

Algoritmos y Estructuras de Datos

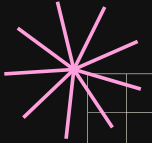
Profesor Yisheng León





¿Qué es un lenguaje de programación?

Un lenguaje de programación es un conjunto de reglas gramaticales (tanto sintácticas como semánticas) que instruyen a que un ordenador o dispositivo se comporte de una cierta manera. Cada lenguaje de programación tiene un vocabulario, un conjunto único de palabras clave que sigue a una sintaxis especial para formar y organizar instrucciones del ordenador



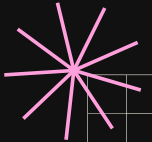


Tipos de lenguaje

Los lenguajes de bajo nivel incluyen lenguajes ensambladores y de máquina.

Un lenguaje ensamblador contiene una lista con instrucciones básicas y es mucho más difícil de leer que un lenguaje de alto nivel. Está solo un nivel por encima del lenguaje de máquina en cuanto a abstracción, usa códigos simples que se convierten fácilmente en cadenas de 1s y 0s (representación binaria). No se puede usar para estructurar y manipular información compleja.

El lenguaje de máquina se entiende directamente por la unidad de procesamiento del ordenador. Un programador escribirá primero su código en un lenguaje de alto nivel, luego lo compilará en un formato legible por máquina donde las instrucciones se representan en binario.





Tipos de lenguaje

Por otro lado, los lenguajes de alto nivel están diseñados para ser fáciles de leer y entender, permiten así a los programadores escribir el código fuente al usar palabras y símbolos lógicos y significativos. Encapsulan todo, desde los primeros lenguajes algorítmicos como FORTRAN hasta lenguajes más extendidos y orientados a objetos como C++, C# y Java.





Tipos de paradigmas de programación

1. Paradigma imperativo

El paradigma imperativo o de procedimientos es, probablemente, uno de los paradigmas más conocidos en el mundo de la programación. Este es un método que permite desarrollar programas a través de procedimientos. Mediante una serie de instrucciones, se explica paso por paso cómo funciona el código para que el proceso sea lo más claro posible.

2. Paradigma funcional

Una de las características del paradigma funcional es que este, como su nombre lo indica, trabaja a través de determinadas funciones matemáticas. Este es un tipo de paradigma que se usa, principalmente, en el ámbito académico más que en el comercial. A diferencia del paradigma imperativo, aquí importa más el “qué” y no tanto el “cómo” se desarrolla un proyecto.





Tipos de paradigmas de programación

3. Paradigma declarativo

El paradigma declarativo es aquel que se preocupa por el resultado final desde el inicio. Determinar de forma automática la ruta a seguir para conseguir una solución puede resultar muy eficaz a la hora de programar, solo se necesita tener claridad en torno al proceso que se va a llevar adelante.

4. Paradigma reactivo

El paradigma reactivo está enfocado en analizar el flujo de datos, ya sean finitos o infinitos, con el fin de responder a las necesidades que se presenten durante el desarrollo de los proyectos en términos de escalado, y para procurar una reacción inmediata al cambio de valores que se producen por los flujos de datos.





Tipos de paradigmas de programación

5. Paradigmas de programación orientada a objetos

Este tipo de paradigma de programación ofrece una guía que permite identificar cómo trabajar con él a través de objetos y planos de código. Este tipo de paradigma se constituye por piezas simples u objetos que al relacionarse entre sí forman diferentes componentes del sistema que estamos trabajando.

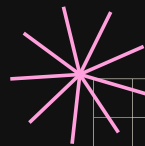




Lenguaje C

C: creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

C++: creado a mediados de 80 por Bjarne Stroustrup. Su intención fue tener mecanismos para la manipulación de objetos.





















Usos



- El kernel, corazón del sistema operativo Linux, está escrito totalmente en lenguaje C.
- La WINAPI, interfaz para el manejo de los recursos del sistema para las aplicaciones de Windows, está escrita en C.
- La mayor porción del código de OpenOffice es código C.
- La mayoría de los desarrollos de bajo nivel de hoy en día están hechos usando lenguaje
- C junto con algunas rutinas en assembler, lo que permite portar funcionalidades de un hardware a otro sin tener que reescribir nuevamente los programas.
- Grandes proyectos desarrollados en C:
<http://eenube.com/index.php/ldp/c/33-cactual>
- Índice TIOBE de lenguajes más usados:
 - <https://www.tiobe.com/tiobe-index/>

CSS

Whoa!

Mar 2024	Mar 2023	Change	Programming Language	Ratings	Change
1	1	<div>Difference compared to last year</div>	 Python	15.63%	+0.80%
2	2		 C	11.17%	-3.56%
3	4		 C++	10.70%	-2.59%
4	3		 Java	8.95%	-4.61%
5	5		 C#	7.54%	+0.37%
6	7		 JavaScript	3.38%	+1.21%
7	8		 SQL	1.92%	-0.04%
8	10		 Go	1.56%	+0.32%
9	14		 Scratch	1.46%	+0.45%
10	6		 Visual Basic	1.42%	-3.33%
11	11		 Assembly language	1.39%	+0.28%



Un programa informático o programa de computador es una pieza de software, es decir, una secuencia compleja de instrucciones y procesos orquestados para cumplir una tarea específica en un computador o sistema de computadores.

Por lo general, los programas de computador disponen de cierto margen de recursos del sistema informático mientras se ejecutan, y cumplen roles de todo tipo en el mismo, desde controlar los recursos y las operaciones internas del computador, hasta mediar con el usuario y permitirle trabajar, recrearse, explorar Internet, etc




CSS

Compiladores

Los compiladores se utilizan para crear programas que son más eficientes y rápidos que los programas escritos en lenguaje de máquina directamente.

Además, permiten a los programadores escribir programas en lenguajes de alto nivel más legibles y mantenibles, en lugar de tener que trabajar con el código de máquina directamente. Se constituye como una herramienta fundamental en la programación moderna y se utiliza en una amplia gama de aplicaciones, desde sistemas operativos y bases de datos hasta aplicaciones web y móviles.

Es responsable de transformar el código fuente de un programa en un código objeto ejecutable por la computadora, realizando una serie de análisis, optimizaciones y generaciones de código intermedio y objeto a lo largo del proceso.





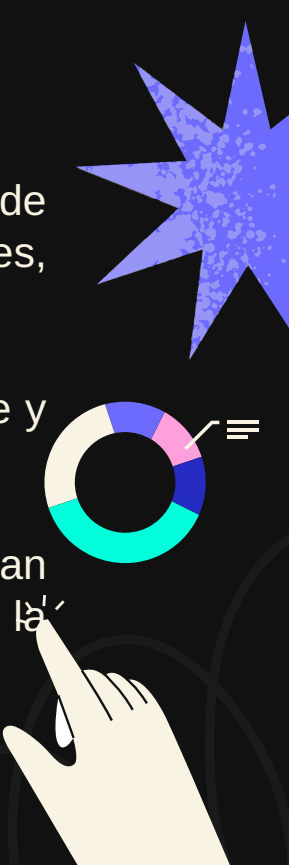
CSS

Funciones de un compilador

Análisis léxico: el compilador divide el código fuente en una serie de tokens o símbolos que representan las palabras clave, identificadores, operadores y otros elementos del lenguaje de programación.

Análisis sintáctico: el compilador verifica la estructura del código fuente y su conformidad con la gramática del lenguaje de programación.

Análisis semántico: este verifica que las instrucciones del programa sean semánticamente correctas y coherentes, y realiza verificaciones como la asignación de tipos y la resolución de nombres.





CSS

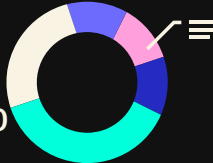
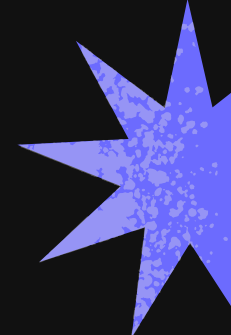

Funciones de un compilador

Generación de código intermedio: este representa el programa de manera más abstracta y se puede optimizar antes de generar el código objeto.

Optimización de código: puede realizar una serie de optimizaciones en el código intermedio o en el código objeto para mejorar su eficiencia, como la eliminación de código redundante o la reorganización de las instrucciones para minimizar la cantidad de ciclos de procesador requeridos.

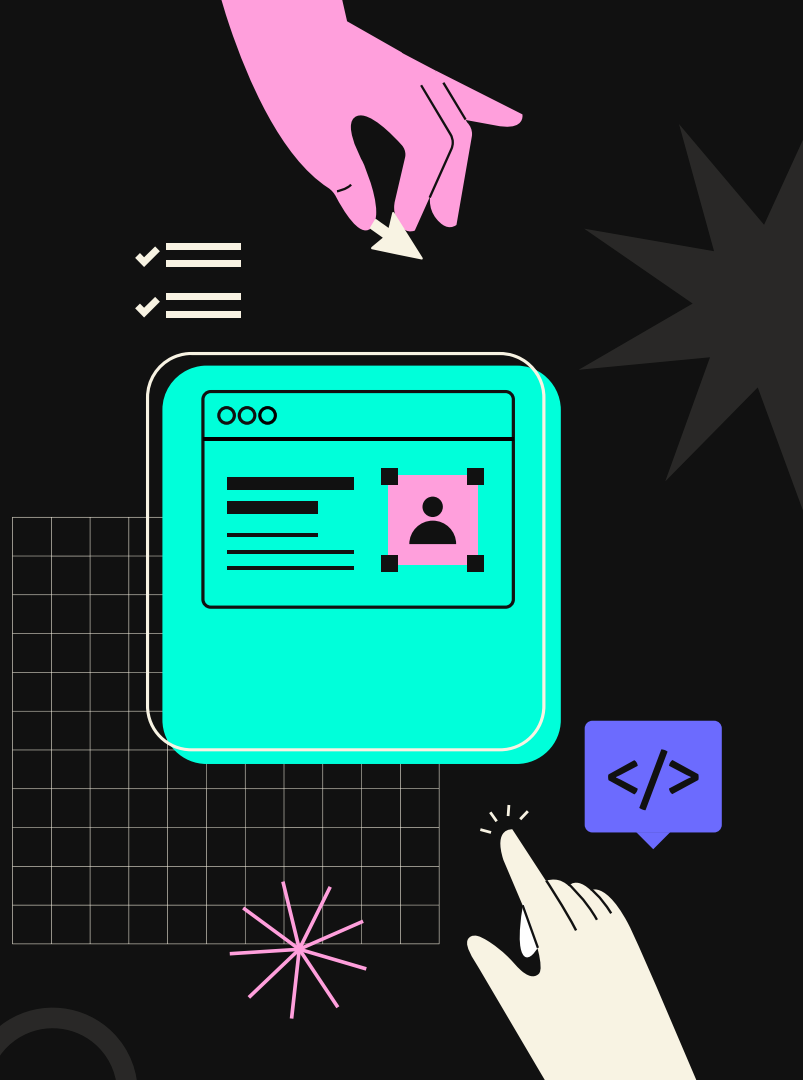
Generación de código objeto: el compilador finalmente genera el código objeto que se puede ejecutar en la computadora.

Vinculación y carga: si el programa utiliza bibliotecas externas, el compilador puede vincularlas al código objeto generado y generar un archivo ejecutable que se carga en la memoria de la computadora para su ejecución.



Intérpretes

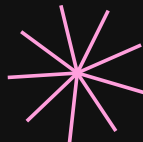
El intérprete es un programa que lee y ejecuta el código fuente línea por línea



Diferencias entre intérprete y compilador



- La traducción del código se realiza línea por línea a medida que se ejecuta el programa, lo que significa que el programa se puede ejecutar directamente sin necesidad de una etapa de compilación previa.
- El intérprete lee y ejecuta el código línea por línea, es más lento que el compilador en términos de velocidad de ejecución
- El Intérprete es más flexible y permite una depuración más fácil
- Traduce todo el código fuente a un lenguaje de máquina antes de su ejecución, lo que significa que el programa se debe compilar antes de su ejecución.
- Genera un código objeto optimizado que se puede ejecutar de manera más eficiente, lo que resulta en una velocidad de ejecución más rápida
- El compilador es más rápido y produce un código ejecutable más optimizado



Estructura de un programa informático

Cabecera

A modo de comentarios se suele especificar:

- Nombre del programa
- Datos de entrada
- Datos de salida

Funciones

Definición de funciones propias creadas por el programador para usarlas en varias ocasiones

Declaraciones

Definiciones y tipos de datos:

- variables
- constantes
- nuevos tipos de datos

Asignaciones

Valores iniciales de los datos declarados previamente

Entradas

Instrucciones para almacenar en memoria los valores de algunos datos iniciales

Control

Instrucciones de control de flujo del programa. Pueden ser:

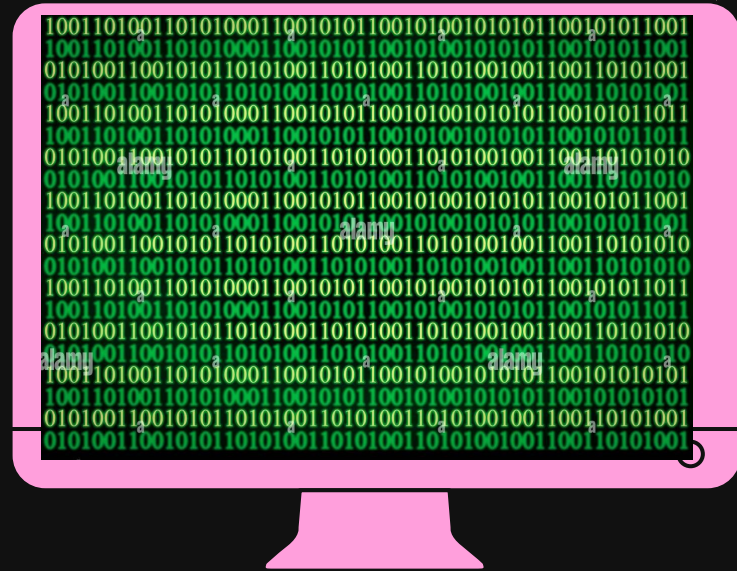
- Secuenciales
- De selección
- Iterativas

Salidas

Instrucciones para devolver los resultados obtenidos

Tipos de Datos

es la clasificación de un dato según sus características. Puede tratarse de una cadena, de un texto, de números, etc., y un valor es cualquier clase de dato que se halle dentro de un tipo de dato.



Tipos de Datos



Numéricos

incluyen números positivos, negativos; cifras decimales, naturales, etc.



Texto

letras, caracteres, símbolos que representan otros idiomas.



Valores booleanos

que son fundamentales para establecer condiciones de verdad o falsedad.



Listas

para almacenar múltiples elementos de un mismo tipo

El código ASCII

sigla en inglés de American Standard Code for Information Interchange
(Código Estadounidense Estándar para el Intercambio de Información)

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	_			

ASCII extendido

DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
128	80h	Ç	160	A0h	à	192	C0h	Ł	224	E0h	Ò
129	81h	ü	161	A1h	á	193	C1h	ł	225	E1h	ó
130	82h	ê	162	A2h	â	194	C2h	Ł	226	E2h	ô
131	83h	ë	163	A3h	ã	195	C3h	ł	227	E3h	õ
132	84h	ä	164	A4h	ä	196	C4h	Ł	228	E4h	ö
133	85h	å	165	A5h	å	197	C5h	Ł	229	E5h	ÿ
134	86h	ä	166	A6h	ä	198	C6h	Ł	230	E6h	µ
135	87h	ç	167	A7h	ç	199	C7h	Ł	231	E7h	þ
136	88h	è	168	A8h	è	200	C8h	Ł	232	E8h	ð
137	89h	é	169	A9h	é	201	C9h	Ł	233	E9h	ù
138	8Ah	ê	170	AAh	ê	202	CAh	Ł	234	EAh	ú
139	8Bh	ÿ	171	ABh	ÿ	203	CBh	Ł	235	EBh	û
140	8Ch	ï	172	ACH	¼	204	Ch	Ł	236	ECn	ÿ
141	8Dh	í	173	ADh	½	205	CDh	Ł	237	EDh	ÿ
142	8Eh	À	174	Aeh	¾	206	CEh	Ł	238	EEh	ÿ
143	8Fh	Á	175	Afh	¸	207	CFh	Ł	239	EFh	ÿ
144	90h	Ê	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	ó	179	B3h	¸	211	D3h	Ł	243	F3h	½
148	94h	ô	180	B4h	¸	212	D4h	Ł	244	F4h	¾
149	95h	õ	181	B5h	¸	213	D5h	Ł	245	F5h	¸
150	96h	ù	182	B6h	¸	214	D6h	Ł	246	F6h	¸
151	97h	ú	183	B7h	¸	215	D7h	Ł	247	F7h	¸
152	98h	ý	184	B8h	¸	216	D8h	Ł	248	F8h	¸
153	99h	ÿ	185	B9h	¸	217	D9h	Ł	249	F9h	¸
154	9Ah	U	186	BAh	¸	218	DAh	Ł	250	FAh	¸
155	9Bh	ø	187	BBh	¸	219	DBh	Ł	251	FBh	¸
156	9Ch	£	188	BCh	¸	220	DCn	Ł	252	FCn	¸
157	9Dh	ø	189	BDh	¸	221	DDh	Ł	253	FDh	¸
158	9Eh	x	190	BEh	¸	222	DEh	Ł	254	FEh	¸
159	9Fh	f	191	Bfh	¸	223	DFh	Ł	255	FFh	¸



CSS

Ejemplos

Character "A": 0100 0001

Character "C": 0100 0011

Character "I": 0010 0001

Character "#": 0010 0011


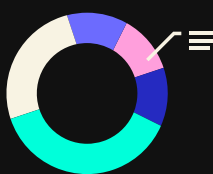
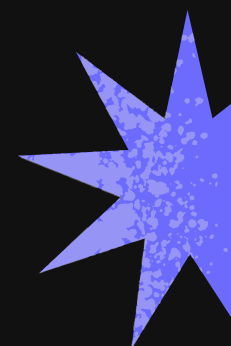
Character "/": 0010 1111

Character "K": 0100 1011

Character "k": 0110 1011

Character "X": 0101 1000

Character "x": 0111 1000



Comentarios

C#

Python

```
1 string filePath = "ruta_del_archivo.txt"; // nombre del fichero
2 string fileContent = File.ReadAllText(filePath); // leer fichero
3 Console.WriteLine("El contenido del archivo es:\n" + fileContent); // mostrar fichero
```

C, C++, C#, Java, JS, TS

Python

SQL, VB

En los lenguajes derivados de **C** se emplea `/*` y `*/`

```
1 /*
2 Este es un comentario
3 de múltiples líneas en
4 Se utiliza para proporcionar
5 explicaciones más extensas
6 */
7 int miVariable = 25;
```

Operadores

```
1 int a = 12;
2 int b = 5;
3 int suma = a + b;           // suma = 17
4 int resta = a - b;          // resta = 7
5 int multiplicacion = a * b;  // multiplicacion = 60
6 int division = a / b;       // division = 2
7 int modulo = a % b;         // modulo = 2
```

```
1 string texto = "Hola, mundo!";
2 int longitud = texto.Length; // Accediendo a la propiedad Length de texto cadena
```

```
1 int[] numeros = { 1, 2, 3, 4, 5 };
2 int segundoNumero = numeros[1]; // Accediendo al segundo elemento del array
```

Los operadores son símbolos o palabras clave que se utilizan para manipular y combinar expresiones. Permiten realizar diferentes acciones, como realizar operaciones aritméticas, comparar valores o asignar valores a variables.

En esencia, los operadores son las herramientas fundamentales para formar con expresiones en la programación

Operadores

```
1 // operador asignacion
2 int x = 5;
3
4 // operadores compuestos
5 x += 3; // Equivalente a x = x + 3
6 x -= 2; // Equivalente a x = x - 2
7 x *= 4; // Equivalente a x = x * 4
8 x /= 2; // Equivalente a x = x / 2
9 x %= 7; // Equivalente a x = x % 7
```

```
1 int a = 5;
2 int b = 10;
3
4 bool igual = (a == b); // igual = false
5 bool distinto = (a != b); // distinto = true
6 bool mayorQue = (a > b); // mayorQue = false
7 bool menorQue = (a < b); // menorQue = true
8 bool mayorOIgual = (a >= b); // mayorOIgual = false
9 bool menorOIgual = (a <= b); // menorOIgual = true
```

```
1 bool or_a_b = a || b;
2 bool and_a_b = a && b;
3 bool not_a_b = !a;
```

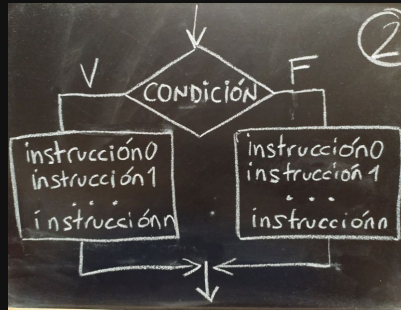
A	B	A AND B	A OR B
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Tipos de Estructuras



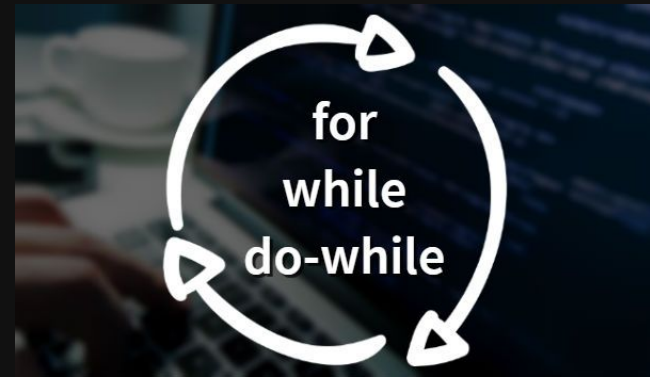
Selectivas

Las Estructuras Selectivas también conocidas como Estructuras Condicionales, es una estructura de control de flujo que permite que desarrollemos lo que se conoce como lógica de programación. Por decir de otra manera son aquellas que dirigen la ejecución de un programa hacia un grupo de sentencias del resultado de la condición.



Iterativas

Las estructuras de control iterativas se utilizan para resolver problemas donde sea necesario repetir un determinado número de veces un conjunto de instrucciones.



Funciones

es una sección de un programa que calcula un valor de manera independiente al resto del programa



Parámetros

son los valores que recibe la función como entrada



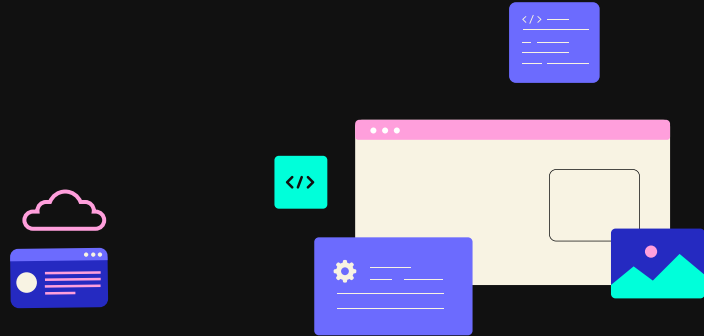
Código de la función

que son las operaciones que hace la función



Resultado

el valor final que entrega la función



Hasta la próxima clase!

