TEMA 2.
LENGUAJE C. CONCEPTOS BÁSICOS Y PROGRAMACIÓN ELEMENTAL.

PARTE 1: VARIABLES, OPERADORES Y CONSTANTES.

1. Introducción al lenguaje C

- C es un lenguaje de programación estructurado de propósito general.
- Sus instrucciones constan de términos que se parecen a expresiones algebraicas además de ciertas palabras clave inglesas como *if, else, for, do* y *while*.
- Otras características son:
  - Portable.
  - Compatible.
  - Flexible, veloz y potente.
  - Fácil modificación.
  - Permite programación estructurada.
  - Compilado.

2. Programas básicos en lenguaje C

Primer programa en C:

```
#include <stdio.h>

void main(void)
{
    printf("Hola mundo\n");
}
```

- `#include <stdio.h>` → Acceso a biblioteca de funciones
- `void main(void)` → Función principal del programa
- `printf("Hola mundo\n");` → Instrucción de salida por pantalla.

Segundo programa en C:

```
#include <stdio.h>

void main(void)
{
    int numero;
    numero = 5;
    printf("Valor de variable = %d\n", numero);
}
```

- `int numero;` → Declaración de variables
- `numero = 5;` → Asignación de valor a variables
- `printf("Valor de variable = %d\n", numero);` → Instrucción de salida por pantalla.

2. Programas básicos en lenguaje C

```
#include <stdio.h>

void main(void)
{
        int numero;

        // Se inicializa la variable numero a 5
        numero = 5;
        printf("Hola mi numero es:%d \n", numero);
}
```

- ✓ **Declaraciones de variables** `int numero`
  - × A lo largo del programa se utilizará una variable de tipo entero y cuyo nombre es 'numero'.
  - × El sistema reserva espacio en memoria para ella.
- ✓ **Comentarios**
  - × Imprescindibles para el seguimiento de un programa. El sistema los ignora.
  - × Formatos:
    - `// ...`
    - `/* ... */`
- ✓ **Operador de asignación '='**
  - × La variable de la izquierda adquiere el valor de la expresión de la derecha.
  - × Es válido '`i = i+1`'

2. Programas básicos en lenguaje C

Tercer programa en C:

```
#include <stdio.h>

void main(void)
{
      int x;

      printf("Introduce el valor de x: ");
      scanf("%d", &x);
      printf("El valor introducido es: %d\n", x);
}
```

Instrucción de entrada de datos por teclado

3. Datos en lenguaje C

- ✓ **Un dato tiene tres atributos**
  - × **Nombre: designa en todo momento el dato, distinguiéndolo de los demás.**
  - × **Tipo: indica la clase o categoría a la que pertenece.**
  - × **Valor: es el contenido del dato en un instante dado.**
- ✓ **Una vez definida la variable, el único atributo susceptible de ser modificado es el valor**
- ✓ **Tipo de datos = valores posibles + operaciones**
  - × **Ejemplo: para el tipo de datos *entero***
    - **Valores posibles: un subconjunto de los números enteros**
    - **Operaciones: +, -, *, /, etc.**
- ✓ **Los datos pueden ser constantes o variables:**
  - × **Constantes: su valor no cambia a lo largo de la ejecución del programa**
  - × **Variables: su valor puede cambiar**

3. Datos en lenguaje C: VARIABLES

Tipos de variables (I)

TIPO	SINTAXIS	CARACTERÍSTICAS	TAMAÑO MEMORIA	RANGO
Entero	int	Sin parte fraccionaria	2 bytes	-32768 a +32767
Real	float	Con parte fraccionaria	1 palabra	$+-(10^{-37}$ a $10^{+38})$
Caracter	char	Letra o símbolo	1 byte	Código ASCII

- ✓ Los rangos de valores de la tabla sólo deben tomarse como indicativos. Dependen de los sistemas en que se usen.
- ✓ El tamaño concreto (en bytes) para un sistema dado, se puede obtener mediante la función *sizeof*.

3. Datos en lenguaje C: VARIABLES

Tabla de códigos ASCII

ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo
32	(espacio)	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	□

3. Datos en lenguaje C: VARIABLES

Tipos de variables (II)

TIPO	SINTAXIS	TAMAÑO MEMORIA	RANGO
Entero Corto	short	1 byte	-128 a 127
Entero largo	long	1 palabra	-2147483649 a -2147483648
Real de doble precisión	double	2 palabras	12 dígitos de precisión aprox.
Entero sin signo	unsigned int	2 bytes	0 a 65535
Entero corto sin signo	unsigned short	1 byte	0 a 255
Entero largo sin signo	unsigned long	1 palabra	0 a 4294967296

3. Datos en lenguaje C: OPERADORES

Operadores Aritméticos y de asignación (I)

OPERADOR	SINTAXIS	COMENTARIOS
Asignación	=	Variable izq. toma valor dch.
Adición	+	Suma dos operandos.
Sustracción	-	Resta dos operandos.
Multiplicación	*	Multiplica dos operandos
División	/	Divide dos operandos. El segundo operando debe ser distinto de cero.
Módulo	%	Devuelve el resto de la división entera. Los dos operandos deben ser enteros. El segundo operando debe ser distinto de cero.

- ✓ Cuando se realiza una operación entre dos variables de distinto tipo, el operando de menor precisión se transforma al de mayor precisión, y el resultado se da en el formato de mayor precisión.
- ✓ Es posible forzar el tipo de una variable poniendo entre paréntesis el nuevo tipo antes de la variable en la expresión.
- ✓ En una operación de asignación, si el tipo de datos asignados es distinto, el dato de la derecha se convierte al tipo del de la izquierda y se realiza la asignación.

3. Datos en lenguaje C: OPERADORES

Ejemplo

```
int i = 7;
float f = 5.5;
char c = 'w';
double d = 3.125
```

Expresión	Valor	Tipo
i+f	12.5	Float
i+c	126	Entero
i+c-'0'	78	Entero
(i + c) – (2 * f / 5)	123.8	Float
f + d	8.625	Double
((int) (i+f)) / 4	3	Int
((int) f) % 2	1	Int
i = 3.8	3	Int
i = f	5	Int
i = 'x'	120	Int

3. Datos en lenguaje C: OPERADORES

Operadores Aritméticos y de asignación (II)

OPERADOR	SINTAXIS	USO	SIGNIFICADO
Signo	-	y = -x	Cambia el signo de una variable.
Incremento	++	x++ ++x	x = x+1
Decremento	--	x-- --x	x = x-1
Suma-asignación	+=	x += y	x = x+y
Resta-asignación	-=	x -= y	x = x-y
Multiplicación-asignación	*=	x *= y	x = x*y
División-asignación	/=	x /= y	x = x/y
Módulo-asignación	%=	x %= y	x = x%y

Diferencia entre ++x y x++:

```
b = 0;
a = b++; // a=0, b=1
```

```
b = 0;
a = ++b; // a=1, b=1
```

3. Datos en lenguaje C: OPERADORES

Ejemplo

int i = 5, j = 7;
float f = 5.5, g = -3.25;

Expresión	Expresión equivalente	Valor
i += 5	i = i+5	10
f -= g	f = f-g	8.75
j *= (i-3)	j = j*(i-3)	14
f /= 3	f = f/3	1.8333333
i %= (j-2)	i = i % (j-2)	0

3. Datos en lenguaje C: OPERADORES

Operadores de comparación

OPERADOR	SINTAXIS	USO	SIGNIFICADO
Igualdad	==	x == y	¿Es x igual que y?
Desigualdad	!=	x != y	¿Es x distinto que y?
Mayor	>	x > y	¿Es x mayor que y?
Menor	<	x < y	¿Es x menor que y?
Mayor o igual	>=	x >= y	¿Es x mayor o igual que y?
Menor o igual	<=	x <= y	¿Es x menor o igual que y?
AND	&&	x && y	x AND y
OR	||	x || y	x OR y

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

3. Datos en lenguaje C: OPERADORES

Ejemplo

```
int i = 1, j = 2, k = 3;
char c = 'w';
float f = 5.5;
```

Expresión	Interpretación	Valor
i < j	Verdadero	1
(i + j) >= k	Verdadero	1
(j + k) > (i + 5)	Falso	0
k != 3	Falso	0
j == 2	Verdadero	1
c == 119	Verdadero	1
c != p	Verdadero	1
(i + f) <= 10	Verdadero	1
(i >= 1) && (c == 'w')	Verdadero	1
(i >= 6) || (c == 119)	Verdadero	1
(f < 11) && (i > 100)	Falso	0
!(f > 5)	Verdadero	1
!(i <=3)	Falso	0

3. Datos en lenguaje C: OPERADORES

Precedencia de operadores:

Categoría	Operadores	Asociatividad
Paréntesis	()	
Operadores unarios	- ++ -- !	D → I
Multiplicación, división y resto	* / %	I → D
Suma y resta	+ -	I → D
Operadores de comparación	> >= < <=	I → D
Operadores de igualdad	== !=	I → D
Y lógica	&&	I → D
O lógica	||	I → D
Operadores de asignación	= += -= *= /= %=	D → I

3. Datos en lenguaje C: OPERADORES

Ejemplo

```
int i = 7;
float f = 5.5;
char c = 'w';
```

Expresión	Interpretación	Valor
i + f <= 10	Falso	0
i >= 6 && c == 'w'	Verdadero	1
c != 'p' || i + f <= 10	Verdadero	1

3. Datos en lenguaje C: E/S DATOS

Salida de datos por pantalla

- ✓ Para mostrar datos por pantalla se utiliza la función *printf*, contenida en la librería *stdio.h*.
- ✓ Ejemplos de uso:

```
printf("Fundamentos de informatica\n");
printf("Fundamentos\nde\nInformatica");
```

- ✓ La salida por pantalla es:

```
Fundamentos de informatica
Fundamentos
de
Informatica
```

Carácter	Secuencia
\a	Sonido (alerta)
\n	Nueva línea
\t	Tabulador
\b	Retroceso
\r	Retorno de carro
\"	"
\'	'
\?	?
\\	\

3. Datos en lenguaje C: E/S DATOS

Salida de datos por pantalla

- ✓ Para mostrar el valor de una o varias variables, es necesario escribir *%* seguido de una letra que indica el tipo de variable que se va a mostrar.
- ✓ Ejemplo:

```
int x = 10;
float y = 0.025;
char c = 'b';

printf("Mis variables son x = %d, y = %f, y = %e,
    c = %c\n", x, y, y, c);
printf("x en octal es = %o, y en hexadecimal es
    = %x\n", x, x);
```

Tipo	Identificador
Entero	%d
Real	%f (con punto decimal) %e (formato científico) %g (min. espacio en pantalla)
Carácter	%c
String	%s
Long Int	%ld
Double	%lf
Unsigned	%u (sólo para enteros)
Octal	%o (sólo para enteros)
Hexadecimal	%x (sólo para enteros)

- ✓ La salida por pantalla es:

```
Mis variables son x =10, y = 0.025000, y = 2.500000e-002, c = b
x en octal es = 12 y en hexadecimal es = A
```


3. Datos en lenguaje C: E/S DATOS

Salida de datos por pantalla

- ✓ Es posible fijar una longitud de campo mínima y un número determinado de decimales para las variables que se muestran mediante *printf*.

Formato	Representación
%A.Bf	A = número total de caracteres B = número de decimales
%Ad **%Af**	A = número total de caracteres

- ✓ Si no se llega a la longitud mínima, se rellena con espacios en blanco a la izquierda de la variable. Si se excede esta longitud, se muestra el dato completo (no se hace caso a la longitud indicada en el formato).
- ✓ Ejemplo:

```
int x=123456;
float y = 34.5;
printf("%3d %5d %8d\n\n", x, x, x);
printf("%f %.3f %.1f\n\n", y, y, y);
printf("%e %.5e %.3e\n\n", y, y, y);
printf("%3g %10g %13g\n\n", y, y, y);
printf("%7f %7.3f %7.1f\n\n", y, y, y);
printf("%12e %12.5e %12.3e\n\n", y, y, y);
```

- ✓ Salida por pantalla:

```
123456 123456   123456

34.500000 34.500 34.5

3.450000e+001 3.45000e+001 3.450e+001

34.5      34.5        34.5

34.500000  34.500    34.5

3.450000e+001 3.45000e+001   3.450e+001
```

3. Datos en lenguaje C: E/S DATOS

Introducción de datos por teclado

- ✓ Para leer datos desde teclado, se utiliza la función *scanf*, contenida en la librería *stdio.h*.
- ✓ Se debe indicar entre comillas el formato de la variable que se va a introducir (precedido de %) y a continuación, el nombre de la variable donde se va a guardar el dato, precedido de &.
- ✓ Ejemplo

```
int x;
printf("Introduzca el valor de x:  ");
scanf("%d", &x);
printf("La variable introducida es x = %d\n", x);
```

- ✓ La salida por pantalla es:

```
Introduzca el valor de x:  5
La variable introducida es x = 5
```


3. Datos en lenguaje C: CONSTANTES

- ✓ Las constantes sirven para representar valores que se utilizarán a lo largo del programa y que no variarán en el mismo.
- ✓ Las constantes se definen antes de main, mediante el siguiente formato:
  - × #define NombreSimbólico Valor
  - × NombreSimbólico no puede contener espacios.
  - × Ejemplo: #define PI 3.1416
- ✓ Una vez definida una constante, el *preprocesador* sustituye los nombres simbólicos por su valor en todo el código fuente, antes de realizar la compilación.
- ✓ "NombreSimbólico" no se reemplaza por "Valor" cuando aparece en un bloque entre comillas.

```
#define CONSTANTE 15
...
int x = 10;
printf("CONSTANTE = %d\n", x);
```

Salida por pantalla:

```
CONSTANTE = 10
```

```
#define CONSTANTE 15
...
int x = 10;
printf("CONSTANTE = %d\n", CONSTANTE);
```

Salida por pantalla:

```
CONSTANTE = 15
```

3. Datos en lenguaje C: CONSTANTES

3 versiones válidas de un programa:

```
// circunferencia version A

main()
{
   float perim, area, radio;
   printf("Introduzca radio: ");
   scanf("%f", &radio");

   perim = 2*3.1416*radio;
   area = 3.1416*radio*radio;
   printf("Perimetro = %f\n" perim);
   printf("Area = %f\n", area);
}
```

```
// circunferencia version B

main()
{
   float perimetro, area, radio, pi=3.1416;
   printf("Introduzca radio: ");
   scanf("%f", &radio");

   perim = 2*pi*radio;
   area = pi*radio*radio;
   printf("Perimetro = %f\n" perim);
   printf("Area = %f\n", area);
}
```

```
// circunferencia version C

#define PI 3.1416

main()
{
   float perimetro, area, radio;
   printf("Introduzca radio: ");
   scanf("%f", &radio");

   perim = 2*PI*radio;
   area = PI*radio*radio;
   printf("Perimetro = %f\n" perim);
   printf("Area = %f\n", area);
}
```

- ✓ Inconvenientes versión A
  - × Falta de legibilidad del programa.
  - × Dificultad para modificaciones futuras.
- ✓ Inconvenientes versión B
  - × Menor eficiencia en ejecución.