (abs(x - y) < precision) ...
```

Funciones

- Permiten la división y estructuración del programa en subrutinas.

Definición de una función:

```
TipoDevuelto IdentificadorFunción (Parámetros)
{
Variables locales
Instrucciones
}
```

Declaración y llamada a una función:

```
TipoDevuelto IdentificadorFunción (Parámetros);


... = IdentificadorFunción (Parámetros);
```

Ejemplo:

```
1.   int doble (int dato)
2.   {
3.   int resultado;
4.   resultado = 2 * dato;
5.   return resultado;
6.   }
```

```
1.   int x, y;
2.   int doble (int);
3.
4.   x = 3;
5.   y = doble (x);
```

Declaración de variables:

Consiste en asociar un tipo de datos determinado a una variable. Todas las variables a usar en un programa se deben declarar antes de utilizarlas. A continuación, se muestra el formato general de una declaración:

tipo lista_variables;

Ejemplos:

char letra, digito; //Declaración de 2 variables de tipo caracter

int dia = 15; //Declaración e inicialización de una variable de tipo entero

float tension; //Declaración de una variable en coma flotante

Tipos de variables:

- Locales: Definidas dentro de una función.

- Globales: Definidas fuera de todas las funciones y accesibles en todo el programa.

Las variables globales se declaran al principio, para que se puedan utilizar después en cualquier punto.

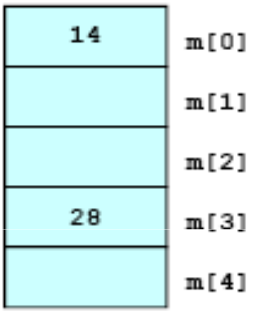
Matrices unidimensionales

- Declaración: *tipo identificador* `[capacidad];`

```
int m [5];
```

- Subíndices desde `0` hasta *capacidad-1*

```
m[0] = 14;
m[3] = m[0] * 2;
```



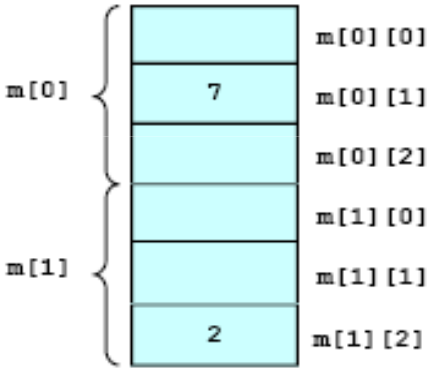
Matrices bidimensionales

- Declaración: *tipo identificador* `[`*filas*`][`*columnas*`];`

```
int m [2][3];
```

- Subíndices:

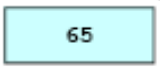
```
m[0][1] = 7;
m[1][2] = 2;
```



Caracteres

- Byte de tipo `char` en el que se representa el código del carácter según la **tabla ASCII**.

Equivalentes
```
char caracter;
caracter = 'A';
caracter = 65;
```



- Primera parte de la tabla ASCII: **códigos de control**

Dec	Hex	ASCII
0	00	NUL (NULl)
1	01	SOH (Start Of Heading)
2	02	STX (Start Of Text)
3	03	ETX (End of TeXt)
4	04	EOT (End Of Transmission)
5	05	ENQ (ENQuiry)
6	06	ACK (ACKnowledge)
7	07	BEL (BELl)
8	08	BS (BackSpace)
9	09	HT (Horizontal Tab)
10	0A	LF (Line Feed)
11	0B	VT (Vertical Tab)
12	0C	FF (Form Feed)
13	0D	CR (Carriage Return)
14	0E	SO (Shift Out)
15	0F	SI (Shift In)

Dec	Hex	ASCII
16	10	DLE (Data Link Escape)
17	11	DC1 (Device Control 1)
18	12	DC2 (Device Control 2)
19	13	DC3 (Device Control 3)
20	14	DC4 (Device Control 4)
21	15	NAK (Negative AcKnowledge)
22	16	SYN (SYNchronous idle)
23	17	ETB (End of Transmission Block)
24	18	CAN (CANcel)
25	19	EM (End of Medium)
26	1A	SUB (SUBstitute)
27	1B	ESC (ESCape)
28	1C	FS (File Separator)
29	1D	GS (Group Separator)
30	1E	RS (Record Separator)
31	1F	US (Unit Separator)

Segunda parte de la tabla ASCII: **caracteres de edición**

Dec	Hex	ASCII
32	20	SP
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3

Dec	Hex	ASCII
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G

Dec	Hex	ASCII
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[

Dec	Hex	ASCII
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	'
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

Dec	Hex	ASCII
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	6	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	DEL

SP = SPace
DEL = DELete

Operaciones aritméticas

- Con **números enteros**.

Ejemplo:

```
int a, b;
a = 5;
b = 3;
```

Operación	Ejemplo	Resultado
Suma	`a + b`	8
Resta	`a - b`	2
Multiplicación	`a * b`	15
División entera	`a / b`	1
Resto	`a % b`	2

- Con **números reales**.

Ejemplo:

```
float x, y;
x = 7.0;
y = 4.0;
```

Operación	Ejemplo	Resultado
Suma	`x + y`	11.0
Resta	`x - y`	3.0
Multiplicación	`x * y`	28.0
División	`x / y`	1.75

- **Combinación**: el formato menor se convierte al mayor y se opera. Ejemplo: `a + x`

- **Paréntesis** para modificar el orden de precedencia. Ejemplo: `(a + x) * (y + b)`

Incrementos y decrementos de números enteros

- Incremento:

`x ++;` equivalente a `x = x + 1;`

- Decremento:

`x --;` equivalente a `x = x - 1;`

- En otras instrucciones:

  - Incrementos y decrementos **previos**:

```
x = --y * 4;        equivalente a   { y = y - 1;
                                      x = y * 4;
x = ++y * 4;        equivalente a   { y = y + 1;
                                      x = y * 4;
```

  - Incrementos y decrementos **posteriores**:

```
x = y-- * 4;        equivalente a   { x = y * 4;
                                      y = y - 1;
x = y++ * 4;        equivalente a   { x = y * 4;
                                      y = y + 1;
```

Comparaciones: generan valores buleanos

Igual	==
Mayor	>
Menor	<
Mayor o igual	>=
Menor o igual	<=
Diferente	!=

```
1.    int a, b, c;
2.    a = 12;
3.    b = 3;
4.    c = a == b;
5.    c = a > b;
6.    c = a != b;
```

falso 0

verdadero 1

Utilizados frecuentemente para condiciones en bucles y sentencias condicionales:

```
1.    unsigned a, b, maximo;
2.    a = 12;
3.    b = 3;
4.    if (a < b) maximo = b;
5.        else maximo = a;
```

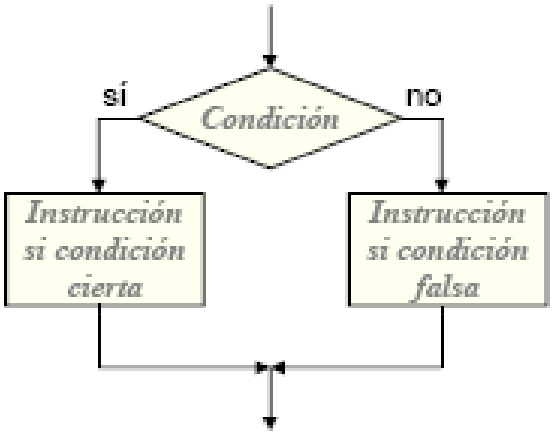
falso 0

ESTRUCTURAS DE CONTROL

- Condicionales:
  - if…else
  - siwtch …case
- Iterativas (bucles):
  - while
  - do while
  - for

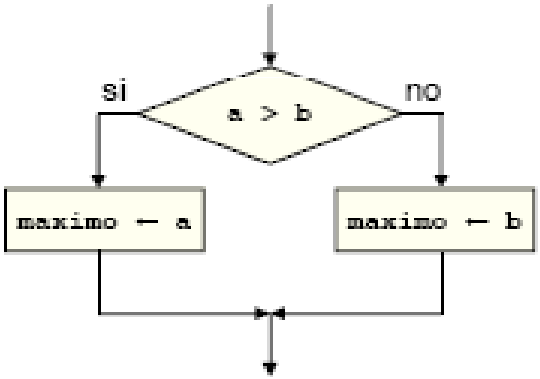
Instrucción `if`

```
1.    if (Condición)
2.         Instrucción si condición cierta
3.         else Instrucción si condición falsa
```

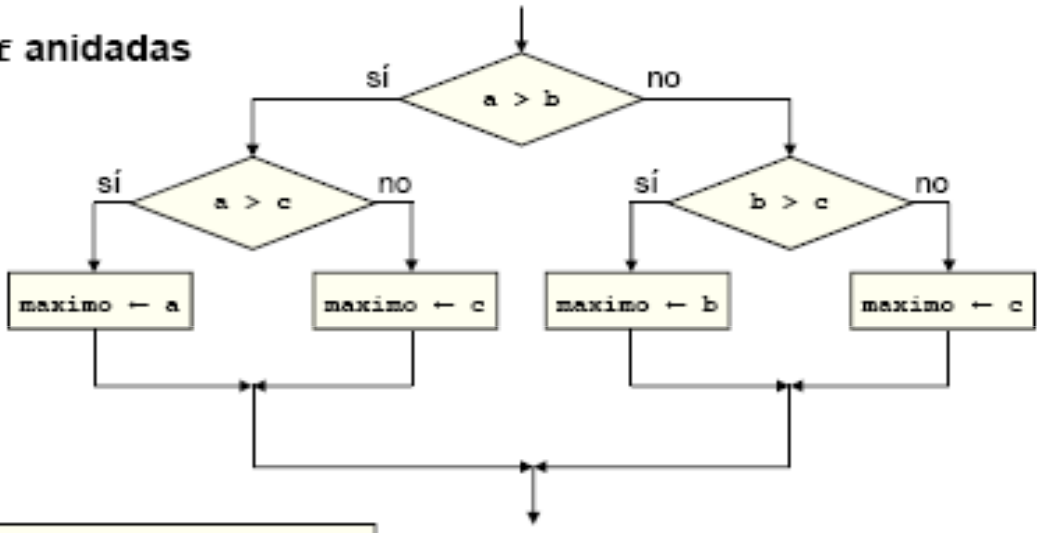


Ejemplo: hallar el máximo de dos valores numéricos

```
1.    int a, b, maximo;
2.
3.     Dar valores a las variables a, b
4.    …
5.    if (a > b) maximo = a;
6.      else maximo = b;
```



Varias instrucciones `if` anidadas



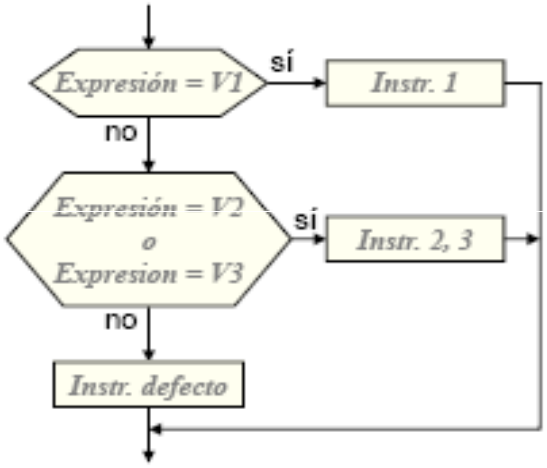

```
1.    int a, b, c, maximo;
2.
3.     Dar valores a las variables a, b, c
4.    …
5.    if (a > b)
6.      {
7.      if (a > c) maximo = a;
8.        else maximo = c;
9.      }
10.     else
11.     {
12.     if (b > c) maximo = b;
13.       else maximo = c;
14.     }
```

Ejemplo: hallar el máximo de tres valores numéricos

Instrucción `switch`

```
1.    switch (Expresión)
2.        {
3.        case V1:
4.             Instr. 1
5.             break;
6.        case V2, V3:
7.             Instr. 2, 3
8.             break;
9.        default:
10.            Instr. defecto
11.       }
```

La expresión y los posibles valores tienen que corresponder a un tipo entero.

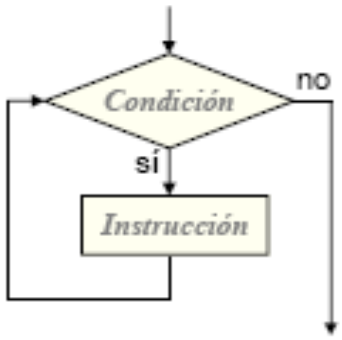


Ejemplo: cálculo del número de días en función del mes.

```
1.    unsigned mes, dias;
2.
3.    mes = Valor de 1 a 12
4.    switch (mes)
5.        {
6.        case 1, 3, 5, 7, 8, 10, 12:
7.             dias = 31;
8.             break;
9.        case 4, 6, 9, 11:
10.            dias = 30;
11.            break;
12.       default:
13.            dias = 28;
14.       }
```

Bucle `while`

```
while (Condición) Instrucción
```

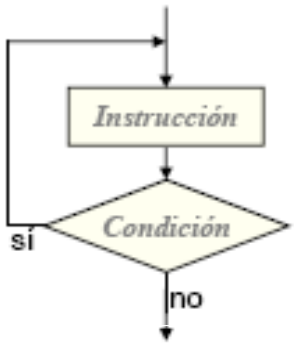


Ejemplo: cálculo del factorial 8!

```
1.    unsigned resultado, i;
2.
3.    resultado = 1;
4.    i = 2;
5.    while (i <= 8)
6.        {
7.        resultado = resultado * i;
8.        i ++;
9.        }
```

Bucle `do while`

```
do Instrucción while (Condición);
```

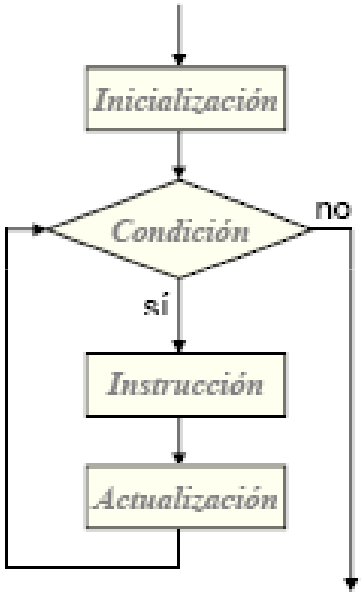


Ejemplo: cálculo del factorial 8!

```
1.    unsigned resultado, i;
2.
3.    resultado = 1;
4.    i = 1;
5.    do
6.        {
7.        i ++;
8.        resultado = resultado * i;
9.        } while (i < 8);
```

Bucle `for`

```
for (Inicialización; Condición; Actualización) Instrucción
```



Equivalente a:

```
Inicialización;
while (Condición)
    {
    Instrucción;
    Actualización;
    }
```

Ejemplo: cálculo del factorial 8!

```
1.    unsigned resultado, i;
2.
3.    resultado = 1;
4.    for (i = 2; i <= 8; i ++)
5.        resultado = resultado * i;
```