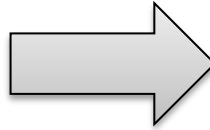
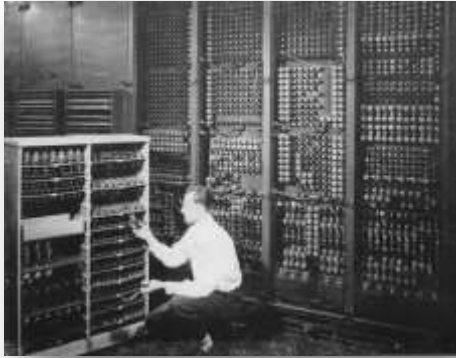


Programación en Lenguaje C;

Erick Becker's Code...

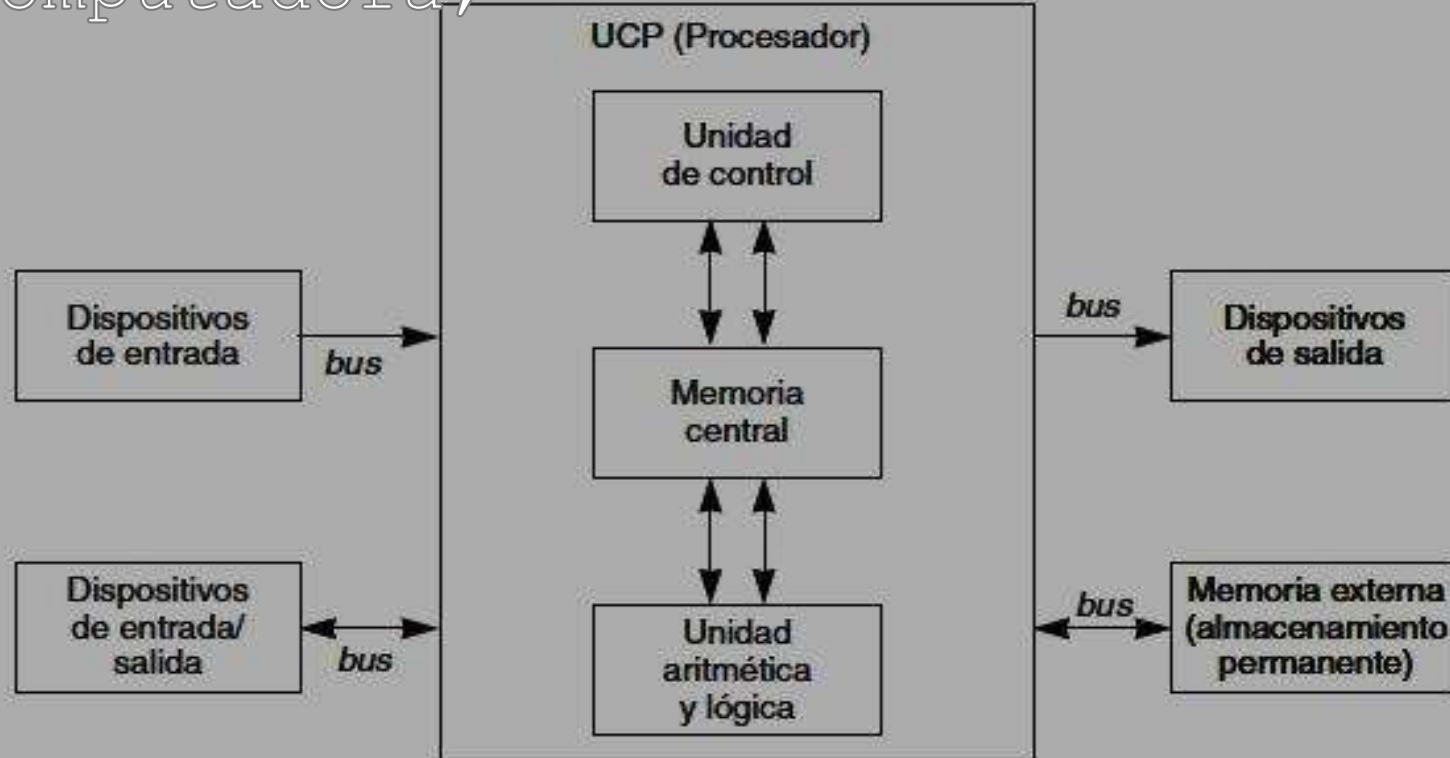


¿Qué es una computadora?;



Es una maquina electrónica, analógica o digital, dotada de una memoria de gran capacidad y de métodos de tratamiento de la información capaz de resolver problemas matemáticos y lógicos mediante la utilización automática de **programas informáticos** a gran velocidad.

Organización física de una computadora;



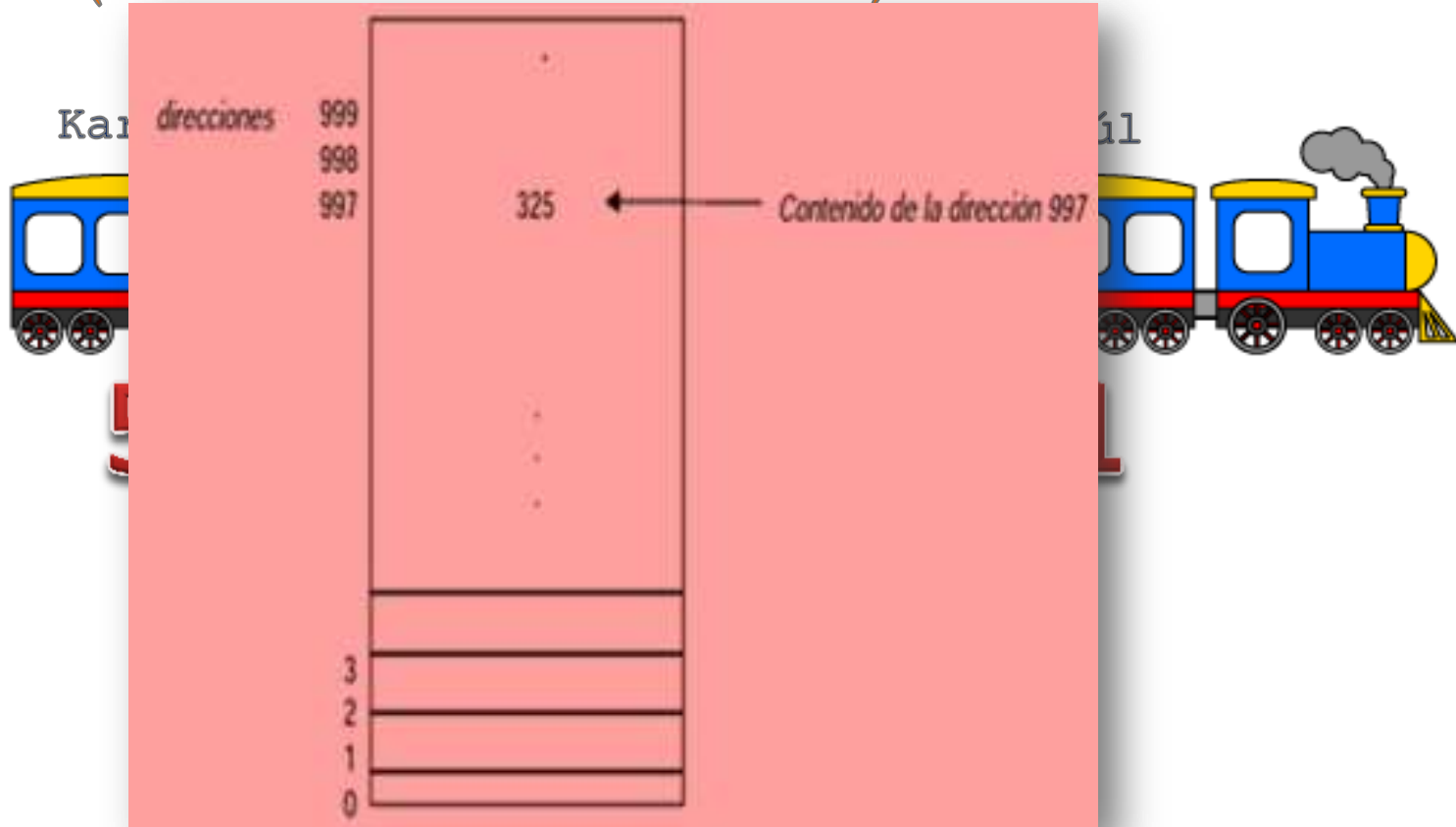
Memoria principal o
central;



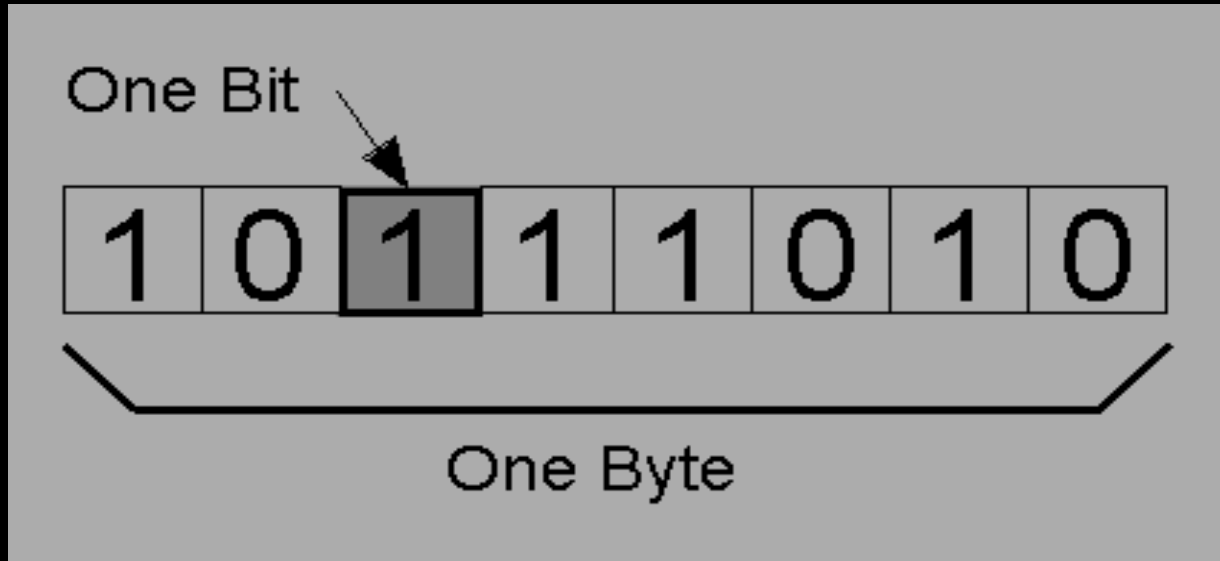
KURTIS BICKHAUS, 2009

Memoria principal

RAM (Random Access Memory);



Relación Bit - Byte;



¿Para qué sirve
un byte?;

Respuesta;

Los bytes sirven para representar los caracteres (letras, números y signos de puntuación adicionales) en un código estándar internacional denominado **ASCII (American Standard Code for Information Interchange)**, utilizado por todos los computadores del mundo.



Código ASCII (American Standard Code for Information Interchange);

Caracteres de control ASCII				Caracteres ASCII imprimibles												ASCII extendido													
DEC	HEX	Símbolo ASCII		DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo					
00	00h	NULL	(carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó					
01	01h	SOH	(inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	í	193	C1h	ł	225	E1h	ô					
02	02h	STX	(inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	194	C2h	ŧ	226	E2h	ö					
03	03h	ETX	(fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	â	163	A3h	û	195	C3h	Ŧ	227	E3h	õ					
04	04h	EOT	(fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ñ	196	C4h	—	228	E4h	ö					
05	05h	ENQ	(enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	à	165	A5h	Ñ	197	C5h	+	229	E5h	Õ					
06	06h	ACK	(acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	á	166	A6h	°	198	C6h	ä	230	E6h	µ					
07	07h	BEL	(timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	199	C7h	Å	231	E7h	þ					
08	08h	BS	(retroceso)	40	28h	(72	48h	H	104	68h	h	136	88h	ê	168	A8h	¿	200	C8h	Ľ	232	E8h	þ					
09	09h	HT	(tab horizontal)	41	29h)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	®	201	C9h	Œ	233	E9h	Û					
10	0Ah	LF	(salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	™	202	CAh	ƒ	234	EAh	Ü					
11	0Bh	VT	(tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	½	203	CBh	ƒ	235	EBh	Ù					
12	0Ch	FF	(form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	î	172	ACH	¼	204	CCh	ƒ	236	ECh	Ý					
13	0Dh	CR	(retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ï	173	ADh	»	205	CDh	ƒ	237	EDh	Ÿ					
14	0Eh	SO	(shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ā	174	AEh	«	206	CEh	ƒ	238	EEh	ˆ					
15	0Fh	SI	(shift In)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Ā	175	AFh	»	207	CFh	ƒ	239	EFh	˙					
16	10h	DLE	(data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	Ē	176	B0h	ˆ	208	D0h	ð	240	F0h	±					
17	11h	DC1	(device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	ˆ	209	D1h	ð	241	F1h	±					
18	12h	DC2	(device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	ˆ	210	D2h	ð	242	F2h	±					
19	13h	DC3	(device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	ˆ	211	D3h	ð	243	F3h	¼					
20	14h	DC4	(device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ò	180	B4h	ˆ	212	D4h	ð	244	F4h	¶					
21	15h	NAK	(negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ò	181	B5h	ˆ	213	D5h	ð	245	F5h	§					
22	16h	SYN	(synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	û	182	B6h	ˆ	214	D6h	ð	246	F6h	÷					
23	17h	ETB	(end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ù	183	B7h	ˆ	215	D7h	ð	247	F7h	°					
24	18h	CAN	(cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	©	216	D8h	ð	248	F8h	ˆ					
25	19h	EM	(end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	ÿ	185	B9h	ˆ	217	D9h	ð	249	F9h	ˆ					
26	1Ah	SUB	(substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	ÿ	186	BAh	ˆ	218	DAh	ð	250	FAh	ˆ					
27	1Bh	ESC	(escape)	59	3Bh	;	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	BBh	ˆ	219	DBh	ð	251	FBh	ˆ					
28	1Ch	FS	(file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	ˆ	220	DCh	ð	252	FCh	ˆ					
29	1Dh	GS	(group separator)	61	3Dh	=	93	5Dh]	125	7Dh	}	157	9Dh	Ø	189	BDh	ˆ	221	DDh	ð	253	FDh	ˆ					
30	1Eh	RS	(record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	x	190	BEh	ˆ	222	DEh	ð	254	FEh	ˆ					
31	1Fh	US	(unit separator)	63	3Fh	?	95	5Fh	_				159	9Fh	f	191	BFh	ˆ	223	DFh	ð	255	FFh	ˆ					
127	20h	DEL	(delete)							elCodigoASCII.com.ar																			

Hora de Matemática Discreta;



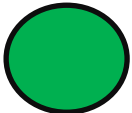
¿Qué es Software?;

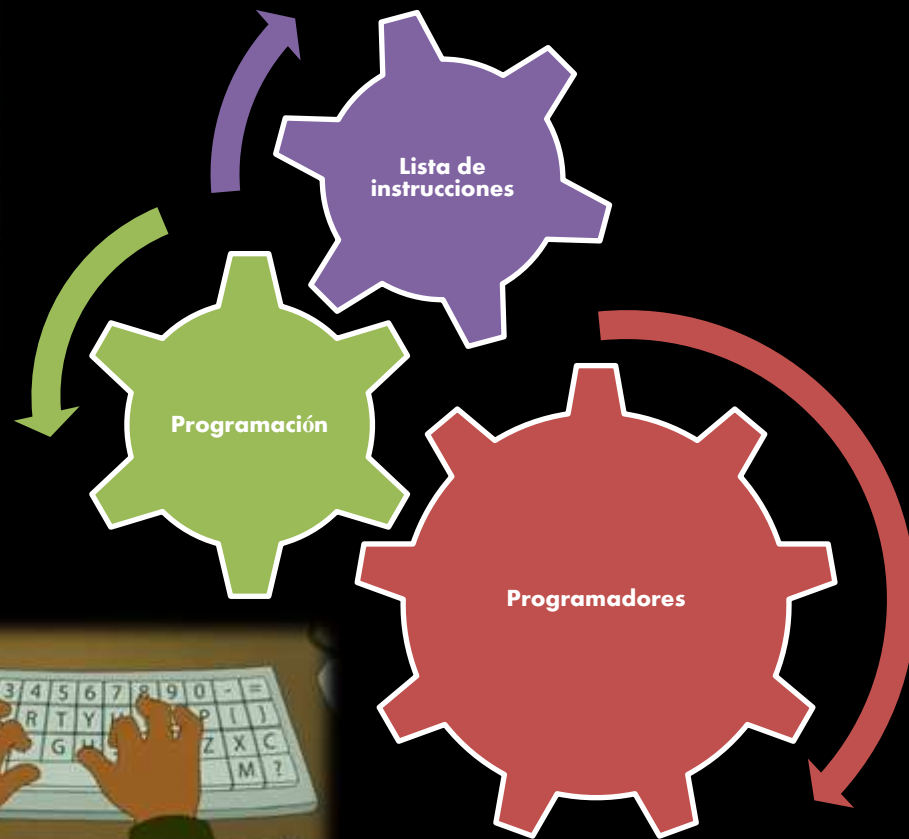


El **software** de una computadora es un conjunto de instrucciones de programa detalladas que controlan y coordinan los componentes **hardware** de una computadora y controlan las operaciones de un sistema informático.



En pocas palabras el **software** le dice al **hardware** “que hacer”...





Lenguajes de programación;

Se usan para escribir programas.



Pero...

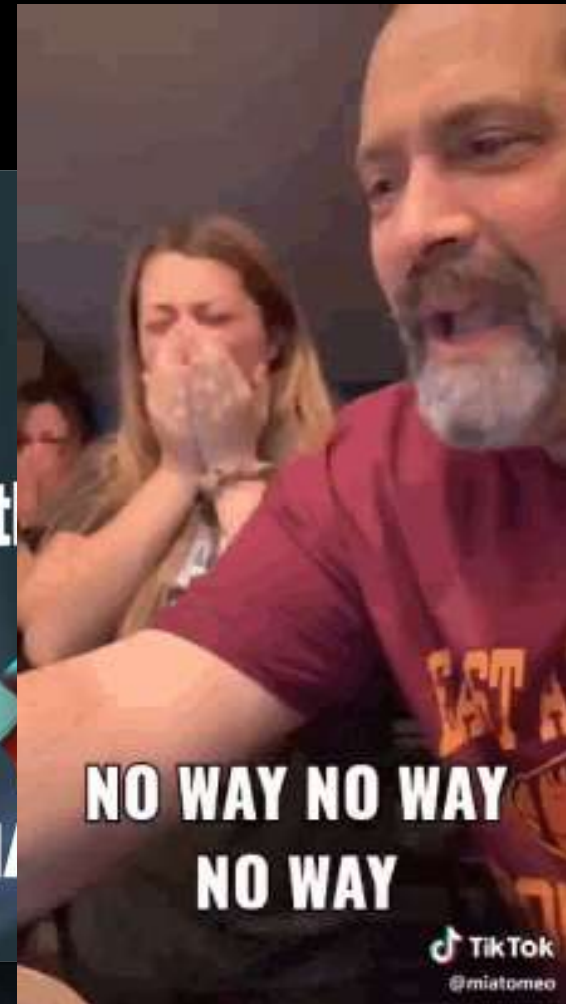
El lenguaje nativo de una computadora se llama: **lenguaje maquina.**

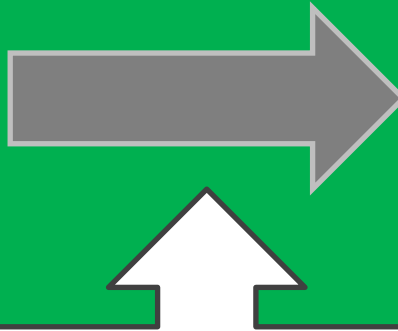
Desgraciadamente;

11001010	00010111	11110101	00101011
00010111	11110101	00101011	00101011
11001010	00010111	11110101	00101011
00010111			00101011
11001010			00101011
11001010			00101011
11001010			00101011
11001010			00101011
11001010			00101011
00010111	11110101	00101011	00101011
11001010	11110101	00101011	00101011



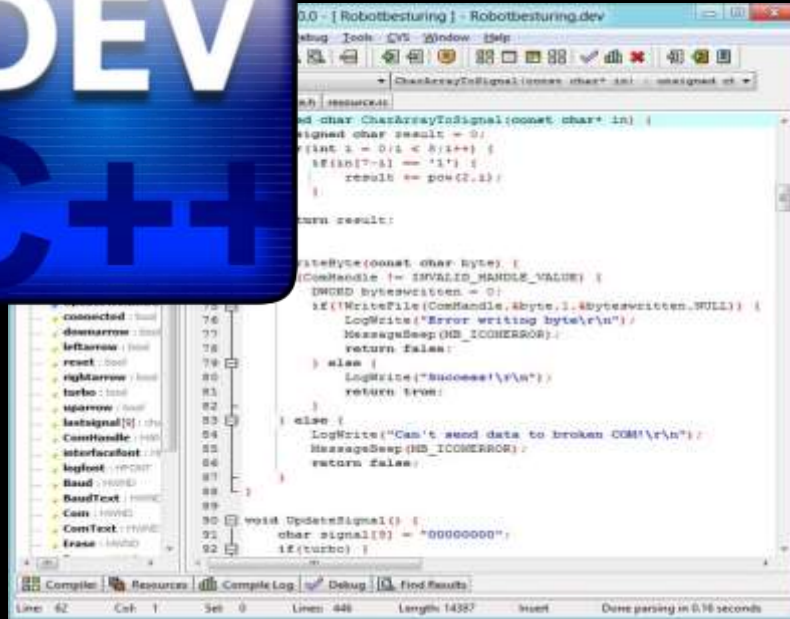
Afortunadamente;





```
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
11001010 11001010 11110101 00101011
11001010 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
```

Los lenguajes de alto nivel necesitan ser traducidos a lenguaje maquina para ello se necesitan traductores denominados técnicamente, **compiladores**.



IDE
(Integrated
Development
Environment)

[Dev-C++ download | SourceForge.net](#)

Estructura general de un programa en C

```
#include <stdio.h>  ← Archivo de cabecera  
  
int main() ← Cabecera de función  
{  
    ↑  
    ← Nombre de la función  
  
... ← Sentencias  
  
}
```

THE
C
PROGRAMMING
LANGUAGE


COBOL



Procedimental
o imperativo.

Funcional

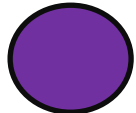
**Paradigmas de
programación.**

Declarativo

Orientado a
Objetos



Microsoft
VB.net



Tipos de Lenguajes

```
graph TD; A[Tipos de Lenguajes] --> B[Lenguaje Máquina]; A --> C[Lenguajes de bajo nivel]; A --> D[Lenguajes de alto nivel]; B --> E["0010 0000<br/>0010 1000"]; C --> F[Ensamblador]; D --> G["Fortran, COBOL, Pascal, C,<br/>C++, Java, C#, HTML,<br/>Python..."];
```

Lenguaje Máquina

0010 0000
0010 1000

**Lenguajes de bajo
nivel**

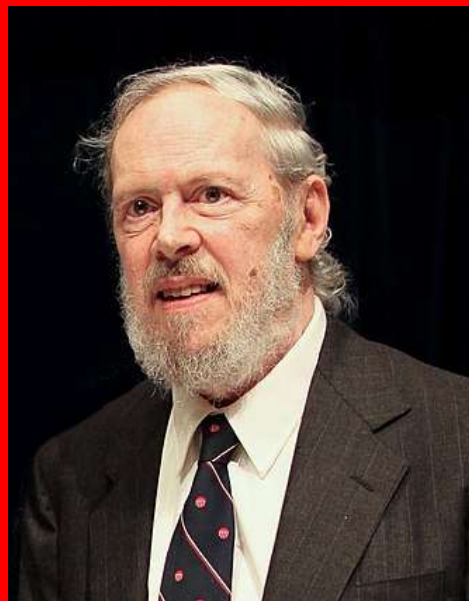
Ensamblador

Lenguajes de alto nivel

Fortran, COBOL, Pascal, C,
C++, Java, C#, HTML,
Python...



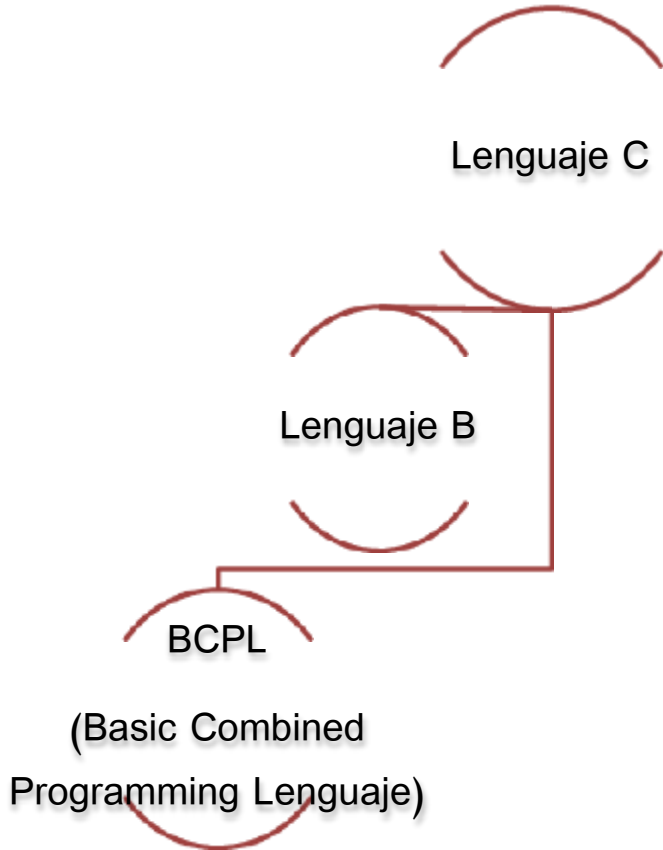
Historia del Lenguaje C



Dennis Ritchie



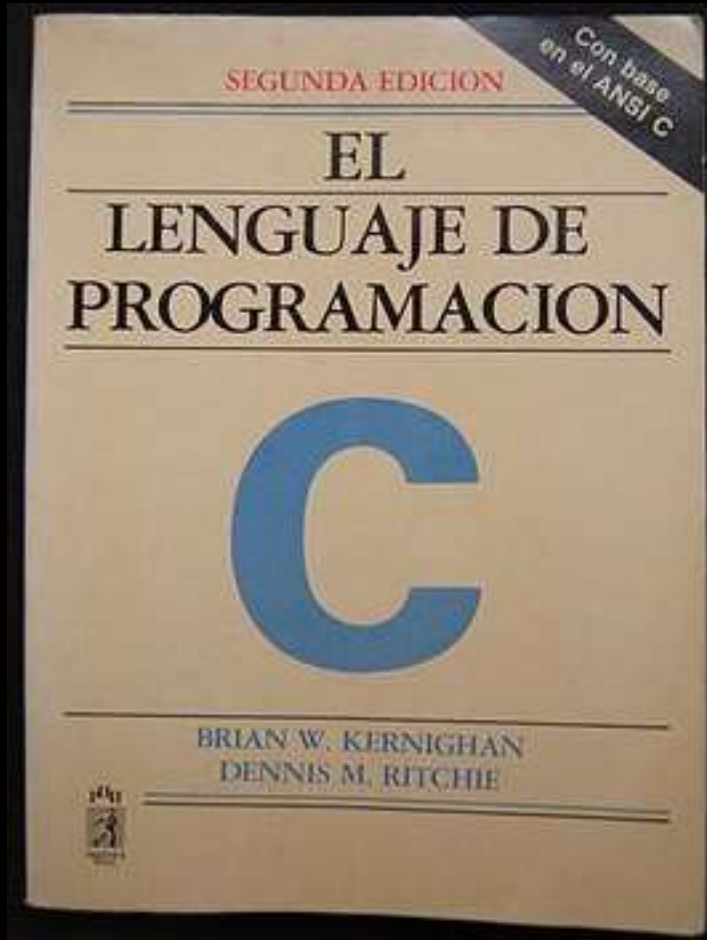
Ken Thompson



C, es conocido como el **lenguaje de programación de sistemas, por excelencia.**



Bell Labs



1983, ANSI (American National Standard Institute)





1. Sistema binario

Convertir los siguientes números de sistema decimal a sistema binario.

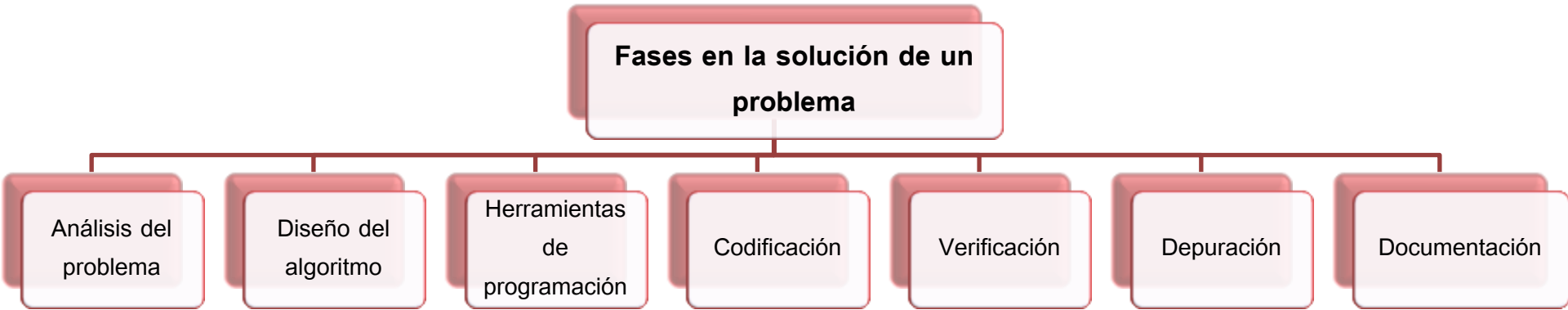
- | | |
|-------|----------|
| a) 5 | f) 100 |
| b) 14 | g) 256 |
| c) 23 | h) 1204 |
| d) 31 | i) 3897 |
| e) 50 | j) 11951 |

Convertir los siguientes números de sistema binario a sistema decimal.

- | | |
|--------------|----------------|
| a) 101101 | f) 1011011110 |
| b) 1000001 | g) 101101111 |
| c) 10101011 | h) 10100010101 |
| d) 11110001 | i) 10101111111 |
| e) 100111100 | j) 10111111101 |

Averiguar qué peso es la calculadora con los números entre el 32,760 y el 32,770.

Metodología de la programación;



Algoritmo;



Un **algoritmo** es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

Alkhô-warîzmi

Calcular el valor de la suma

$$1+2+3+4+5+\dots+100$$

Hora de break
matemático;

Tipos de instrucciones

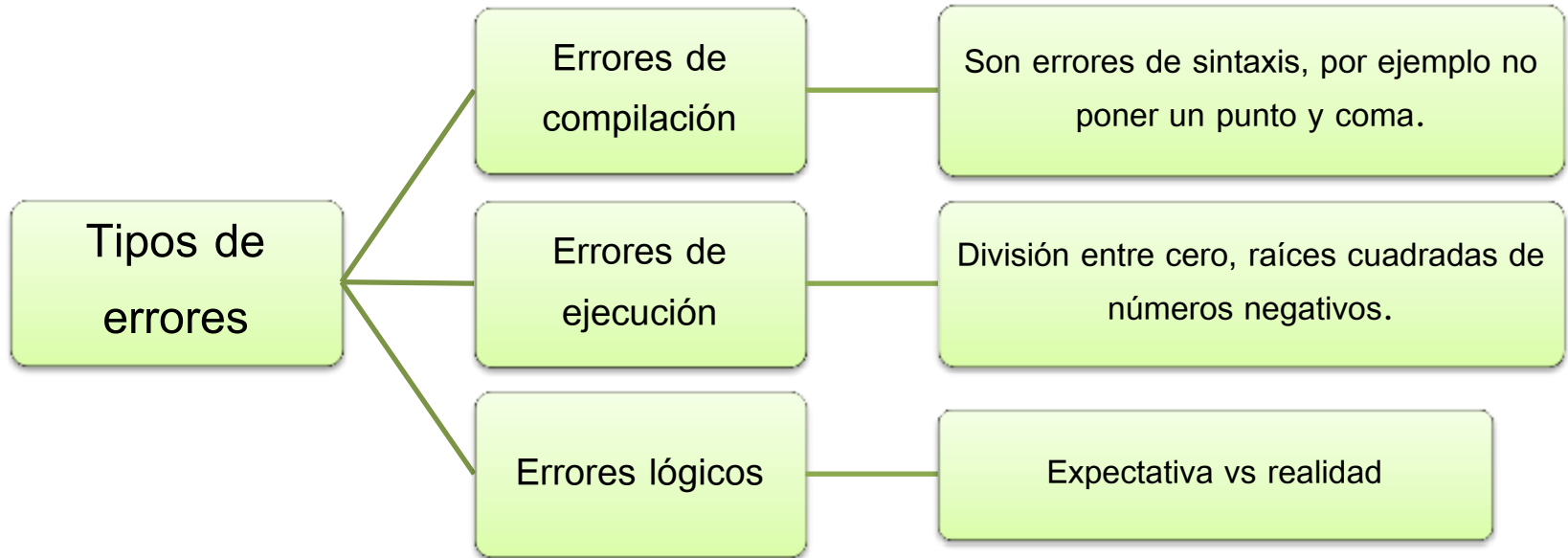
```
graph LR; A[Tipos de instrucciones] --- B[Instrucciones de entrada/salida]; A --- C[Instrucciones de cálculo]; A --- D[Instrucciones de control];
```

Instrucciones de ***entrada/salida***

Instrucciones de ***cálculo***

Instrucciones de ***control***

Verificación y depuración de un programa (bugs);



Programación estructurada;

La programación estructurada quiere decir escribir un programa de acuerdo a las siguientes reglas:

- ❖ Diseño modular
- ❖ Diseño descendente
- ❖ Cada modulo incluye las estructuras básicas de control: *secuencia*, *selección* y *repetición*.

La programación estructurada reúne técnicas que incorporan:

1. Recursos abstractos (*Divide and conquer*)
2. Diseño descendente (*top-down*)
3. Estructuras básicas



Teorema de la programación estructurada;

En 1996 Böhm y Jacopini demostraron que un *programa propio* puede ser escrito solamente usando tres tipos de estructuras de **control**.

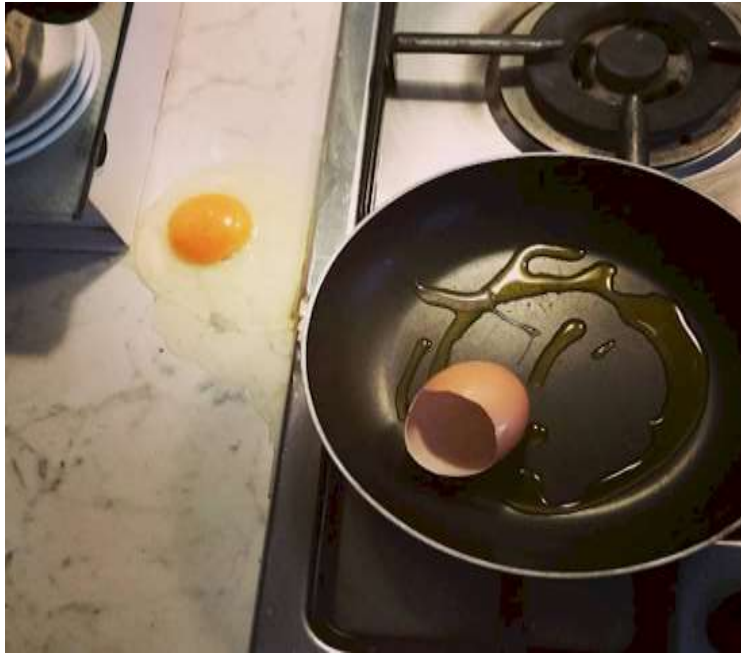


Secuenciales

Selectivas

Repetitivas

Características de los algoritmos;



El algoritmo de una receta de cocina tendrá:

Entrada: ingredientes y utensilios.

Proceso: elaboración de la receta.

Salida: terminación del plato

Calcular el valor de la suma

$$1+2+3+4+5+\dots+100$$

Algoritmo

Se utiliza una variable **Contador** como un contador que genere los sucesivos números enteros, y **Suma** almacenar las sumas parciales 1, 1+2, 1+2+3...

1. Establecer Contador a 1
2. Establecer Suma a 0
3. **mientras** Contador <= 100 **hacer**
 Sumar Contador a Suma
 Incrementar Contador en 1
 fin_mientras
4. Visualizar Suma

Ejemplo 1

Un cliente hace un pedido online a una tienda de tecnología. La tienda examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la tienda acepta el pedido; en caso contrario, rechazará el pedido. Escribir el algoritmo correspondiente.

Los pasos del algoritmo son:

1. Inicio.
2. Leer pedido.
3. Examinar la ficha del cliente.
4. Si el cliente es solvente, entonces aceptar pedido; en caso contrario, rechazar pedido.
5. Fin.

Ejemplo 2

Se desea diseñar un algoritmo para saber si un número es primo o no.

Algoritmo:

1. Inicio.
2. Poner x igual a 2 ($x=2$, x representa los divisores del número n que se busca).
3. Dividir n por x (n/x).
4. Si el resultado de n/x es entero, entonces n no es un número primo y saltamos al punto 7; en caso contrario, continuar el proceso.
5. Sumar 1 a x (x es igual a $x+1$).
6. Si x es igual a n , entonces n es un número primo; en caso contrario , regresar al punto 3.
7. Fin.

Probemos éste algoritmo con el número 7.

Ejemplo 3 (Opcional)

Realizar la suma de todos los números pares entre 2 y 11

Algoritmo:

1. Inicio.
2. Establecer SUM a 0.
3. Establecer NUM a 2.
4. Sumar NUM a SUM. El resultado será el nuevo valor de SUM (la suma).
5. Incrementar NUM en 2 unidades.
6. Si $\text{NUM} \leq 11$ regresar al paso 4; en caso contrario, escribir el ultimo valor de SUM y pasar al paso 7.
7. Fin

Comprobémoslo...

Escritura de un algoritmo;

Importante: es de suma importancia escribir bien un programa para poderlo leer posteriormente.

Indentación=Sangrado.

Algoritmo:

1. Inicio
2. tomar la tetera
3. llenarla con agua
4. encender fuego
5. poner tetera en el fuego
6. **mientras** no hierva el agua
7. esperar
8. tomar la bolsa de té
9. introducirla en la tetera
10. **mientras** no esté hecho el té
11. esperar
12. echar el té en la taza
13. Fin

Estructura general de un programa en C

```
#include <stdio.h> ← Archivo de cabecera
```

```
int main() ← Cabecera de función
```

```
{  
  ↑  
  Nombre de la función
```

```
... ← Sentencias
```

```
}
```

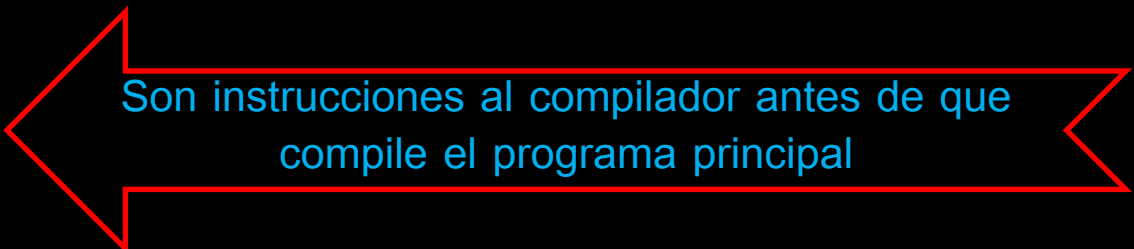
Directivas del preprocesador;

El preprocesador es un editor de texto inteligente.

Todas las directivas llevan signo de libro (#).

Habitualmente van al principio del programa, pero no es obligatorio.

```
#include  
#define
```



Son instrucciones al compilador antes de que compile el programa principal

Directivas del preprocesador

```
graph TD; A[Directivas del preprocesador] --> B[#include]; A --> C[#define]; B --> D[Indican al compilador que lea el archivo fuente que viene a continuación.]; C --> E[Indica al preprocesador que defina un ítem de datos u operación para el programa C.]
```

#include

Indican al compilador que lea el archivo fuente que viene a continuación.

#define

Indica al preprocesador que defina un ítem de datos u operación para el programa C.

#include

Esta directiva utiliza **archivos de cabecera** que son archivos que tienen extensión `.h` y que contienen código fuente.



Ejemplos:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

`#define`



```
#define TAM_LINEA 30
```

olo con un valor asociado.

Sustituirá TAM_LINEA
por el valor 30 cada
vez que aparezca en el
programa



Ejemplo 3 (Opcional)

Realizar la suma de todos los números pares entre 2 y 11

Algoritmo:

1. Inicio.
2. Establecer SUM a 0.
3. Establecer NUM a 2.
4. Sumar NUM a SUM. El resultado será el nuevo valor de SUM (la suma).
5. Incrementar NUM en 2 unidades.
6. Si $\text{NUM} \leq 11$ regresar al paso 4; en caso contrario, escribir el ultimo valor de SUM y pasar al paso 7.
7. Fin

Comprobémoslo...

Función main();

Todo programa en Lenguaje C debe tenerla, es la partida de inicio de un programa en C. Su estructura se ve así:

```
main()  
{  
  ... ← Bloque de  
}      sentencias
```

Nota: Una y solo una función main() se debe tener en un programa en C.

```
/*Ejemplo*/  
int main()  
{  
  obtenerdatos();  
  alfabetizar();  
  verpalabras();  
  
  return 0;  
}
```

Función C;

Una función C es un subprograma que devuelve un único valor, un conjunto de valores o realiza alguna tarea específica como E/S.

Un programa en C es un conjunto de funciones que se integran para crear una aplicación.

Estructura de una función en C;

```
tipo_retorno nombre_funcion(argumentos) Principio de la  
{ función  
    sentencias Cuerpo de la función  
    return; Retorno de la función  
} Fin de la función
```



```
/*Veamos un ejemplo...*/
```

Una función
aparece 3 veces
en un programa,
cuando se
declara, cuando
se manda a llamar
y cuando se
define.

/*Ejemplo*/

```
1.#include <stdio.h>
2.
3.void felicitar();
4.
5.int main()
6.{
7.    felicitar();
8.    return 0;
9.}
10.
11.void felicitar()
12.{
13.    printf("Feliz
14.cumpleaños\n");
15.}
```

Hora de programar;



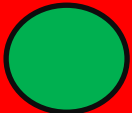
Comentarios;

Son información
sobre el
programa para
documentación,
solo para
programadores.

El compilador
ignora los
comentarios.

Ejemplo 1.

```
1.#include <stdio.h>
2./*
3.Programa: Demo1.c
4.Programador: Dennis Ritchie
5.Descripción: Primer programa en C
6. Fecha de creación: Diciembre 2010
7.Revisión: Niguna
8.* /
9.int main()
10.{
11. ...
12.}
```



Comentarios;

Ejemplo 2.

```
1.#include <stdio.h>
2.
3.//Los comentarios también pueden ir así
4.//Pero por cada nueva línea de comentarios
5.//Hay que volver a poner el par de diagonales
6.
7.int main()
8.{
9. ...
10.}
```

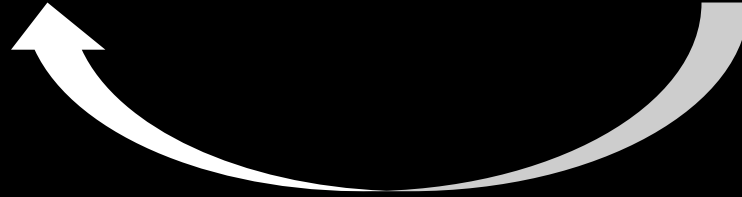
#include

Esta directiva también puede utilizar archivos fuente con extensión .c que fueron pre-codificados y se mandan a llamar en un programa.

Ejemplos:

```
/*info.c*/  
printf("Código 34529\n");
```

```
#include <personas.c>  
#include <info.c>
```



Probémoslo...

Conclusión:

```
graph TD; A[Conclusión:] --- B[Los archivos de cabera tienen extensión .h]; A --- C[Los archivos fuente tienen extensión .c];
```

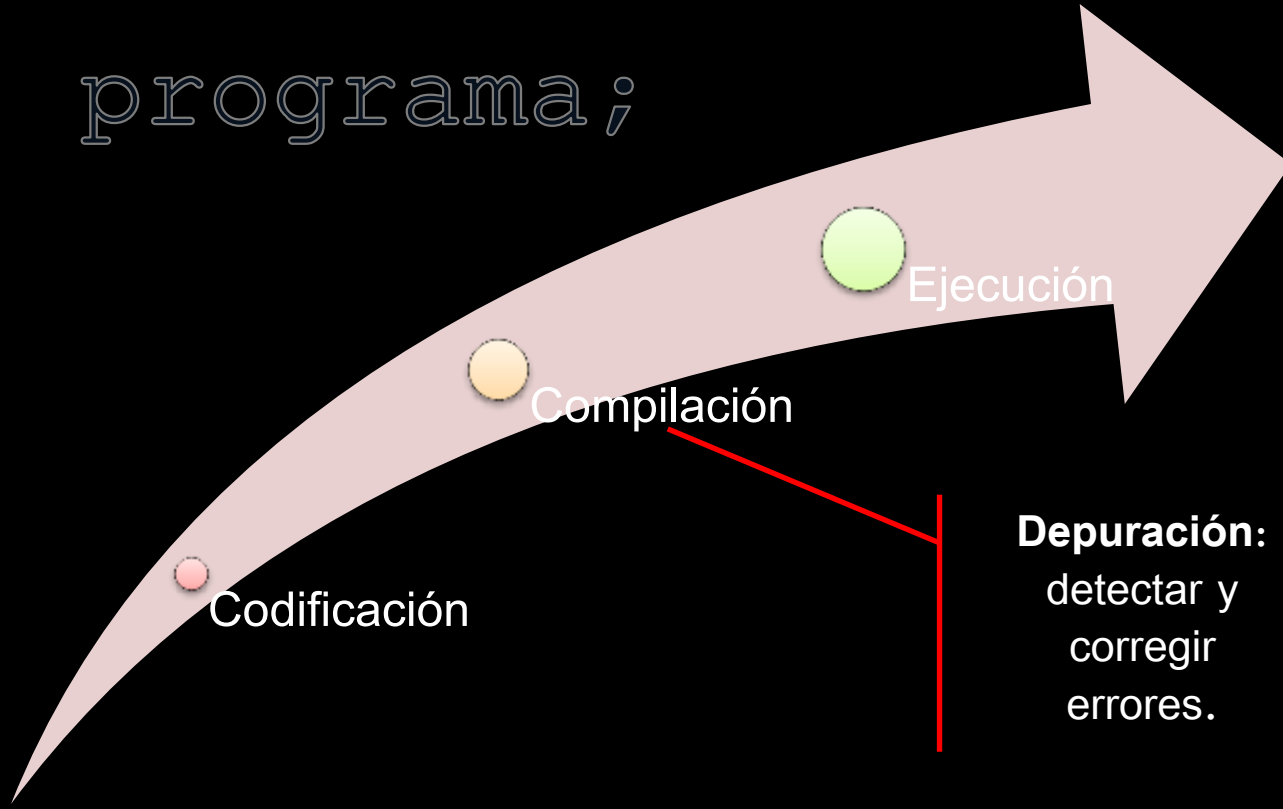
Los archivos de
cabera tienen
extensión .h

Los archivos
fuente tienen
extensión .c

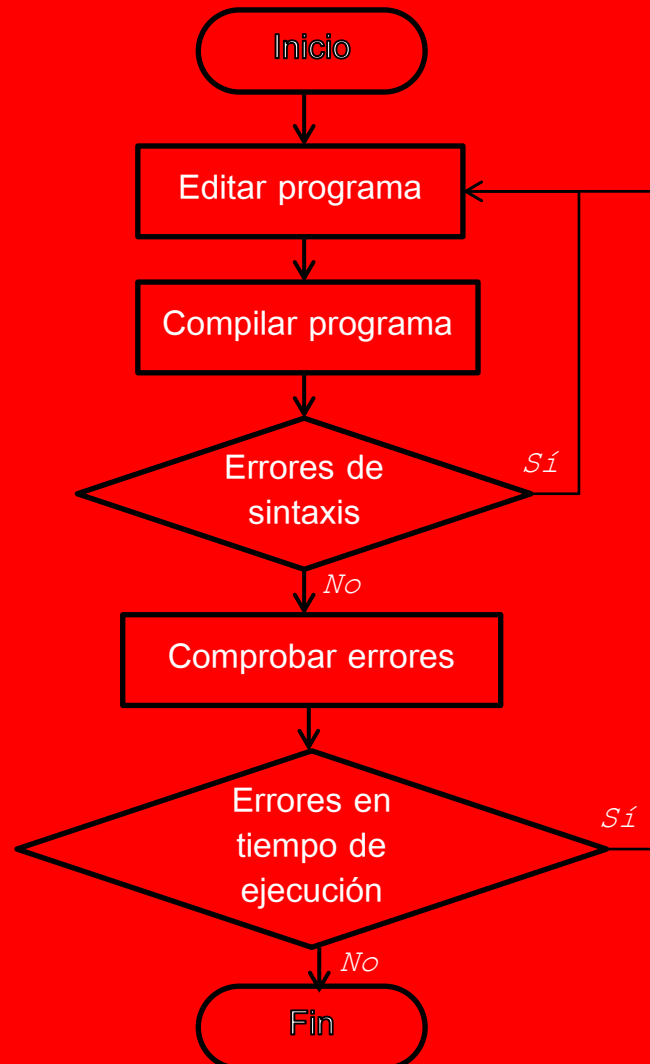
Creación de un programa;

1. Definir programa
2. Definir directivas del preprocesador
3. Definir declaraciones globales
4. Crear main()
5. Escribir cuerpo del programa
6. Crear sus propias funciones definidas por usted
7. Compilar, ejecutar y comprobar
8. Utilizar comentarios

Creación de un programa;

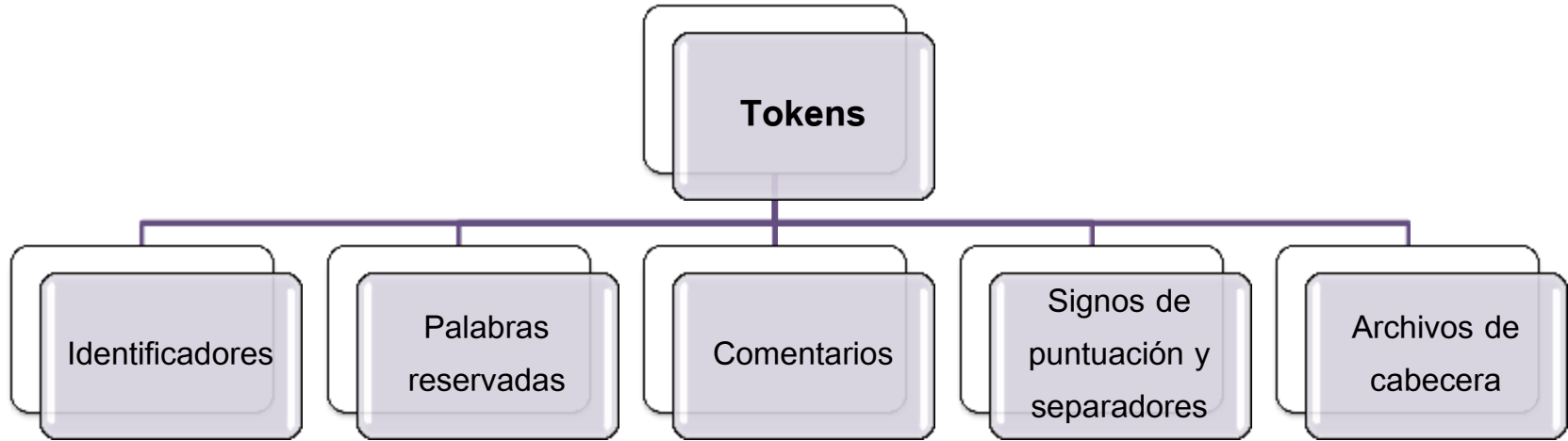


Proceso de
depuración de un
programa en C.



Elementos de un programa en C;

Un programa en C usa uno o más archivos que son traducidos en diferentes fases, una de ellas es el pre-procesado, el resultado de éste pre-procesado es una secuencia de Tokens:



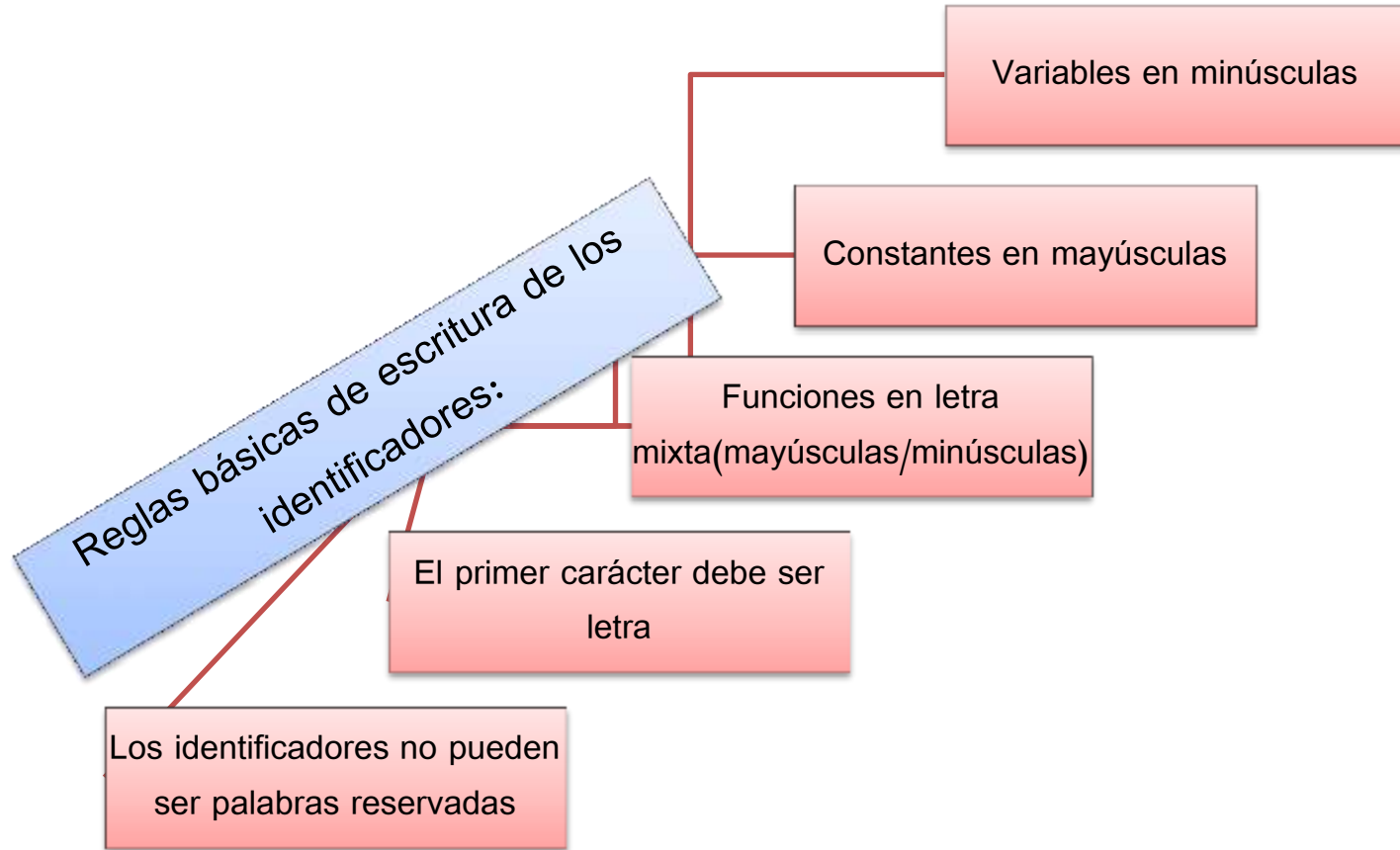
Identificadores;

Un identificador es una secuencia de caracteres, letras, dígitos y guiones. No uses palabras reservadas como identificadores.

Ejemplos:

nombre_usuario	indice	dia_mes_anio
num_mayor	num_menor	x
num_cuarto	sueldo_empleado	i

Nota: C es sensible a las mayúsculas, quiere decir que no es lo mismo un identificador num_mayor que Num_mayor que nUm_mayor.



Palabras reservadas;

Se conocen también como keywords o reserved words, un ejemplo antes visto es `void`, son palabras con un significado especial.

auto

break

case

char

const

continue

default

do

double

else

enum

extern

float

for

if

int

long

register

return

short

signed

sizeof

static

struct

switch

typedef

union

unsigned

void

volatile

while

Palabras reservadas;

c	o	n	t	i	n	u	e	c	u	r	a
s	r	e	t	u	r	n	h	o	l	a	q
f	t	c	x	v	a	r	f	n	e	h	l
l	p	h	b	c	e	b	o	s	m	j	f
o	d	a	r	o	a	i	n	t	r	a	s
a	o	r	e	t	u	s	h	a	x	c	w
t	a	k	a	i	f	y	e	r	i	i	i
o	y	d	k	e	l	i	h	w	o	f	t
k	i	p	r	a	n	f	s	n	e	l	c
p	o	p	e	y	e	s	h	o	r	t	h

Comentarios;

Son información sobre el programa para el programador.
El compilador ignora los comentarios.

Tienen el formato: `/*...*/`

Pueden extenderse varias líneas:

```
/* Éste programa calcula el interés  
   generado en un año para clientes  
   especiales.
```

```
*/
```

También tienen el formato: `//Inserte aquí su comentario.`

Signos de puntuación;

Todas las sentencias deben
terminar en punto y coma.

! @ # \$ % ^ & * () _ - + =
{ } : « » < > , . \ / ; [] `

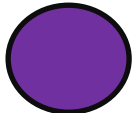
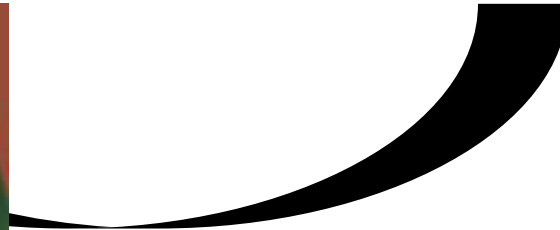
Archivos de cabecera;

Son archivos especiales que contienen declaraciones de elementos y funciones de la biblioteca.

`string.h` hace que el contenido de la biblioteca de cadenas esté disponible para el programa.

Ejemplo:

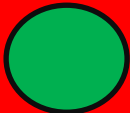
```
#include <string.h>
```



Tipos de datos en C;

Los tipos de datos simples de C son, básicamente, números. Los 3 tipos más comunes son: *enteros*, *coma flotante* y *caracteres*.

Tipo	Ejemplo	Tamaño en bytes	Rango Mínimo .. Máximo
char	`c`	1	0 .. 255
short	-15	2	-128 .. 127
int	1024	2	-32768 .. 32767
long	262144	4	-2147483648 .. 2147483647
float	10.5	4	1.5e-45 .. 3.4e38
double	0.00045	8	5.0e-324 .. 1.7e308
long double	1e-8	8	Igual que el double



Enteros (int);

Es el tipo de dato más famoso, son adecuados para trabajar con datos numéricos que no requieren punto decimal.

Declaración de variables

La forma tradicional de declarar una variable es escribir el *tipo de dato* y en seguida el *nombre de la variable*.

Formato:

<tipo de dato> <nombre de la variable> = <valor inicial>

Ejemplos:

```
int longitud;  
int valor=99;  
int var1,var2;  
int num1=5, num2=8;
```

Tipos de datos enteros.

Tipo C	Rango de valores	Uso recomendado
int	-32,768..+32,767	Aritmética de enteros, bucles for, conteo
unsigned int	0..65,535	Conteo, bucles for, indices
short int	-128..+127	Aritmetica de enteros, bucles for, conteo

Tipos de datos enteros largos.

Tipo C	Rango de valores
long	-2,147,483,548..2,147,483,647
unsigned long	0..+4,294,967,295

Ambos tipos requieren 4 bytes (32 bits) de almacenamiento.

Si se requiere C para desarrollar sistemas operativos o para hardware de computadora, sería útil escribir constantes enteras en *octal* o *hexadecimal*.

Break matemático;



Flotante (float/double) ;

Se usan para representar números reales con punto decimal como 3.141592 o números muy grandes como $1.85 \cdot 10^{15}$.

Ejemplos:

```
float num;  
float uno, dos, tres;  
float valor=19.2398;
```

Memoria:

El tipo float requiere 4 bytes de memoria, mientras double requiere 8 bytes y long double requiere 10 bytes.

Tipos de datos en punto flotante.

Tipo C	Rango de valores	Precisión
float	$3.4 * 10^{-38} \dots 3.4 * 10^{38}$	7 dígitos
double	$1.7 * 10^{-308} \dots 1.7 * 10^{308}$	15 dígitos
long double	$3.4 * 10^{-4932} \dots 1.1 * 10^{4932}$	19 dígitos

Caracteres (char);

Un carácter es cualquier conjunto de caracteres predefinidos o alfabeto. La mayoría de computadoras usan el conjunto de caracteres ASCII.

Tipo C	Rango de valores	Tamaño
Char	0...255	1 byte=8bits

Ejemplos:

```
char dato_car;  
Char letra='A';  
Char respuesta='S';
```

Ejemplo de utilidad:

Transformar minúsculas a mayúsculas.

```
char letra='a';  
...  
char otra_letra=letra-32;
```

De manera similar lo hacemos al revés.

Programémoslo...

Código ASCII (American Standard Code for Information Interchange);

Caracteres de control ASCII

DEC	HEX	Símbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de línea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	\$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(72	48h	H	104	68h	h
41	29h)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	
61	3Dh	=	93	5Dh]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	elCodigoASCII.com.ar		

ASCII extendido

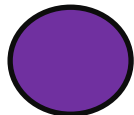
DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó
129	81h	ù	161	A1h	í	193	C1h	ł	225	E1h	ô
130	82h	é	162	A2h	ó	194	C2h	Ł	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	ł	227	E3h	Õ
132	84h	ä	164	A4h	ñ	196	C4h	Ł	228	E4h	ö
133	85h	à	165	A5h	Ñ	197	C5h	ł	229	E5h	Ö
134	86h	â	166	A6h	ª	198	C6h	Ł	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	ł	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	Ł	232	E8h	ß
137	89h	è	169	A9h	®	201	C9h	ł	233	E9h	Û
138	8Ah	è	170	AAh	¬	202	CAh	Ł	234	EAh	Ü
139	8Bh	ï	171	ABh	½	203	CBh	ł	235	EBh	Ù
140	8Ch	î	172	ACH	¼	204	CCh	Ł	236	ECh	Ý
141	8Dh	ï	173	ADh	¡	205	CDh	ł	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	Ł	238	EEh	ˆ
143	8Fh	Å	175	AFh	»	207	CFh	ł	239	EFh	˙
144	90h	É	176	B0h	⋮	208	D0h	Ł	240	F0h	±
145	91h	æ	177	B1h	⋮	209	D1h	ł	241	F1h	±
146	92h	Æ	178	B2h	⋮	210	D2h	Ł	242	F2h	¼
147	93h	ð	179	B3h	⋮	211	D3h	ł	243	F3h	½
148	94h	ò	180	B4h	⋮	212	D4h	Ł	244	F4h	¾
149	95h	ó	181	B5h	⋮	213	D5h	ł	245	F5h	§
150	96h	û	182	B6h	⋮	214	D6h	Ł	246	F6h	÷
151	97h	ü	183	B7h	⋮	215	D7h	ł	247	F7h	ˆ
152	98h	ÿ	184	B8h	⋮	216	D8h	Ł	248	F8h	˙
153	99h	Œ	185	B9h	⋮	217	D9h	ł	249	F9h	˙
154	9Ah	Œ	186	BAh	⋮	218	DAh	Ł	250	FAh	˙
155	9Bh	ø	187	BBh	⋮	219	DBh	ł	251	FBh	˙
156	9Ch	£	188	BCh	⋮	220	DCh	Ł	252	FCh	˙
157	9Dh	Ø	189	BDh	⋮	221	DDh	ł	253	FDh	˙
158	9Eh	×	190	BEh	⋮	222	DEh	Ł	254	FEh	˙
159	9Fh	f	191	BFh	⋮	223	DFh	ł	255	FFh	˙

Lógico;

Los compiladores de C que siguen la norma ANSI no incorporan el tipo de dato lógico. Cuyos valores son `verdadero(true)` y `falso(false)`.

Valores distinto de cero	representa <i>true</i> (verdadero)
0	representa <i>false</i> (falso)

Este tipo de dato es de los más importantes en el lenguaje C, para ello utilizamos el tipo de dato `int`.



Constantes

```
graph TD; A[Constantes] --- B[Literales]; A --- C[Definidas]; A --- D[Enumeradas]; A --- E[Declaradas]
```

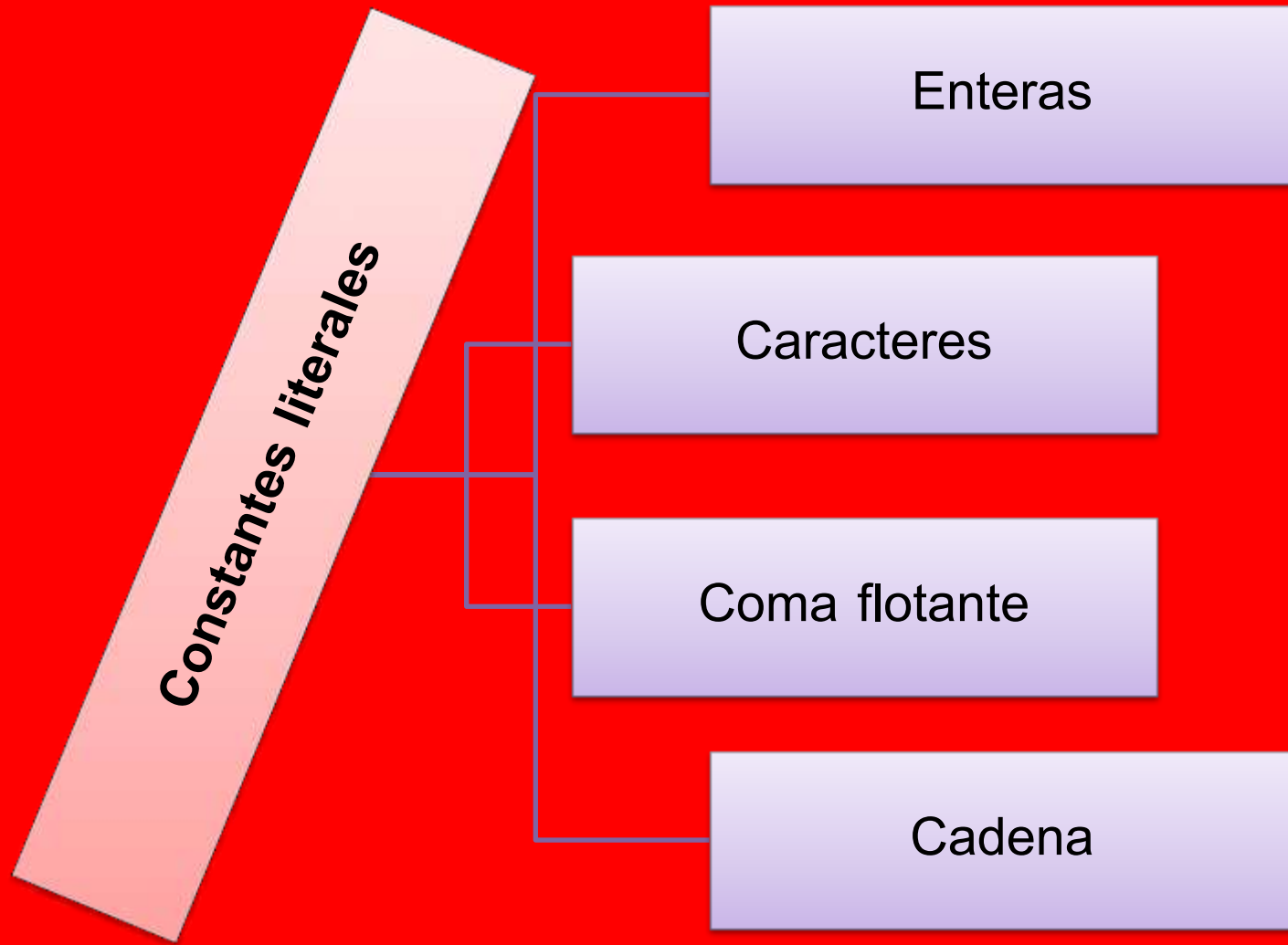
Literales

Definidas

Enumeradas

Declaradas





Constantes enteras;

Reglas para escribir constantes enteras:

- No usar signos de puntuación

`54321` en vez de `54.321`

- Para forzar un valor entero al tipo long, ponerle una L o l al final.

`2048` es entero

`2048L` es entero largo

- Para representar un entero en octal (base 8), anteponer un 0.

Formato decimal `123`

Formato octal `0777`

- Para representar un entero en hexadecimal (base 16), este debe de estar precedido por "0x".

Formato hexadecimal `0xFF3A` (es lo mismo poner "0x" que "OX")

Programémoslo...

Constantes caracteres;

Son tipo `char` y son caracteres del código ASCII encerrados entre apóstrofes.

`'B'`

`'C'`

`'R'`

Nota: Una característica especial de una constante carácter es que además de código ASCII también soportan caracteres especiales tales como las secuencias de escape.

Código de escape	Significado	ASCII decimal	ASCII hexadecimal
<code>'\n'</code>	Nueva línea	10 o 13	0D o 0A
<code>'\r'</code>	Retorno de carro	13	0D
<code>'\t'</code>	Tabulación	9	09
<code>'\v'</code>	Tabulación vertical	11	0B
<code>'\a'</code>	Alerta (Sonido)	7	07
<code>'\0000'</code>	Número octal	Todos	Todos
<code>'\xhh'</code>	Número hexadecimal	Todos	Todos

Constantes caracteres;

Por ejemplo: el carácter sigma (Σ) en código ASCII es 228 y en hexadecimal E4, para representar éste símbolo utilizamos una combinación del prefijo `\x` y a continuación el número ASCII hexadecimal.

```
char sigma = '\xE4';
```

Código de escape	Significado	ASCII decimal	ASCII hexadecimal
<code>'\n'</code>	Nueva línea	10 o 13	0D o 0A
<code>'\a'</code>	Alerta (Sonido)	7	07
<code>'\xE4'</code>	Sigma (Σ)	228	E4

Programémoslo...

Aritmética de caracteres;

Ejemplos:

```
char t;
```

```
t='T+5'; //Suma 5 al caractes ASCII, se guarda Y en t.
```

```
int j='p'; // p en código ASCII vale 80, j guarda ese valor.
```

```
int m;
```

```
m=m+'a'-'A'; //m entrega el código ASCII de M (mayúscula).
```

Constantes de coma flotante;

Representan un número real, representan aproximaciones en lugar de valores exactos(tenerlo en cuenta).

Existen 3 tipos:

float 4 bytes

double 8 bytes

long double 10 bytes

Ejemplos:

82.3487

0.63

83.0

47e-4

Notación científica:

2.5E4 es igual a 25000

5.437E-3 es igual a 0.005437

Constantes cadena;

Son simplemente cadenas(secuencias de caracteres) encerrados entre comillas.

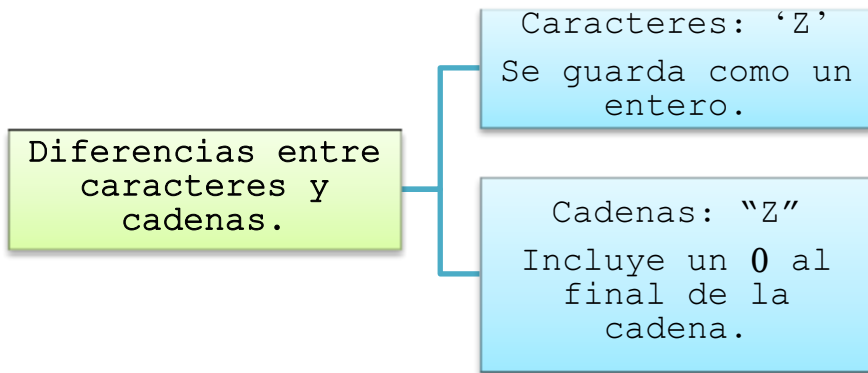
Ejemplos:

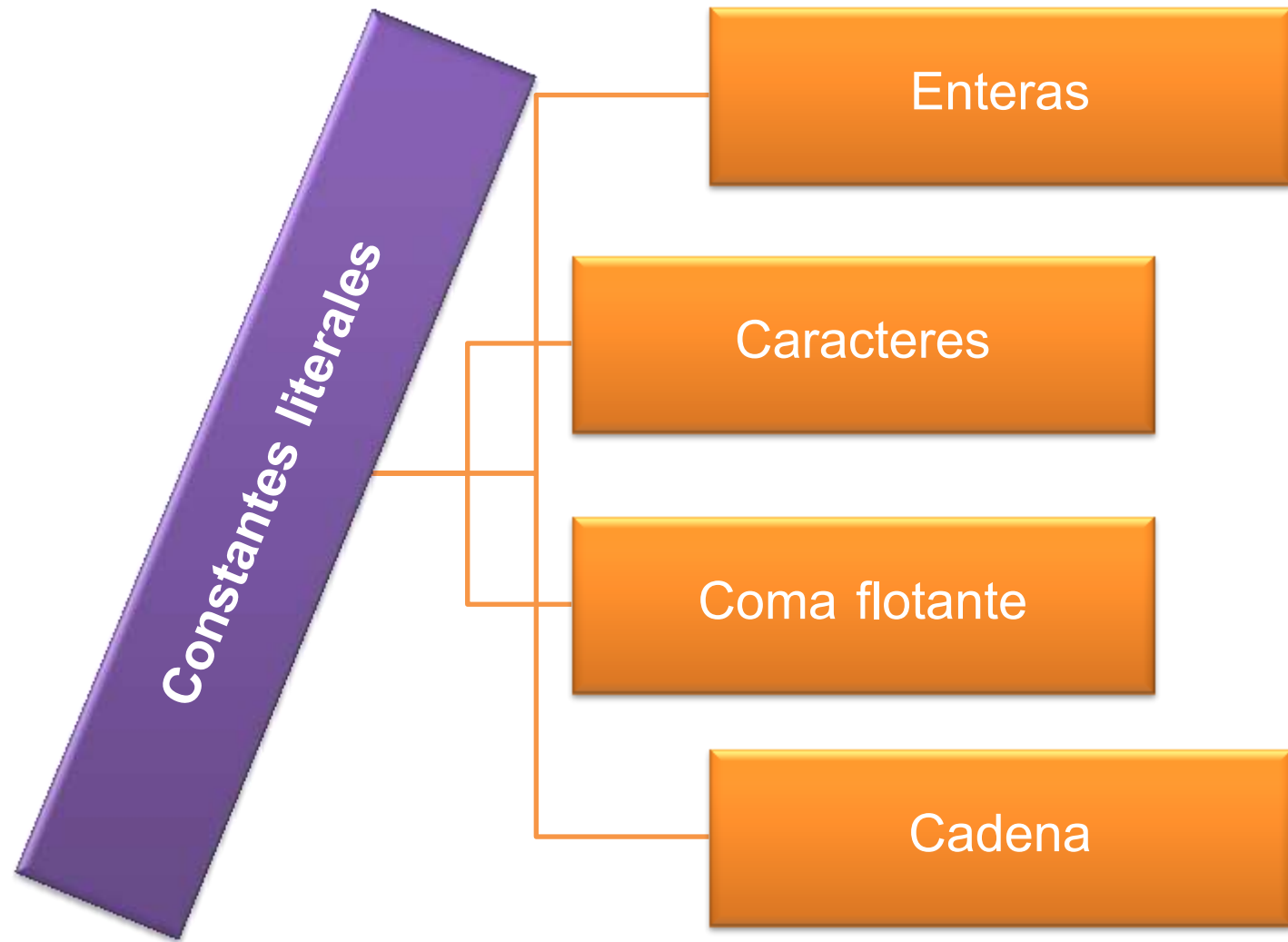
"321"

"21 de mayo"

"esto es una cadena"

Nota: El compilador C inserta siempre un carácter llamado nulo al final de las constantes de las cadenas para saber que ahí termina la cadena.





Constantes definidas;

Utilizan la directiva de preprocesador `#define`, reciben nombres simbólicos.

Ejemplos:

```
#define SALTO \n
#define E 2.71
#define MAX 100
```

Si C encuentra por ejemplo SALTO lo sustituye por `\n` y así con todos los ejemplos.

Las siguientes líneas de código son equivalentes:

```
printf("El precio es: \n",100);
printf("El precio es: SALTO",MAX);
```

Programémoslo...

Constantes enumeradas;

Permiten crear listas de elementos afines. Un clásico de clásicos son los colores.

Ejemplo:

```
enum colores{negro, marrón, rojo, naranja, amarillo};
```

El compilador asigna negro a 0, marrón a 1, etc...

Constantes declaradas;

Las constantes declaradas pueden ser `const` y `volatile`.

Formato: `const <tipo> <nombre>=<valor>;`
(si se omite el tipo C utilizar `int` por default).

Ejemplos:

```
const char CAR='@';  
const int OCT=0233;  
const char CAD[]="Mi curso de lenguaje C";
```

El tipo de variable `const` especifica que el valor de una variable será fijo y cualquier intento de modificar el valor dará error. La palabra reservada `volatile` pero su valor si puede ser modificado.

Diferencias y ventajas entre `const` y `#define`.

`const` especifica un tipo de dato.

- El compilador genera código más eficiente, el tipo de dato da mejores condiciones de operación (detección de errores más fácil).

`#define` no utiliza el operador `(=)` y no termina en `(;)`.

- `#define` sólo existe en el texto del programa y no ocupa espacio en tiempo de ejecución.

Variables en C;

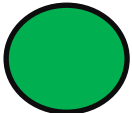
Es una dirección con nombre en memoria donde se almacena un valor de un cierto tipo de dato.

Todas la variables tienen un identificador válido que generalmente describe su propósito.

A diferencia de las constantes el valor de una variable en C sí puede ser modificado.

Ejemplos:

```
char resto_acumulado;  
int _porcentaje;  
float var1;
```



Declaración de una variable;

Es una sentencia que da información de la variable al compilador.

Formato: `<tipo> <variable>`

`tipo` es el nombre de un tipo de dato de C.

`variable` es el identificador(nombre) valido en C.

Más ejemplos:

```
long dNum;  
double HorasTrabajas;  
float HorasPorDia;  
float PromedioEscolar;  
short dia_del_mes;
```

Nota: debes declarar variables antes de utilizarlas, ya sea al principio de un archivo o bloque de código o al principio de una función.

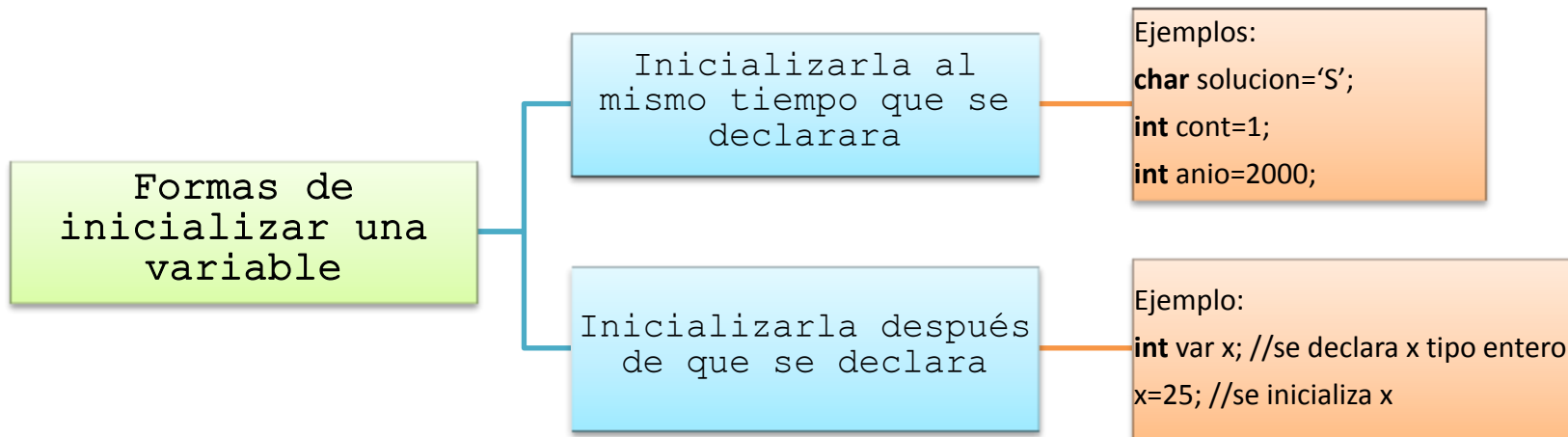
Programémoslo...

Inicialización de una variable;

Hemos hecho ya algunos programas haciendo uso de un **valor inicial** cuando declaramos una variable.

Formato: `<tipo> <variable> = <expresión>`

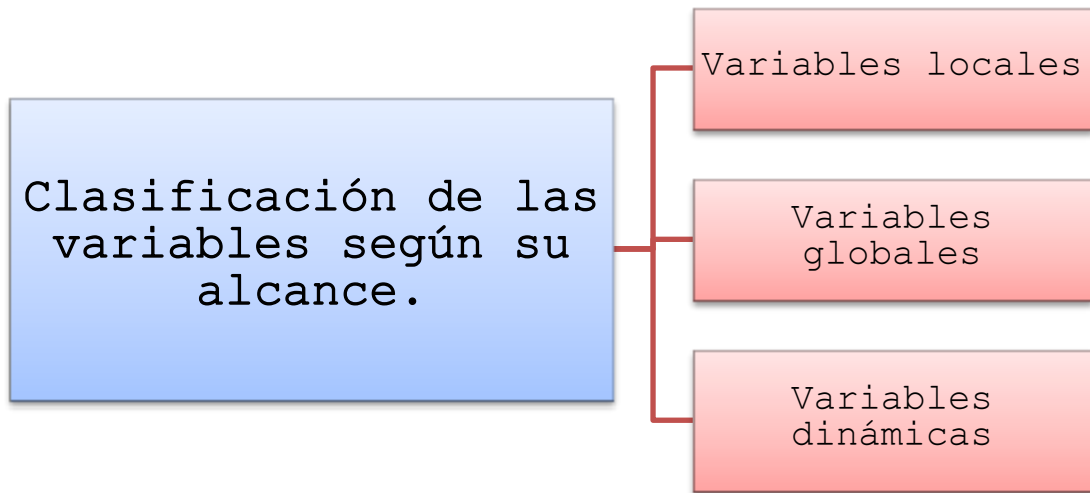
El **valor inicial** de una variable es una expresión válida cuyo valor es del mismo tipo de **tipo**.



Duración de una variable;

Dependiendo del lugar donde se declare una variable, ésta puede ser temporal, ya sea dentro de un bloque o de una función.

En programación la zona en la que está activa una variable se llama *ámbito* o *alcance(scope)*.



Variables locales;

También llamadas automáticas. Son aquellas que se declaran dentro de una función y solo están activas dentro de tal función.

Reglas para variables locales:

1. No pueden ser modificadas por ninguna sentencia externa a la función donde pertenecen.
2. Los identificadores de las variables locales no son únicos. Por ejemplo: el nombre *dragón* puede estar declarado en varias funciones.
3. Las variables locales solo existen en memoria(RAM) hasta que se ejecutan (Regla importante).

Programémoslo...

Variables globales;

Son aquellas que se declaran fuera de la función y por lo tanto son visibles para cualquier función incluyendo `main()`.

Diferencias entre
variables locales
y globales.

Locales, desaparecen cuando
su bloque termina.

Globales, es visible desde la
sentencia donde se declara
hasta el final del programa
(archivo fuente).

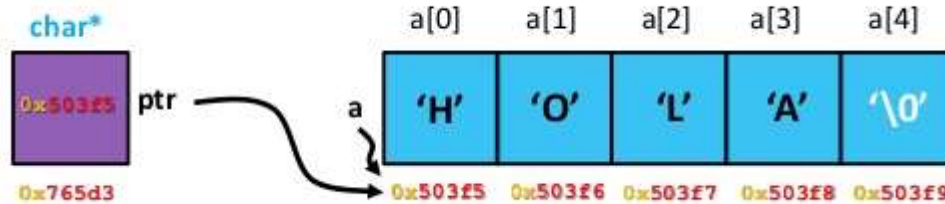
Nota: Se debe evitar usar muchas
variables globales en un programa
en C, ya que utilizan más memoria
y la detección de errores se
vuele más difícil.

Programémoslo...

Variables dinámicas;

Son variables que se crean a su petición y se libera su espacio cuando ya no se necesita. Pueden ser accesibles a múltiples funciones.

En algunos casos funciona semejante a variables locales y en otros a variables globales.



```
1. /*Este ejemplo ilustra los diferentes tipos de variable en C.*/
2. #include <stdio.h>
3. int Q; //Alcance o ámbito global Q, variable global.
4. void main()
5. {
6.     int A; //Local a main() A, variable local.
7.     A=123; Q=2;
8.     {
9.         int B; //Primer subnivel a main() B, variable local.
10.        B=124;
11.        A=500;
12.        Q=6;
13.        {
14.            int C; //Subnivel más interno de main() C, variable local.
15.            C=120;
16.            B=10;
17.            A=898;
18.            Q=3;
19.        }
20.    }
21. }
```

Entradas y salidas;

Tipos de
instrucciones

Instrucciones de
entrada/salida

Instrucciones de ***cálculo***

Instrucciones de ***control***

El usuario interactúa con un programa a través de datos de entrada y salida. El archivo cabecera antes visto `stdio.h` proporciona funciones de E/S.



Salida;

La salida en pantalla es la más común y además puede ser formateada. Hablamos de la función `printf()`, transforma datos binarios a ASCII.

Ejemplo:

```
suma=0;
suma=suma+10;
printf("La suma es: %d",suma);
```

Output:

La suma es 10

Recordemos que `%d` es un código de formato para imprimir enteros.

La función `printf()` permite un número de argumentos indefinidos por lo que se pueden transmitir cuantos se desee.

Los códigos de formato más utilizados y su significado:

%d	El dato se convierte a entero decimal.
%o	El dato entero se convierte a octal.
%x	El dato entero se convierte a hexadecimal.
%u	El dato entero se convierte a entero sin signo.
%c	El dato se considera tipo carácter.
%e	El dato se convierte de float a notación científica.
%f	El dato tipo float se imprime con parte entera y parte dígitos de precisión.
%g	El dato se convierte en notación científica o float según la representación más corta.
%s	El dato ha de ser una cadena de caracteres.
%lf	El dato se considera de tipo double.

Programémoslo...

C también utiliza *secuencias de escape* para visualizar caracteres que no están representados por símbolos tradicionales.

Ejemplo:

```
printf("Error\n Gracias por su visita\n Pulsar para continuar\n");
```

Output:

```
Error
Gracias por su visita
Pulsar para continuar
```

Código de escape	Significado	ASCII decimal	ASCII hexadecimal
'\n'	Nueva línea	10 o 13	0D o 0A
'\r'	Retorno de carro	13	0D
'\t'	Tabulación	9	09
'\v'	Tabulación vertical	11	0B
'\a'	Alerta (Sonido)	7	07
'\0000'	Número octal	Todos	Todos
'\xhh'	Número hexadecimal	Todos	Todos



Entrada;

La entrada de datos a un programa más común es a través del teclado. La función por excelencia para entrada formateada es `scanf()`.

Ejemplo:

```
5. const float IVA=0.16;
6. float sin_iva, con_iva;

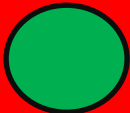
7. printf("Escriba la cantidad sin IVA: ");
8. scanf("%f",&sin_iva);

9. con_iva=sin_iva*IVA+sin_iva;

10.printf("La cantidad con iva es: %f",con_iva);
```

Las variables que capta `scanf()`, las capta con la técnica paso por referencia, para ello se deben preceder por `&`.

Programémoslo...



Salida de cadenas de caracteres;

La función `printf()` puede dar salida a cualquier dato. Asociándole el código de formato correspondiente.

Ejemplo:

```
4. char materia[]="electronica";  
5. printf("%s",materia);
```

El compilador de C cuenta con una función específica para cadenas `puts()`. La línea de código 5 se puede sustituir por:

```
4. char materia[]="electronica";  
5. puts(materia);
```

Programémoslo...

Entrada de cadenas de caracteres;

La entrada de cadenas se hace con la función `scanf()` y el código de formato `%s`.

```
/*Código:*/  
6. char titulo[30];  
7. printf("Escriba el titulo: ");  
8. scanf("%s",titulo);  
9. Printf("El titulo es: %d",titulo);
```

Nota: El identificador titulo no tienen que ir precedido de & cuando se trata de cadenas.

Input:

El llano en llamas

Output:

El titulo es: El llano en llamas

Entrada de cadenas de caracteres;

C tiene una función específica para la captación de cadenas de caracteres `gets()`, .

```
/*Código usando gets y puts:*/
```

```
6. char titulo[30];  
7. puts("Escriba el titulo: ");  
8. gets(titulo);  
9. printf("El titulo es: %d",titulo);
```

Nota: Siempre hay que definir las cadenas con más espacio del previsto.



Operadores y expresiones;

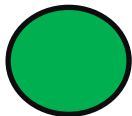
Los programas en C tienen datos, sentencias y *expresiones*. Las expresiones son normalmente expresiones o ecuaciones matemáticas como `10+4`.

C soporta un conjunto potente de operadores *unarios*, *binarios* y de *otros tipos*.

Sintaxis: `<variable> = <expresión>`

¿Qué puede contener una expresión?

1. *Valores constantes (como 25)*
2. *Variables simples (como 'X')*
3. *Funciones (como pow(x,2))*



Operador de asignación;

El operador = asigna el valor que escribes a la derecha del identificador situado a su izquierda.

```
/*Ejemplos:*/
```

```
etiqueta=235490;  
coordenada_x=34;
```

```
/*asignaciones especiales*/
```

```
x=y=z=17; ←  
//Ésta sentencia equivale a:  
x=(y=(z=17));
```

El operador de asignación asocia de derecha a izquierda.

Operadores de asignación;

Sirven para abreviar notación matemática, por ejemplo si queremos si queremos sumar $x+1$ y guardar el valor en x podemos escribir $x+=1$.

Símbolo	Uso	Descripción
=	$x=y$	Asigna el valor de x a y .
+=	$x+=y$	Suma y a x y asigna el resultado a la variable x .
-=	$x-=y$	Resta y de x y asigna el resultado a la variable x .
=	$x=y$	Multiplica x por y y asigna el resultado a la variable x .
/=	$x/=y$	Divide x entre y y asigna el resultado a la variable x .
%=	$x\%=y$	Divide x entre y de forma entera y asigna el resto a x .

Equivalencia de operadores de asignación.

Operador	Sentencia abreviada	Sentencia no abreviada
<code>+=</code>	<code>p+=q;</code>	<code>p=p+q;</code>
<code>-=</code>	<code>p-=q;</code>	<code>p=p-q;</code>
<code>*=</code>	<code>p*=q;</code>	<code>p=p*q;</code>
<code>/=</code>	<code>p/=q;</code>	<code>p=p/q;</code>
<code>%=</code>	<code>p%=q;</code>	<code>p=p%q;</code>

Estos operadores de asignación suponen un ahorro en la escritura, por lo que algunos programadores se acostumbran a utilizarlos.

Programémoslo...

Operadores aritméticos;

Éstos operadores siguen las reglas algebraicas tradicionales de jerarquía o prioridad.

`/*Ejemplo:*/`

Consideremos la expresión: $4+6*2$

¿Cuál sería su valor?

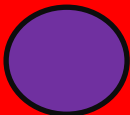
20 o 16

Respuesta: 16, ya que primero se ejecuta la multiplicación y posteriormente la suma.

Operadores aritméticos;

Debemos considerar que los operadores en ocasiones trabajan diferente según los tipos de datos de los operandos.

Operador	Tipos enteros	Tipos reales	Ejemplos
+	Suma	Suma	$x+y$
-	Resta	Resta	$f-g$
*	Producto	Producto	$m*z$
/	División entera: cociente	División en coma flotante	b/e
%	División entera: resto		d/t



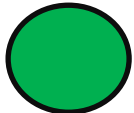
Programémoslo...

Operadores de incremento y decremento;

Una de las características más útiles que incorpora C son los operadores especiales ++ y --, incrementación y decrementación respectivamente.

++ suma 1 y -- resta uno cada vez que se aplica a una variable.

Incrementación	Decrementación
++n	--n
n+=1	n-=1
n=n+1	n=n-1



Operadores de incremento y decremento;

Éstos operadores tienen la propiedad que permite usarlos como prefijos o sufixo (dando un resultado diferente).

```
++n; n++; //tienen el mismo efecto  
--n; n--; //también tienen el mismo efecto
```

```
/*Ejemplo 1:*/
```

```
int n,m;  
n=1;  
m=n++;  
/*Incrementa 1 al pasar la sentencia*/  
printf("m=%d,n=%d",m,n);
```

Output:

m=1 n=2

Operadores de incremento y decremento;

Si los operadores ++ y -- están de prefijos el incremento o decremento se realiza antes de la asignación, pero si éstos están de sufijos el incremento o decremento se hace después de la asignación.

```
/*Ejemplo 2:*/  
  
int n,m;  
n=1;  
m=++n;  
/*Incrementa 1 en la misma sentencia*/  
printf("m=%d,n=%d",m,n);
```

Output:

m=2 n=2

Programémoslo...



Operadores relacionales;

C carece de tipos de datos lógicos o booleanos para representar los valores **verdadero** y **falso**, en su lugar usamos el tipo **int**.

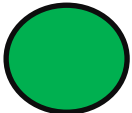
0	falso
Distinto de cero	verdadero

Sintaxis: <expresión_1> <operador_relacional> <expresión_2>

*/*Ejemplos:*/*

3 < 8 x + y != 0 m < 100 p >= 5

Los operadores relacionales normalmente se usan dentro de sentencias de selección o de repetición como while, switch o for.



Operadores relacionales;

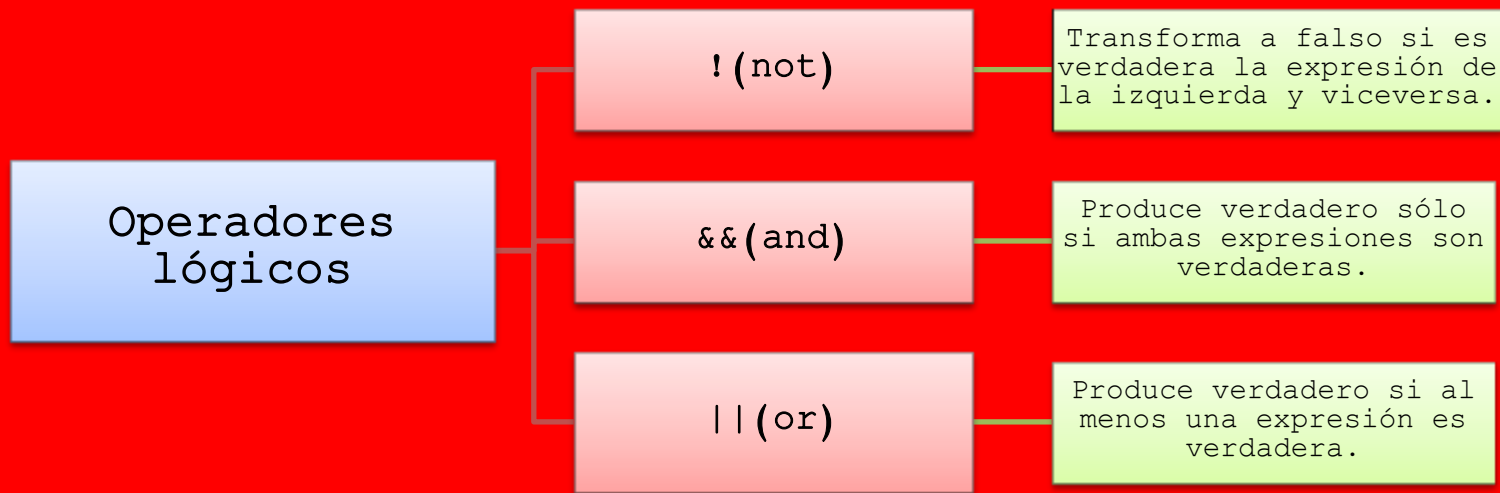
Operador	Significado	Ejemplo
==	Igual a	x==y
!=	Diferente de	x!=y
>	Mayor que	x>y
<	Menor que	x<y
>=	Mayor o igual que	x>=y
<=	Menor o igual que	x<=y

Nota: es muy común incluso entre los programadores experimentados, confundir el operador de asignación (=) con el operador de igualdad (==).

Operadores lógicos;

Los operadores lógicos también se denominan *operadores booleanos* en honor de George Boole, creador del álgebra de Boole.

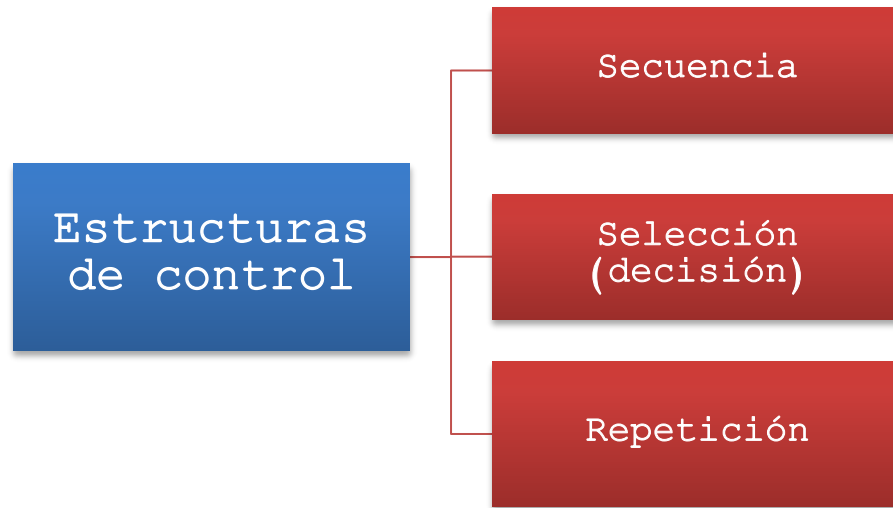
Sintaxis: <expresión_1> <operador_lógico> <expresión_2>



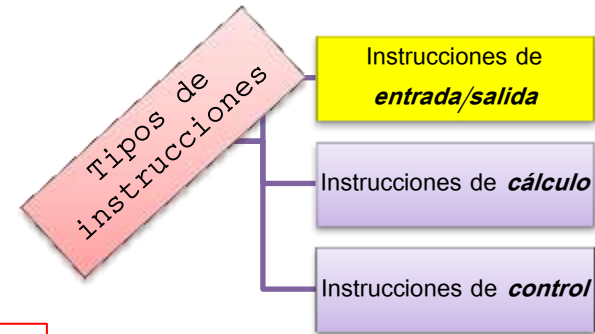
Hagamos sus tablas de verdad...

Estructuras de control;

Las estructuras de control controlan el flujo de ejecución de un programa o función. Tienen un punto de entrada y uno de salida.



Dan capacidad de controlar que sentencias se ejecutan y en que momento.



Nota: Hasta el momento solo hemos usado el flujo secuencial.

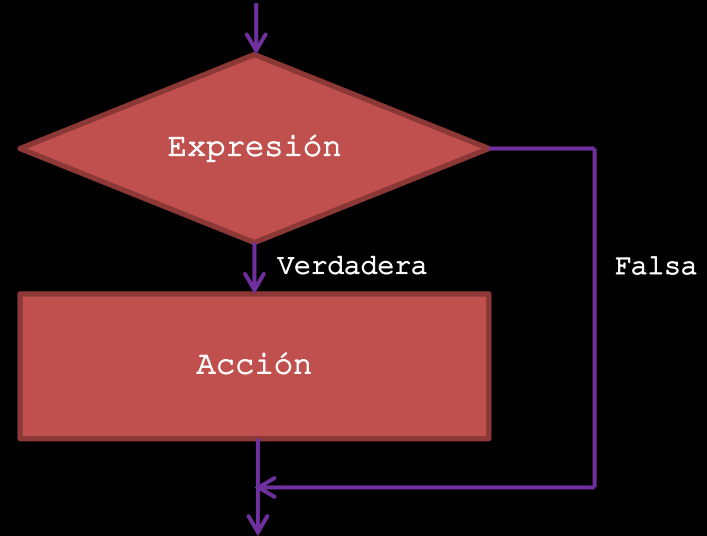
Sentencia if;

En el lenguaje C, `if` es la estructura de control principal.

Sintaxis: `if (<expresión>) <sentencia o acción>;`

`expresión` es una expresión entera lógica
`sentencia` es cualquier sentencia ejecutable
que se ejecutará si y sólo si toma
valor distinto de cero.

Cuando se llega a la línea de código
donde se encuentra `if` se evalúa la
expresión entre paréntesis, si la
expresión es verdadera se ejecuta `if`,
de lo contrario se omite la acción.

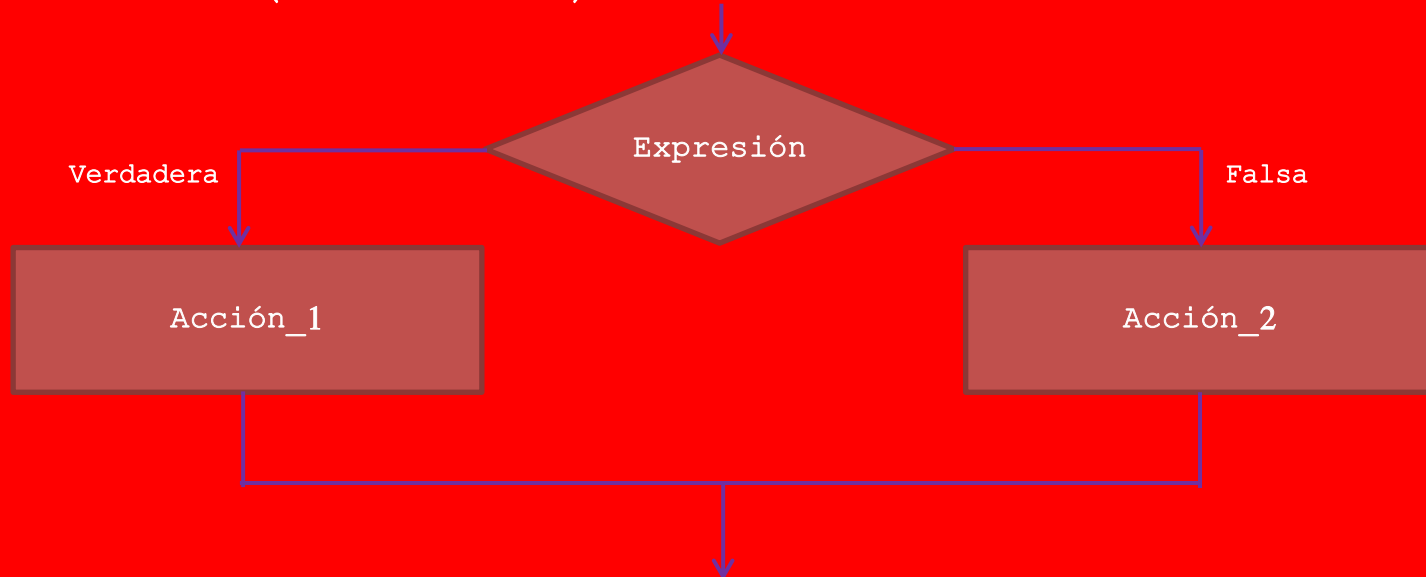


Programémosla...

Sentencia if-else;

Esta sentencia tiene dos alternativas, se evalúa la expresión, si es verdadera se ejecuta la primera acción en caso contrario se ejecuta la segunda.

Sintaxis: `if (<expresión>) <Acción_1>; else <Acción2> ;`



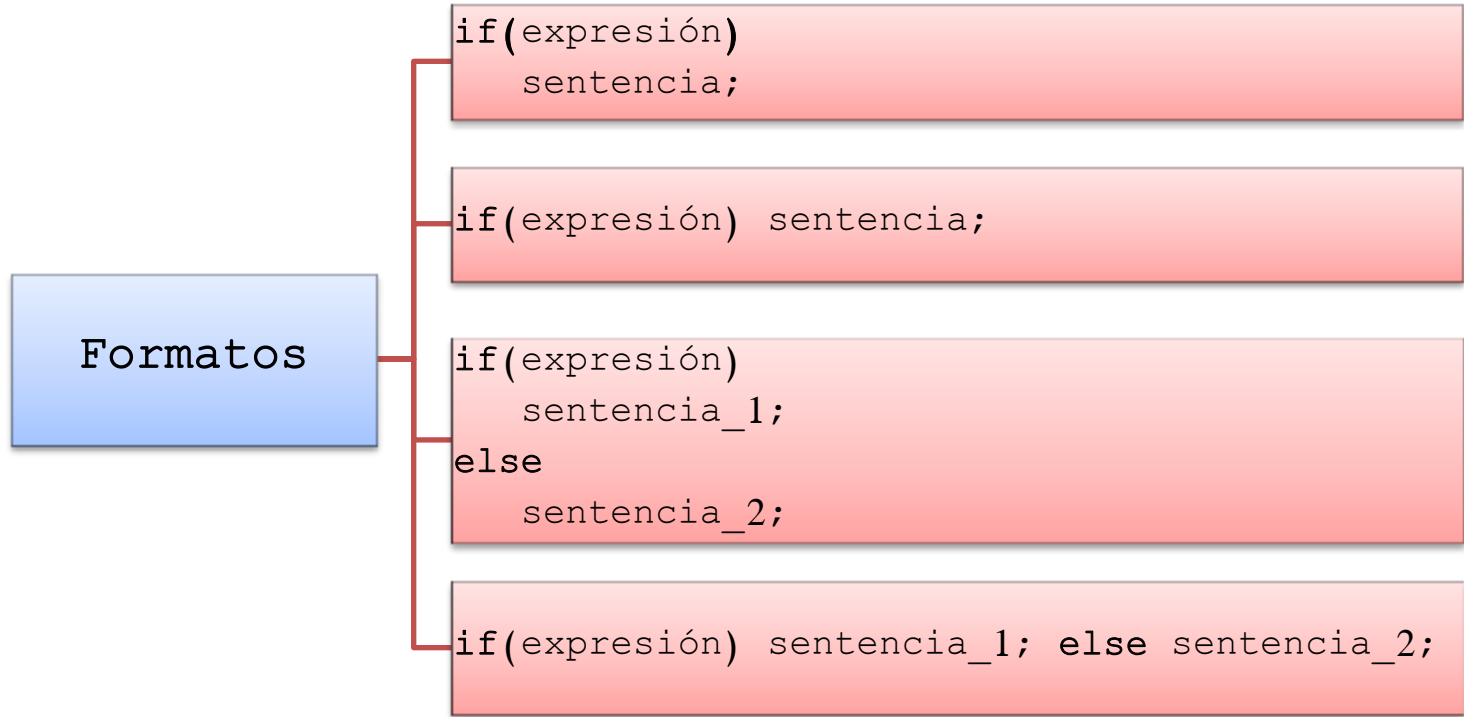
Ejemplo 1

Un cliente hace un pedido online a una tienda de tecnología. La tienda examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la tienda acepta el pedido; en caso contrario, rechazará el pedido. Escribir el algoritmo correspondiente.

Los pasos del algoritmo son:

1. Inicio.
2. Leer pedido.
3. Examinar la ficha del cliente.
4. Si el cliente es solvente, entonces aceptar pedido; en caso contrario, rechazar pedido.
5. Fin.

Sentencia if-else;



Programémosla...

Sentencia switch;

Switch es útil cuando la selección se basa en el valor de una variable o una expresión simple llamada **selector**.

Se usa para seleccionar una de entre múltiples opciones (menús).

Sintaxis:

```
switch(selector)
{
    case etiqueta1: sentencia1; break;
    case etiqueta2: sentencia2; break;
    .
    .
    .
    case etiqueta_n: sentencia_n; break;
    default: sentencias_d; /*Opcional*/
}
```

Selector sólo puede ser de tipo int o char.

etiqueta es un valor único, constante y todas las etiquetas deben ser diferentes.

Break termina la ejecución del switch.

default ejecuta una sentencia en caso de que se introduzca un valor de *selector* no incluido.

Programémosla...

Bucles;

Un bucle es cualquier construcción de programa que repite una sentencia o secuencia de sentencias un número de veces.

Bucles;

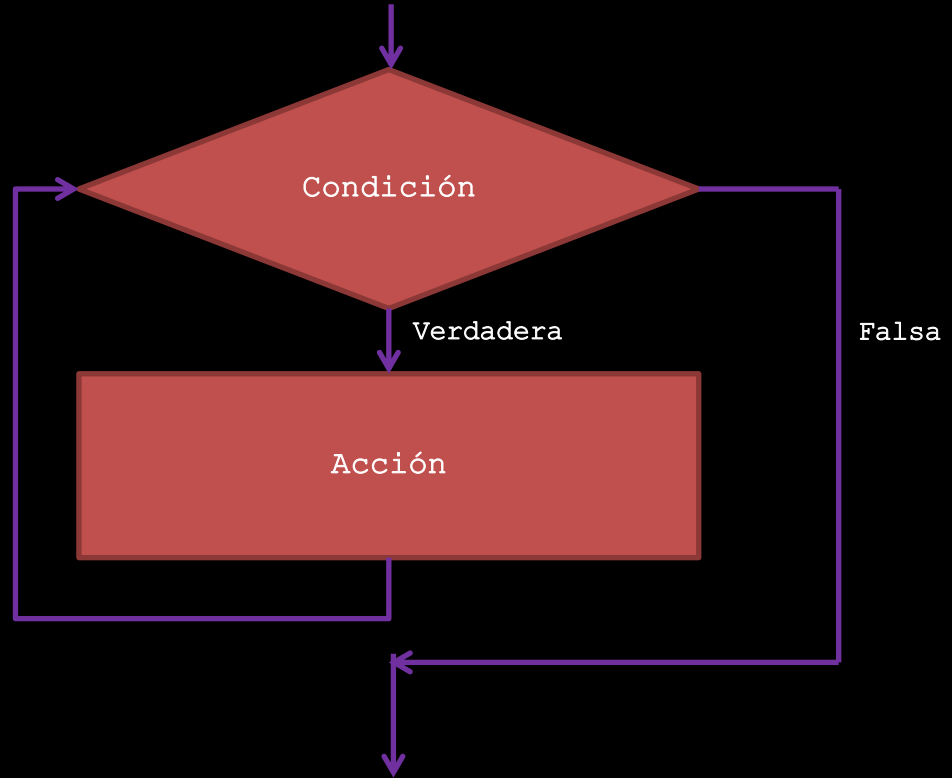
Las instrucciones que se repiten dentro del bucle se denominan cuerpo del bucle y cada repetición del bucle se llama iteración.

Sentencia while;

Sintaxis:

```
while (condición)  
    sentencia;
```

```
While (condición)  
{  
    sentencia1;  
    sentencia2;  
    .  
    .  
    .  
    sentenciaN;  
}
```



Programémosla...

Calcular el valor de la suma

$$1+2+3+4+5+\dots+100$$

Algoritmo

Se utiliza una variable **Contador** como un contador que genere los sucesivos números enteros, y **Suma** almacenar las sumas parciales 1, 1+2, 1+2+3...

1. Establecer Contador a 1
2. Establecer Suma a 0
3. **mientras** Contador <= 100 **hacer**
 Sumar Contador a Suma
 Incrementar Contador en 1
 fin_mientras
4. Visualizar Suma

Sentencia for;

Inicializa la variable de control del bucle.

Expresión lógica que determina si han de ejecutar mientras sea verdadera.

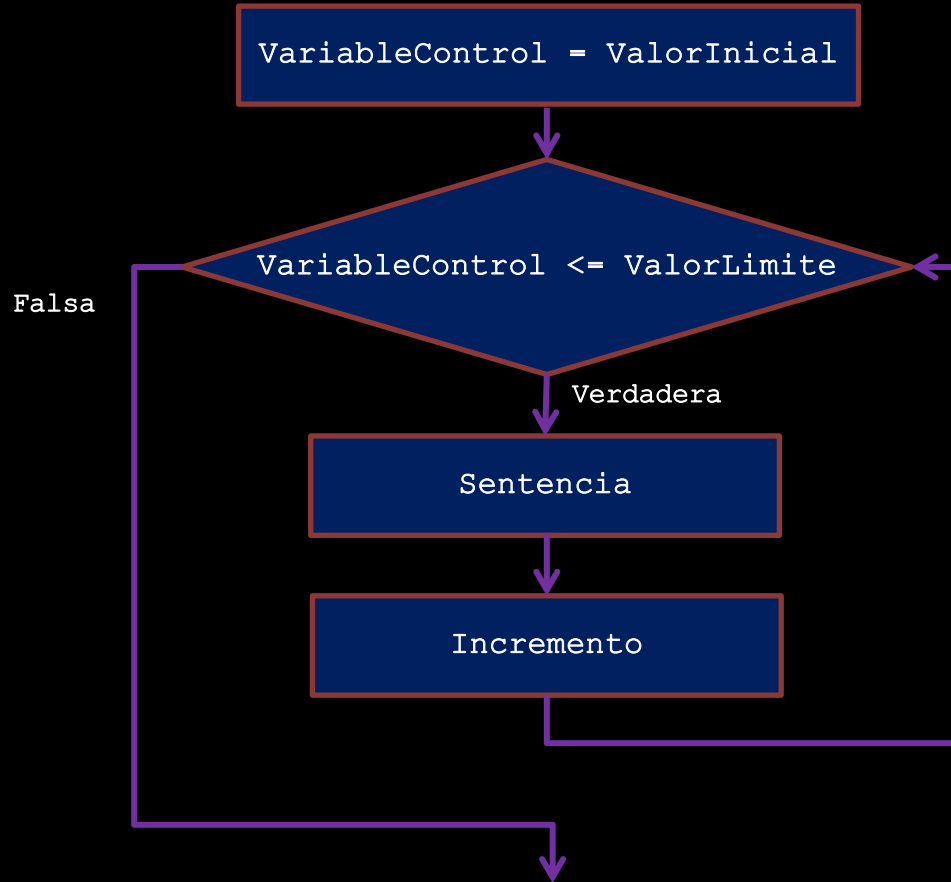
Sintaxis:

```
for (Iniciación; CondiciónIteracion; Incremento)
{
    sentencias;
}
```

Sentencias a ejecutar en cada iteración del bucle.

Incrementa o decrementa la variable de control del bucle.

Sentencia for;



Programémosla...