<u>Características</u> del lenguaje C:

- lenguaje de nivel medio
- Pocas palabras clave (minúsculas): 32= 27 (C original) + 5 ANSI (*enum*, *const*, *signed*, *void* y *volatile*).

auto	**break**	**case**	**char**	**const**	**continue**	**default**	**do**
double	**else**	**enum**	**extern**	**float**	**for**	**goto**	**if**
int	**long**	**register**	**return**	**short**	**signed**	**sizeof**	**static**
struct	**switch**	**typedef**	**union**	**unsigned**	**void**	**volatile**	**while**

- Repertorio reducido de sentencias: se puede describir en poco espacio y aprender rápidamente.
- Falta de restricciones en combinaciones de tipos de datos
- Algunos aspectos del lenguaje C son cuestionables

<u>Elementos</u> del lenguaje C:

Comentarios /* Comentario de C*/ //Comentario de C++

Identificadores

- Longitud < 31 (recomendado)
- Letra o subrayado seguido de letras, números o símbolos de subrayado.
- Se distingue minúsculas y mayúsculas
- Diferente a cualquier palabra clave o nombre de función de biblioteca.

Constantes literales

- **Números enteros**: Su formato es ***signo dígitos marcadores***.

  - ***signo***: "-" o "+" - por ejemplo -38 o +234 -.
  - ***dígitos*** : decimal -1893-, octal –0137- o en hexadecimal –0x58F3-.
  - Los ***marcadores***: 'l' (o L) entero long y "u" (o U) de tipo unsigned -por ejemplo 1234lu-.

- **Números reales** (con parte decimal):
  Su formato es ***signo dígitos e signo_exponente exponente marcador***.

  - El ***signo*** indica el signo de la mantisa.
  - ***Dígitos*** indica una secuencia de números que pueden llevar un punto separando la parte entera y la decimal.
  - ***e*** indica el comienzo del valor del exponente de base 10.
  - ***Exponente*** es una constante entera decimal.
  - ***marcador*** es una (f o F) y/o (l o L), donde las primeras indican una constante float y las segundas una doble precisión.

  Por ejemplo –13.13e-17f (es –13.13 por 10 a la –17)

- **Caracteres**:'*carácter': 'a', 'b', 'n' o* ' \carácter': '\n','\a'

- **Cadenas de caracteres**: "Secuencia caracteres"

Variables

tipo identificador = valor

Constantes: ***const tipo identificador = valor***

#define NOMBRECONST valor

Operadores

Aritméticos, relacionales, de asignación, lógico, de dirección y de movimiento.

Sentencias

De declaración y resto.

Terminación '**;**'.

Macros del preprocesador

#define NOMBRECONST valor

#include <nombrelibreria.h>

TIPOS DE DATOS

Tipos atómicos

Tipo	Tamaño de bits	Rango
char	8	0 a 255
int	16	–32768 a 32767
float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308
void	0	Sin valor

Modificadores

- char: **signed, unsigned**
- int: **signed, unsigned, long** y **short**
- double: **long**.

Tipo	Tamaño de bits	Rango
char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
int	16	–32768 a 32767
unsigned int	16	0 a 65535
signed int	16	–32768 a 32767
short int	16	–32768 a 32767
unsigned short int	16	0 a 65535
signed short int	16	–32768 a 32767
long int	32	-2147483648 a 2147483647
signed long int	32	-2147483648 a 2147483647
unsigned long int	32	0 a 4294967295
float	32	3.4E-38 a 3.4E+38
double	64	1.7E-308 a 1.7E+308
long double	128	1.2E-4932 a 1.2E+4932

Tipos derivados

- enum identificador {lista de nombres};

  Por ejemplo: enum estacion {invierno, primavera, verano, otonno};

  enum estacion variable_estación

- Tipos definidos: typedef: nuevo nombre para un tipo existente

  typedef tipo nombre;
  Por ejemplo, typedef float balance;

Operador *sizeof():* Devuelve un entero que representa el tamaño en bytes del tipo indicado como argumento

Conversión de tipos:

– Implícita:

Se da cuando se mezclan expresiones aritméticas de un tipo con las de otro.

- **Sentencia de asignación:** El valor del lado derecho se convierte al tipo de la variable del lado izquierdo. Puede haber pérdida de información

  unaVariableEntera = unaVariableReal * 0.5

- **Expresiones aritméticas**: si hay variables o valores de distintos tipos el compilador realiza determinadas conversiones antes de evaluar la expresión

  3 + 5.3 → 3.0 + 5.3 = 8.3

– **Explícita (typecast):** El resultado de una expresión o una variable se puede cambiar explícitamente a otro tipo utilizando la notación **(tipo) expresión** o **tipo (expresión)**

Por ejemplo :

```
float f=0.5;
f=f+1/2          ==     f=0.5+0     ==  0.5
f=f+float(1/2)   ==     f=0.5+0.5   ==   1
```

OPERADORES

Aritméticos

Operador	Acción
-	Resta, también menos monario
+	Suma
*	Multiplicación
/	División (entera o real)
a % b	Módulo, resto de la división

Para realizar otras operaciones se recurre a las funciones de la librería “math.h”: sqrt(n), pow(base,exp)

De Asignación

OPERADOR	SENTENCIA ABREVIADA	SENTENCIA NO ABREVIADA
=	=	=
++	m++	m=m+1
--	m--	m=m-1
+=	m+=n	m=m+n
-=	m-=n	m=m-n
*=	m*=n	m=m*n
/=	m/=n	m=m/n
%=	m%=n	m=m%n

- Operadores monarios: incremento (++) y decremento (--)

x=x+1; es equivalente a ++x; y a x++
x=x-1 es equivalente a --x y a x—

Pueden ir delante (preincremento) o detrás (postincremento)

- El resto son **binarios**

Lógicos

OPERADOR	SIGNIFICADO
!	Not (NO lógico)
&&	And (Y lógico)
||	Or (O lógico)

C no tiene definido el tipo booleano: expresión que de un valor numérico es 0 FALSO y distinto de 0 VERDADERO.

Relacionales

OPERADOR	SIGNIFICADO
==	Igual a
!=	No igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Error típico en C: confundir = con ==

Prioridad y asocitividad de operadores

Operador	Asociatividad
() []	De izquierda a derecha
- ++ -- ! ~ * & sizeof(tipo)	De derecha a izquierda
* / %	De izquierda a derecha
+ -	De izquierda a derecha
<< >>	De izquierda a derecha
< <= > >=	De izquierda a derecha
== ¡=	De izquierda a derecha
&	De izquierda a derecha
&&	De izquierda a derecha
||	De izquierda a derecha
?:	De derecha a izquierda
= *= /= %= += -= &= <<= >>=	De derecha a izquierda
,	De izquierda a derecha

ESTRUCTURA GENERAL DE UN PROGRAMA EN C

Palabras clave del lenguaje tienen que ir en minúsculas. Por convenio, las constantes en mayúsculas.

Inicio y fin de bloque: '**{**' '**}**' respectivamente. En particular bloque de main

Siempre debe estar la función principal main

int main ()

Al final de la función **main, return (0)**

Para que el compilador reconozca funciones invocadas, es necesario indicarle la **librería** (fichero con extensión **.h)** en la que residen.

#include <libreria.h>

```
/*Directivas del preprocesador*/
#include ..................
#define ..........
/*Declaración de variables globales y externas*/

/*Declaración de funciones*/

/*Módulo principal (main) del programa:*/
int main()
{/*Declaración de variables locales a main*/
        /*Sentencias del programa*/
}
```

SENTENCIAS DE ENTRADA-SALIDA POR CONSOLA

CON FORMATO: printf( ) y scanf( )

printf(): Sus argumentos son una *cadena con formato* y posiblemente varias expresiones o variables de un tipo y número acorde con lo expresado en la cadena con formato.

scanf(): Sus argumentos son una cadena 'con formato' y una lista de variables (precedidas por &)

a) Cadena 'con formato': cadena con 3 tipos de caracteres:

1. Especificadores de formato (%c, %d, %i, %e, %f,%o,%s,%p)

2. Caracteres de espacio en blanco (espacio en blanco, tabulación, Enter):

3. Caracteres que no son espacio en blanco (resto de caracteres

b) Variables de un tipo acorde en lo expresado en la cadena con formato. Estas variables deben ir **precedidas por el operador &**

Los especificadores de formato son los siguientes:

Código	Formato
%c	Un único carácter
%d	Decimal
%nºd	Indica la longitud total del número
%ld	Entero largo (long)
%hd	Entero corto (short)
%i	Decimal
%e	Notación científica
%f	Decimal en punto flotante
%nº.nºf	Indica la longitud de la parte entera y la de la parte decimal
%.nºf	Indica la longitud de la parte decimal
%g	Usar %e o %f, el más corto
%o	Octal
%s	Cadena de caracteres
%u	Decimal sin signo
%x	Hexadecimales
%%	Imprime un signo %
%p	Muestra un puntero

Con **printf** y **secuencias de escape**.

Código	Significado
\a	alarma
\b	Espacio atrás
\f	Salto de página
\n	Salto de línea
\r	Retorno de Carro
\t	Tabulación Horizontal
\"	Comillas dobles
\'	Comilla simple
\0	Nulo
\\	Barra invertida
\v	Tabulación vertical
\xdd	Carácter ASCII cuyo código hexadecimal es *dd*
OOO	Carácter ASCII cuyo código octal es *OOO*

E/S DE CARACTERES SIN FORMATO

getche( )

putchar( )

E/S DE CADENAS SIN FORMATO

gets( )

puts( )