Algoritmo

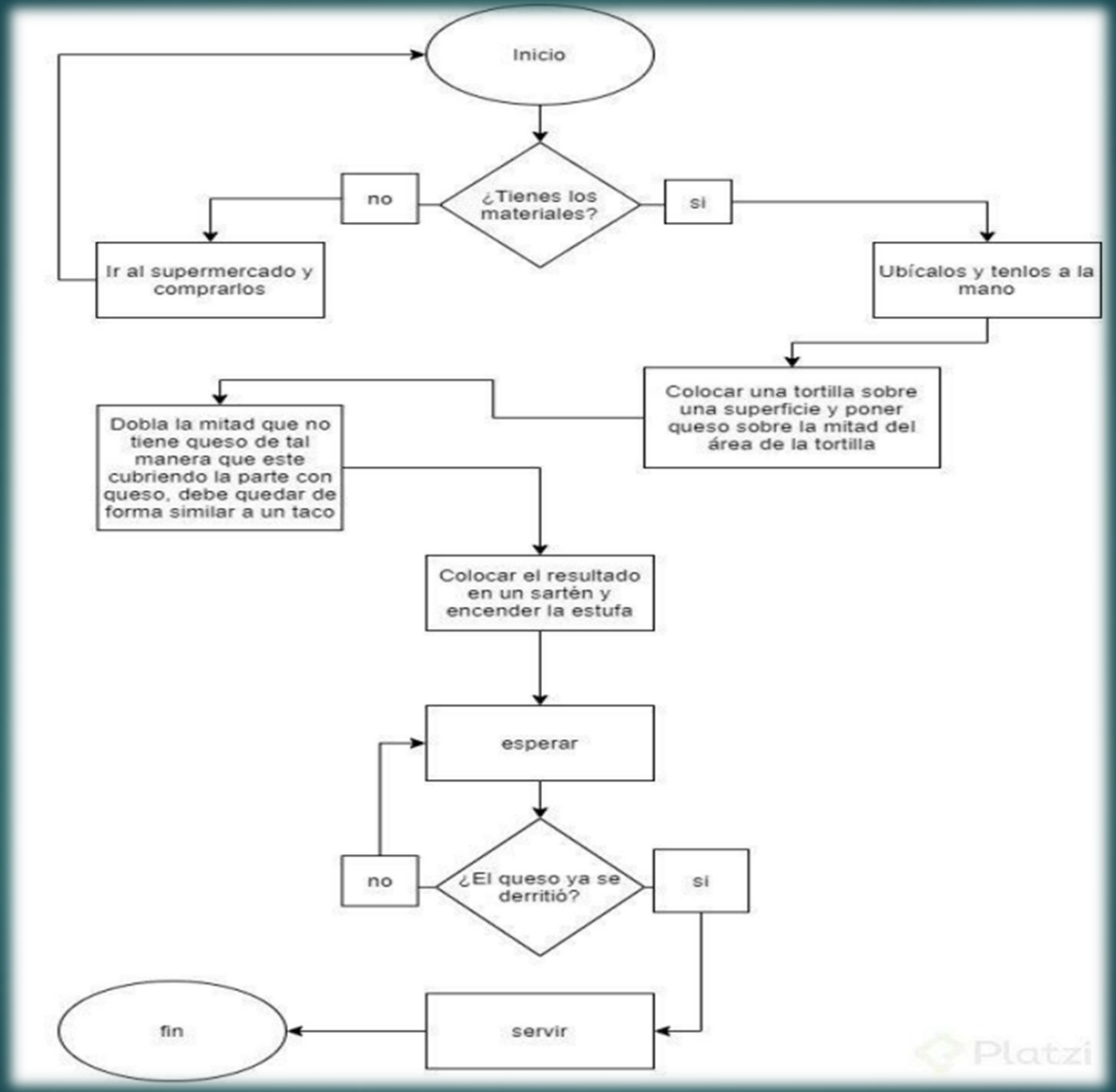
Un algoritmo consiste en una serie de instrucciones detalladas finitas y escritas en un lenguaje cotidiano sobre cómo resolver un problema o ejecutar una acción por medio del razonamiento lógico, su uso amplía el panorama sobre los elementos que son necesarios para resolver un problema y permite evaluar de forma permanente si el razonamiento utilizado para encontrar una solución es el adecuado o hay que cambiar de enfoque.

Los algoritmos pueden clasificarse en cualitativos y cuantitativos. Los cualitativos se utilizan para listar los pasos a seguir en actividades diarias cómo preparar una bebida o peinarse mientras que los cuantitativos implican cálculos numéricos, por ejemplo, calcular el finiquito de un empleado u obtener la raíz cuadrada de un número.

Varias de las acciones de tu vida cotidiana pueden verse como algoritmos ya que para completar ciertas tareas, realizas las mismas acciones una y otra vez. Por ejemplo, cuando lavas tus dientes. Para redactar un algoritmo define las entradas de tu proceso, es decir, los elementos que son necesarios para realizar una acción. En el caso de lavarse los dientes, los insumos que requiere son la crema dental, el cepillo y un vaso con agua. Después determina las acciones que llevas a cabo, redáctalas de forma sencilla y numera cada instrucción. Por ejemplo, tomar la pasta dental, abrir el producto, colocar la tapa de la pasta en algún lugar, tomar el cepillo de dientes, exprimir cierta cantidad de pasta en el cepillo, dejar la pasta sobre algún lugar, abrir la boca, colocar el cepillo en las encías, cepillar los dientes, cepillar la lengua suavemente, tomar el vaso con agua y enjuagarse la boca, repetir el procedimiento hasta que la boca esté libre de residuos. Determina qué resultados se pretende llegar. Para el algoritmo del ejemplo, el objetivo es obtener dientes sin residuos de comida.

La resolución de problemas a través de algoritmos es la forma más efectiva para desarrollar un pensamiento estructurado y lógico.

Algoritmo


Inicio
¿Tienes los materiales?
no
si
Ir al supermercado y comprarlos
Ubícalos y tenlos a la mano
Colocar una tortilla sobre una superficie y poner queso sobre la mitad del área de la tortilla
Dobla la mitad que no tiene queso de tal manera que este cubriendo la parte con queso, debe quedar de forma similar a un taco
Colocar el resultado en un sartén y encender la estufa
esperar
¿El queso ya se derritió?
no
si
servir
fin
Platzi

Algoritmo

El siguiente ejemplo describe un algoritmo cuantitativo para calcular el valor de la hipotenusa de un triángulo rectángulo cuyo valor de un cateto es 3 y del otro 4.

Identifica el valor del primer cateto: 3
Identifica el valor del segundo cateto: 4
Eleva el primer cateto al cuadrado: 3 x 3 = 9
Eleva el segundo cateto al cuadrado: 4 x 4 = 16

Suma el valor de cada uno de los catetos al cuadrado: 9 + 16 = 25
Toma la raíz cuadrada del la suma de los cuadrados de los catetos: $\sqrt{25}$ =5
Da el valor de la hipotenusa: $\sqrt{25}$ = 5

Algoritmo

Reto. Diseña un algoritmo sobre alguna actividad de tu vida cotidiana, como preparar una receta de cocina.

Reto. Diseña un algoritmo para hallar el ángulo interior de un triángulo dado los ángulos de los otros dos ángulos interiores del mismo triángulo.

Reto. Desarrolla un algoritmo que calcule la edad de una persona con base en la obtención de su fecha de nacimiento.

Operadores

Los operadores aritméticos son los que te permiten realizar operaciones matemáticas como suma, resta, multiplicación, división, potencias, entre otras, y se utilizan en algoritmos cuantitativos para encontrar la solución a un problema.
Al momento de realizar estas operaciones es frecuente agruparlas en uno o más paréntesis para facilitar su cálculo. Además, por medio de este símbolo se indica que las operaciones deben realizarse de izquierda a derecha y respetando la prioridad de ejecución de los operadores que es la siguiente: primero se realiza el cálculo de potencias, después la multiplicación y división, luego se obtiene el módulo o residuo de una división, y por último se calcula la suma y resta.

Operadores

Nivel jerárquico	operador
0	() paréntesis
1	^ potencias
2	* Multiplicación, / división
3	**mod** Módulo o residuo
4	+ Suma, – Resta

Ejemplo:

X = (3^2 + 10/2) + (3*9 mod 4 – 1)

En la expresión anterior primero se agrupan los términos dentro de los paréntesis de izquierda a derecha

(3^2 + 10/2) primero se realiza lo que hay dentro en este paréntesis

(3*9 mod 4 – 1) después se realiza lo que hay dentro de este otro paréntesis

(3^2 + 10/2) dentro de este paréntesis la potencia 3^2 se realiza primero cuyo resultado es 9. Después se realiza la división 10/2 cuyo resultado es 5. Por último se realiza la suma de las dos operaciones previas que es la suma de 9 + 5 cuyo resultado es 14.

(3*9 mod 4 – 1) dentro de este paréntesis primero se hace la multiplicación 3*9 cuyo resultado es 27, Después se realiza 27 mod 4 cuyo resultado es 3. Y por último se realiza la resta 3 – 1 cuyo resultado es 2.

Por último, se realiza la suma de los resultados de las operaciones dentro de los dos paréntesis.

(3^2 + 10/2) + (3*9 mod 4 – 1) = 14 + 2 = 16.

Operadores

Reto. Describe el proceso en que se computa las siguientes fórmulas.

X = (18/9 mod 2 + 16) – (5 * 4 – 3^3)

X = (5 + 2 * 4) – 25 mod 5

X = (100/5^2 + 1) + 11 * 3

X = ((8-6)^2*3 mod 3)^3.

Operadores

Los ***operadores relacionales*** se utilizan para comparar dos o más valores y determinar si el resultado es falso o verdadero:

Operador	Operación
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
<> ó !=	Diferente a
=	Igual a

Ejemplo. Podríamos diseñar un algoritmo que tomara los datos personales de un alumno de una escuela y su desempeño académico para almacenarlos en una base de datos.

Operadores

Ejemplo. Podríamos diseñar un algoritmo que tomara los datos personales de un alumno
de una escuela y su desempeño académico para almacenarlos en una base de datos.

Ya almacenados los datos podríamos preguntar por la situación actual de un alumno para saber si amerita pasar al siguiente nivel. Por ejemplo, podríamos pedir el promedio general de las calificaciones de todas las materias que curso en el ciclo anterior. Si el promedio general es menor a 6 se enviaría un mensaje al alumno de alerta donde se le haría saber que debe tomar una vez más el ciclo anterior inmediato. Si el promedio general es mayor o igual a 9 podríamos enviar un mensaje de alerta que indique al alumno que debe pasar por un reconocimiento por parte de la institución.

Nombre de la variable	Operador	Valor de comparación	Mensaje de alerta que se envía
Promedio del alumno	<	6	Repite ciclo escolar
Promedio del alumno	>=	9	Reconocimiento por buen desempeño

Operadores

Reto. Una empresa de logística ofrece a sus trabajadores un bono de puntualidad. Si el empleado siempre llega puntual durante el mes se le notifica que ha sido acreedor al bono de puntualidad. Si el empleado tiene 2 retardos se le suspende un día y si tiene 3 o más retardos se le da de baja al final del mes. Describe tal situación usando operadores relacionales.

Reto. Analiza el problema y subraya la opción que consideres que lo resuelve:

"C" es mayor que "D". "E" es menor que "F". "G" es menor que "E" y "D" es mayor que "F". ¿Cuál es el menor de todos?

a) G

b) C

c) E

d) D

e) F

Operadores

Los operadores lógicos se utilizan para evaluar dos o más expresiones que utilizan operadores relacionales para determinar si la expresión en general es verdadera o falsa.

Los operadores lógicos son: conjunción o AND que se representa con un doble ampersand (&&)
y disyunción u OR que se representa con dos barras (| |) verticales. El operador AND determina el valor booleano de cada expresión, es decir, si ambas son verdaderas el resultado general de la expresión también lo es, sin embargo, si una de ellas es falsa el valor final también lo será. El operador OR evalúa cada expresión, si uno de los términos es verdadero en automático el valor general de la expresión también lo es. La única forma en que la expresión pueda ser tomada como falsa es cuando el valor bolea no de ambos elementos también es falso. El operador negación o NOT que se representa con el símbolo ¬ , cambia el valor final de una expresión por su contrario, es decir, si tras evaluar una expresión en general se determina que es verdadera y se le aplica el operador NOT, entonces el resultado final será falso.

Operadores

Reto. Evalúa las siguientes expresiones a la derecha.

1) (45<120 OR 12<120) =
2) (6! =6) && (12>22) =
3) ¬ (128<145 && 12>9) =
4) “Daniela”< >”DANIELA” =
5) 10*20< >210 =

Conceptos Básicos de Programación

Cuando realizas una petición a tu equipo de cómputo, (por ejemplo, guardar un documento en el procesador de textos), internamente se ejecutan bloques de instrucciones escritos en un lenguaje de programación que hacen posible las demandas del usuario.
Un lenguaje de programación es un idioma artificial creado para indicarle a la computadora lo que debe hacer. Tiene ciertas reglas de escritura (sintaxis) en las que utiliza símbolos y palabras clave, además de una semántica (interpretación interna).

Sintaxis	Semántica
main()	Indica el punto de entrada de un programa. Instrucción válida.

Conceptos Básicos de Programación

Un programa es un bloque de instrucciones (código fuente) escritas en cierto lenguaje de programación cuyo propósito es resolver un problema.

Abrir GDB

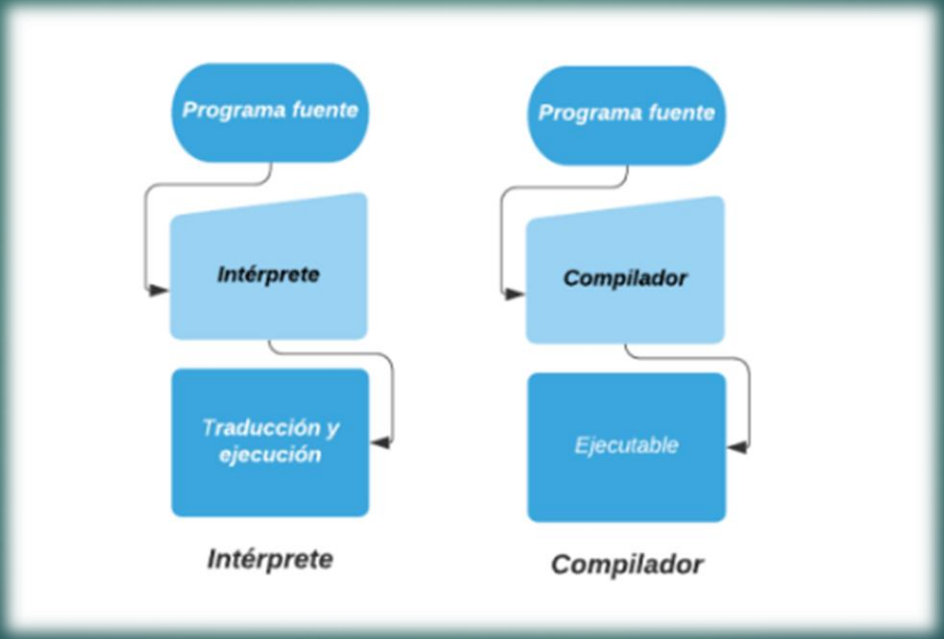

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main() {
7    float c1,c2,hipo=0;
8
9    cout << "Digite cateto 1: ";
10   cin >> c1;
11   cout << "Digite cateto 2: ";
12   cin >> c2;
13
14   hipo = sqrt(pow(c1,2)+pow(c2,2));
15
16   cout << "La hipotenusa es " << hipo << endl;
17
18   return 0;
19 }
```

Conceptos Básicos de Programación

Para ***ejecutar un programa*** la computadora realiza una traducción de sus componentes al lenguaje máquina, es decir, convierte las instrucciones en cadenas de ceros y unos. Este proceso de conversión puede hacerse de dos formas:

Por medio de un programa "*intérprete*" que traduce y ejecuta instrucción por instrucción.

A través de un programa "*compilador*" que toma al bloque de instrucciones lo traduce sólo una vez y lo ejecuta.



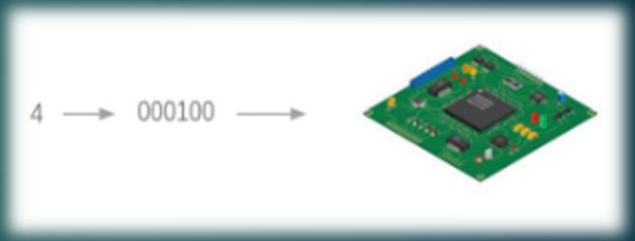
Tipos de Datos

Antes de decir lo que son los datos primitivos, es necesario entender lo qué es un dato.

Dato: es un hecho que se representa en la computadora. Ejemplo: el color rojo es un hecho y dependiendo del contexto puede representar algo.

En el contexto de las reglas de tránsito el color rojo representa alto, y en el contexto de los símbolos patrios este color está asociado con la sangre de los héroes que nos dieron patria.

La computadora sólo puede representar números en su memoria dentro de sus circuitos, y para el dato color rojo puede asignarle un número entero por ejemplo el número 4.



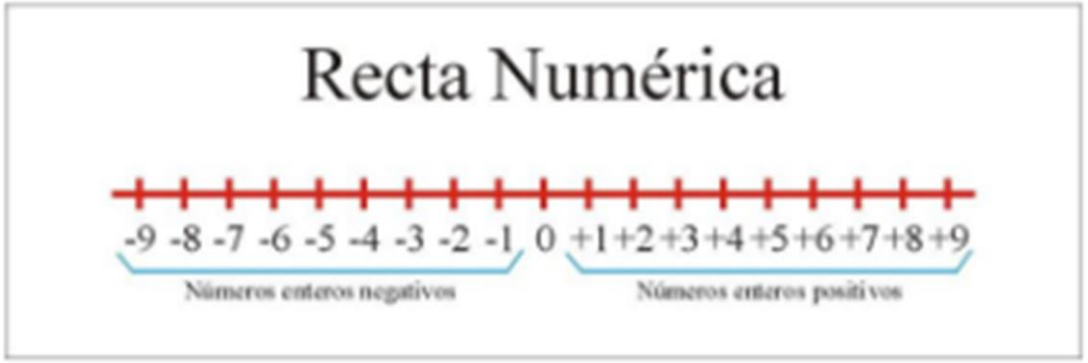
Tipos de Datos

Un ***dato primitivo*** es una abstracción del mundo real (como por ejemplo un número entero, carácter, un valor lógico, etc.), los cuales pueden ser representados internamente por la computadora. Son "primitivos" en el sentido que no pueden descomponerse en componentes más simples. Desde estos tipos básicos puede definir datos compuestos.

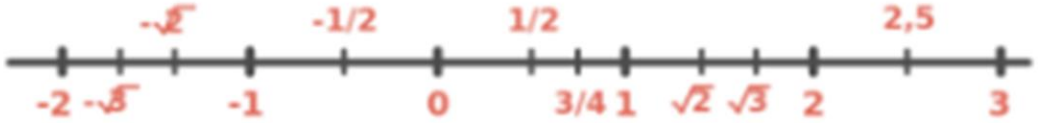
Los ***datos numéricos*** abarca tanto los datos denominados de tipo *entero* y los de tipo *punto flotante* o *reales*.

Tipos de Datos

Un dato de *tipo entero* es un subconjunto *finito* de los números enteros cuyo tamaño depende del lenguaje de programación y de la computadora utilizada.



Un dato de *tipo real* es un subconjunto finito de los números reales. Un número real consta de un entero y una parte decimal y pueden ser positivos o negativos incluyendo el cero.



Tipos de Datos

Un dato de ***tipo carácter*** es la unidad básica e indivisible de lo que llamaremos por el momento "una cadena", también suele ser llamado *símbolo*, como puede ser: a, @, #, 1, etc. Cada carácter es representado por un conjunto de *bits*. Los caracteres suelen conformar conjuntos tales como el código ASCII o el UNICODE.

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(sec. ctrl. datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(confirmación negativa)
22	SYN	(inactividad sincr.)
23	ETB	(fin bloque trans.)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

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

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	▌	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Tipos de Datos

El dato de ***tipo cadena*** o ***string***, es un tipo de dato compuesto debido a que consiste de una serie finita de (datos tipo) caracteres que se encuentran delimitados, dependiendo del lenguaje de programación, por espacios en blanco y por una comilla (') o doble comillas (").

"Hola" "Escuela de códigos"

"Esto es una cadena de caracteres"

Representación de cadenas o strings

Los datos de ***tipo lógicos o Booleanos*** (falso, verdadero) están formados por dos valores que son falso (false) y verdadero (true).

Por ejemplo, si **a** = 5 y **b** = 8, obtendríamos:

a == b: false Pregunta si **a** es idéntico a **b** : la salida es falso

a != b: true Pregunta si **a** es diferente a **b**: la salida es verdadero

a < b: true Pregunta si **a** es menor a **b** : la salida es verdadero

a > b: false Pregunta si **a** es mayor a **b** : la salida es falso

a <= b: true Pregunta si **a** es menor o igual a **b** : la salida es verdadero

a >= b: false Pregunta si **a** es mayor o igual a **b** : la salida es falso

Tipos de Datos

La siguiente tabla resume los tipos de datos:

Tipo de dato	Uso	Ejemplo
Números enteros (int)	Úsalo para almacenar valores numéricos	int edad=18
Números decimales (float)	Utilízalo para almacenar valores numéricos que tengan decimales	float gravedad=9,81
Carácter (char)	Úsalo para almacenar una letra o su equivalente en código ASCII	char respuesta='S'; char respuesta='83';
Cadena de caracteres (string)	Utilízalo para almacenar mensajes.	string Nombre= "Mi nombre es Daniela"; string Alumno= "Soy el alumno 392 en la Escuela de Códigos";

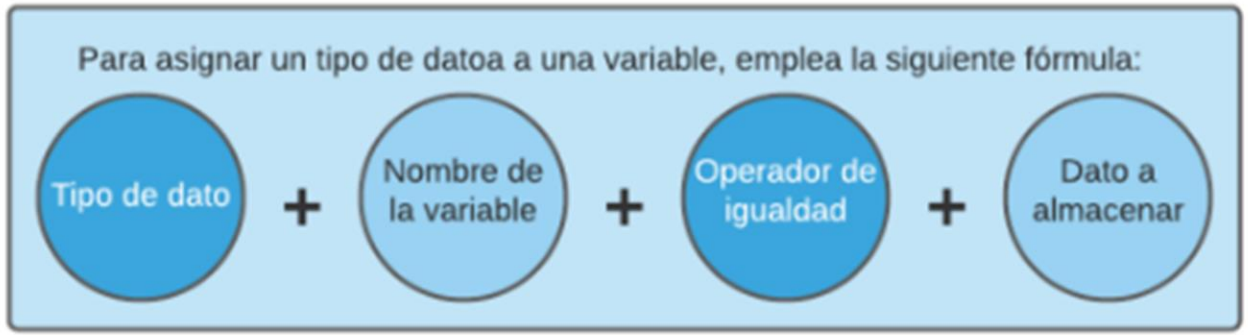
Variables

Reten por un momento el número 5 en tu memoria mental, y reten al mismo tiempo el número 2 en tu memoria mental. Lo que acabas de hacer es almacenar dos valores diferentes en tu memoria. Suma 1 al primer número que guardaste en tu memoria. Entonces, deberías retener los números 6 (es decir, 5 + 1) y 2 en tu memoria. Podrías restar estos dos valores y obtener como resultado el número 4.

Todo el proceso que acabas de hacer con tu memoria mental es un símil de lo que puede hacer una computadora con dos variables. Obviamente, este es un ejemplo muy simple ya que solo hemos usado dos valores enteros pequeños, pero considera que tu computadora puede almacenar millones de números como estos al mismo tiempo y realizar operaciones matemáticas sofisticadas con ellos.

Esto da pie a definir a una ***variable*** como una porción de memoria, que posee un nombre, que sirve para almacenar un valor de cierto tipo de dato, el cual cambia durante la ejecución del programa.

Variables



Hay dos tipos de variables. *Globales*: se escriben al comienzo del programa. *Locales*: Se declaran dentro de un bloque de instrucciones del programa.

Reto. Responde las siguientes preguntas:

¿Qué tipo de dato debe tener una variable para representar la calificación promedio de un curso?

¿Qué tipo de dato debe tener una variable para representar el número de personas en un hogar?

Variables

Clasificación de variables

Numérica
- Entera (int)
- Flotante (float)

Alfanumérica
- Caracter (char)
- Cadena (String)

Lógica
- Booleana (bool)

Comportamiento de variables (Ejemplos)

Variables de trabajo
- c= a + b;
- var = sqrt(valor);
- suma = c1+c2;

Variables contadoras
- x = x+1;
- I = i - 1;
- suma=suma+2;

Variables acumulativas
- suma = suma + valor;
- x = x + i;
- i = i + sum;

¿Qué tipo de dato debe tener una variable para contener el nombre de pila de una persona?

¿Qué tipo de dato debe tener una variable para registrar si está lloviendo o no?

¿Qué tipo de dato debe tener una variable para representar la cantidad de dinero que tienes?

Diagramas

El análisis de un problema y la construcción de los algoritmos para solucionarlo, implican un proceso lógico que puede efectuarse de forma individual o grupal. Éste sólo es el primer paso de la solución, posteriormente debe revisarse para encontrar posibles errores u omisiones en la misma, pedir el punto de vista de otras personas y finalmente llevar a cabo el algoritmo. Cada una de estas acciones necesita, generalmente, comunicar el algoritmo a otras personas.

En este caso, la comunicación oral del algoritmo es poco práctica porque se presentan problemas derivados de la diferencia de conceptos e incluso omisión de detalles. Por ello la mejor opción es utilizar herramientas que nos permiten plasmar en un lenguaje común la solución. Una de las herramientas más utilizadas para representar los algoritmos son los *diagrama de flujo*.

Los ***diagramas de flujo*** son una herramienta para la representación gráfica de un algoritmo a través de símbolos, que corresponden a cada uno de los diferentes tipos de estructuras de control (secuencia, selección e iteración).

Diagramas

Los beneficios que nos proporcionan son:

–Favorecer la comprensión e interpretación de cada uno de los pasos del algoritmo.

–Identificar los problemas y las oportunidades de mejora del algoritmo.

–Mostrar claramente las entradas y salidas esperadas.

–Facilitar la programación o ejecución del algoritmo.

Los diagramas de flujo se utilizan para describir gráficamente un algoritmo, y su simbología muestra la solución de un problema con una trayectoria de inicio a fin.

Diagramas

Sus características fundamentales son:

–El flujo de los pasos es de arriba hacia abajo y de izquierda a derecha.

–Es una secuencia de pasos:

-Inicio del proceso,

-Entrada de datos,

-Proceso a realizarse,

-Salida de los datos procesados y

-Fin del proceso.

Diagramas

–Existe siempre un camino que permite llegar a una solución.

–Existe un único inicio del proceso.

–Existe uno o más puntos de fin para el proceso de flujo.

–Solamente emplea líneas de flujo horizontal y/o vertical.

–Evita el cruce de líneas (usando los conectores)

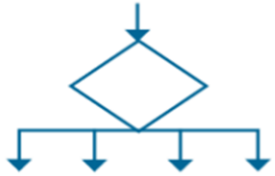
–Deben utilizarse los conectores sólo cuando sea necesario.

–No tienen líneas de flujo sin conectar.

–El lenguaje es conciso y claro

Diagramas

Estas características se representan bajo la siguiente simbología:

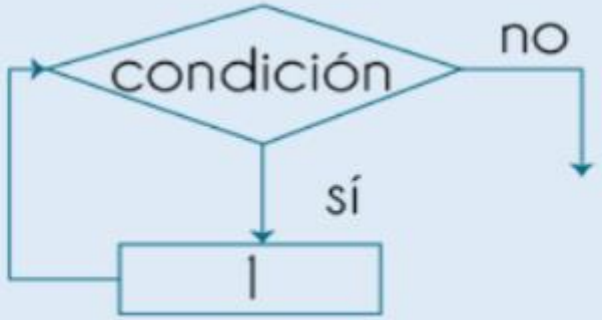
Símbolo	Significado
	Terminal /Inicio
	Entrada de datos
	Proceso
	Decisión
	Decisión múltiple
	Imprimir resultados
	Flujo de datos
	Conectores

Diagramas

Iteración (Mientras)

Símbolo del MIENTRAS dada una expresión al principio de la iteración, esta es evaluada.

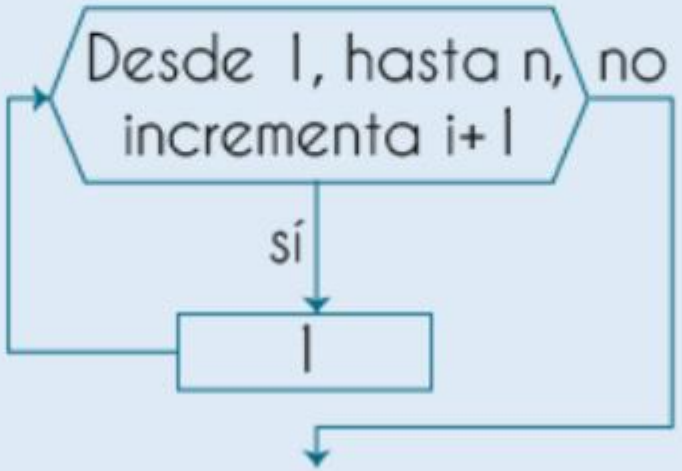
Si la condición es verdadera realizará el ciclo, si es falsa, la repetición cesará.



Diagramas

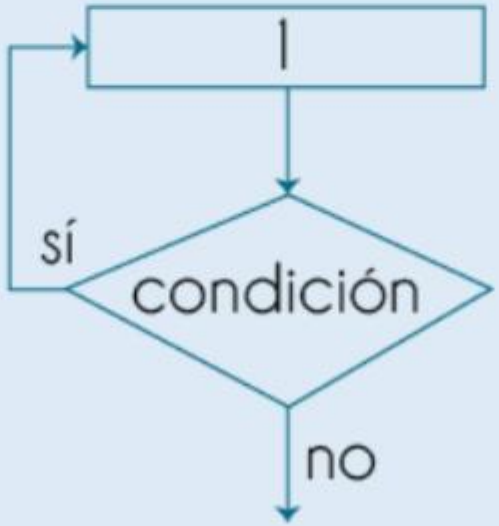
Iteración (Desde/Hasta)

Símbolo DESDE/HASTA. Esta estructura de control se utiliza cuando se conoce de antemano el número de iteraciones



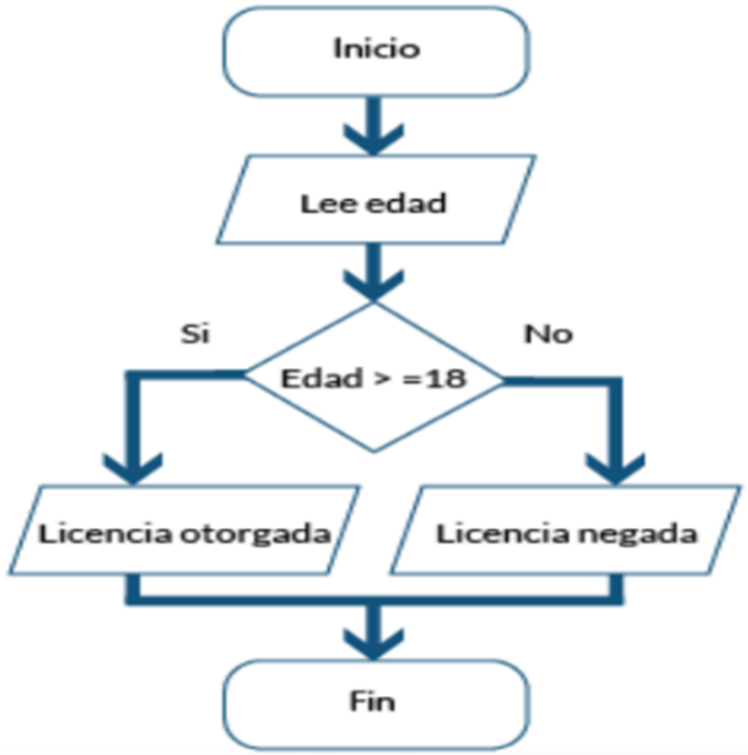
Iteración (Repite mientras)

Símbolo REPITE HASTA. Funciona igual que la estructura MIENTRAS, con la diferencia que al menos una vez hará el grupo de instrucciones y luego evaluará una condición. Si la condición evaluada es falsa continua dentro del ciclo y si es verdadera termina la iteración.



Diagramas

Ejemplo. A continuación se ilustra la solución al problema que consiste en otorgar el permiso de conducir solo a personas mayores de 18 años.



- Inicio de algoritmo
- Entrada de edad
- Decide si edad es mayor o igual a 18
- Si: Despliega “licencia otorgada”
- No: Despliega “licencia negada”
- Fin de algoritmo.

Abrir Draw IO

Diagramas

Reto. Realiza un diagrama de flujo que lee una calificación o nota académica y decide si es aprobatoria o no.

Puedes utilizar un procesador de palabra como word o bien utilizar una herramienta para realizar diagramas de flujo como DIA o Draw.io, búscalas en tu navegador e instala la herramienta si es necesario.

Estructuras de Control

Las ***estructuras de control*** se utilizan para controlar el flujo de un programa (o bloque de instrucciones), son métodos que permiten especificar el orden en el cual se ejecutarán las instrucciones en un algoritmo. Si no existieran las estructuras de control, los programas se ejecutarían linealmente desde el principio hasta el fin, sin la posibilidad de tomar decisiones.

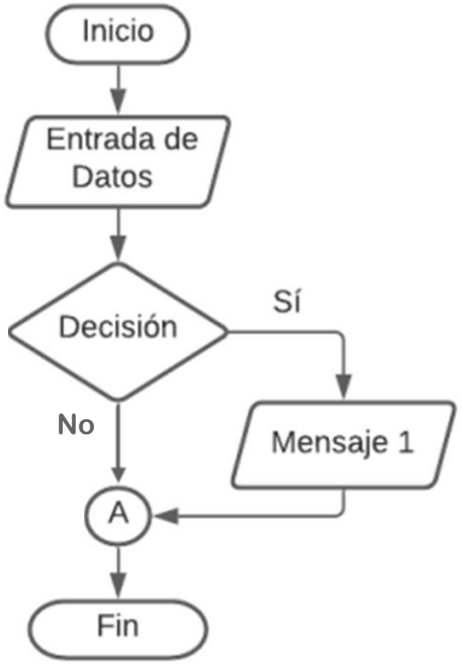
Por lo general, en la mayoría de lenguajes de programación encontraremos dos tipos de estructuras de control. Encontraremos un tipo que permite la ejecución condicional de bloques del programa y que son conocidas como *estructuras condicionales*. Por otro lado, encontraremos las *estructuras iterativas* que permiten la repetición de un bloque de instrucciones, un número determinado de veces o mientras se cumpla una condición.

La finalidad de utilizar la ***estructura condicional*** es tomar una decisión con base en el valor booleano de una expresión, es decir, determinar si la condición es verdadera o falsa.

De acuerdo a su complejidad se clasifica en:

- *Simple*. Donde la estructura ejecuta un bloque de instrucciones cuando la condición es verdadera, en caso contrario ignora ese bloque y continúa con la ejecución del resto de las instrucciones.
- *Compuesta*. Evalúa una condición, si ésta es verdadera ejecuta el bloque de instrucciones más cercano, en caso contrario, realiza acciones alternativas.
- *Múltiple o según sea*. Evalúa una condición y dependiendo de su valor booleano elige el bloque de instrucciones a ejecutar de entre varias opciones.

Simple

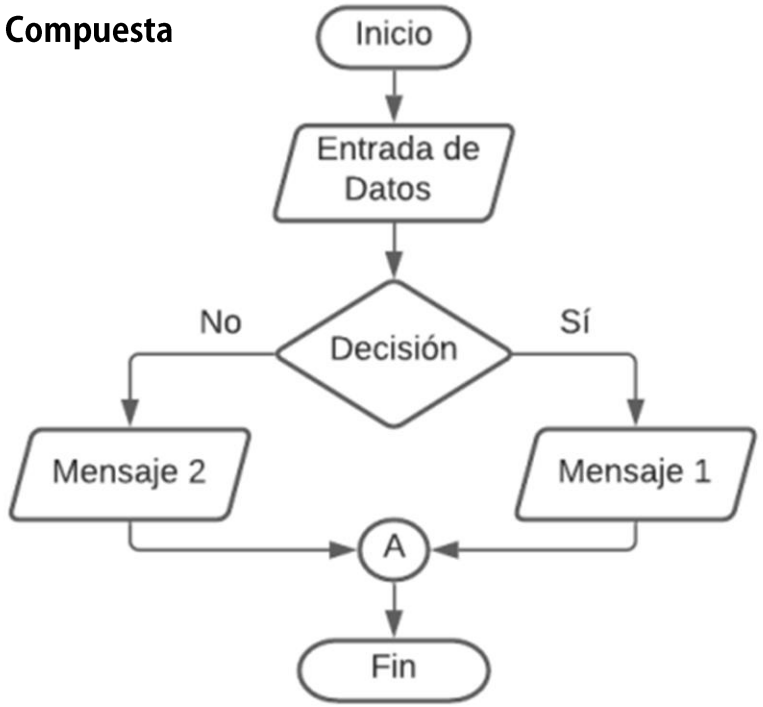


Para representar en un diagrama de flujo una *estructura condicional compuesta*, realiza los siguientes pasos:

- Identifica las entradas, salidas de datos, decisiones y procesos de tu algoritmo.
- Asignarle un título a diagrama de flujo.
- Dibuja el símbolo de inicio, debajo de este traza una línea de flujo vertical para unirlo con el siguiente elemento.

- Utiliza el símbolo de entrada de datos y escribe dentro de este el nombre de la variable que almacenará la información del usuario.
- Une este elemento a un símbolo de decisión y traza una línea de flujo que se dirija de cada uno de los vértices del rombo hacia los siguientes elementos. Estas líneas representan el camino que seguirá el proceso en caso de que la expresión sea verdadera o falsa. Si el resultado de la condición es verdadero, usa la salida de datos para mostrar en pantalla el mensaje pertinente. En caso contrario utilízalo para imprimir en pantalla el otro mensaje pertinente. Traza líneas de flujo de datos que se dirigen de los símbolos de salida a un conector.
- Por último, une el conector con el símbolo de fin

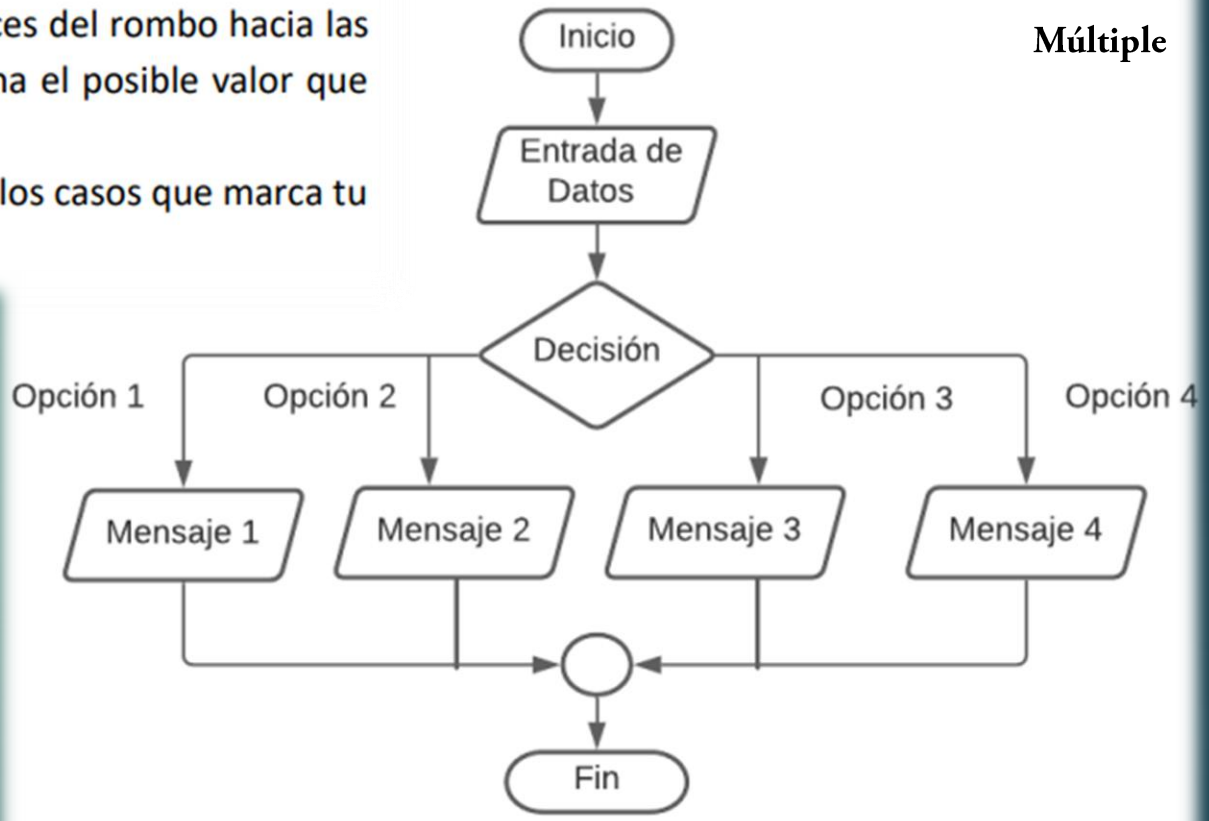
Compuesta



Si deseas representar una *estructura selectiva múltiple* por medio de un diagrama de flujo, adapta el diagrama de la siguiente forma:

- Utiliza el símbolo de decisión (rombo) y escribe el nombre de la variable dentro de este, sin asignarle ningún valor.
- Dibuja una línea de flujo que salga de cada uno de los vértices del rombo hacia las posibles opciones de ejecución y escribe al lado de cada una el posible valor que podría tomar la variable de salida del símbolo de decisión.
- Usa lectura de datos y escribe en su interior el contenido de los casos que marca tu algoritmo.

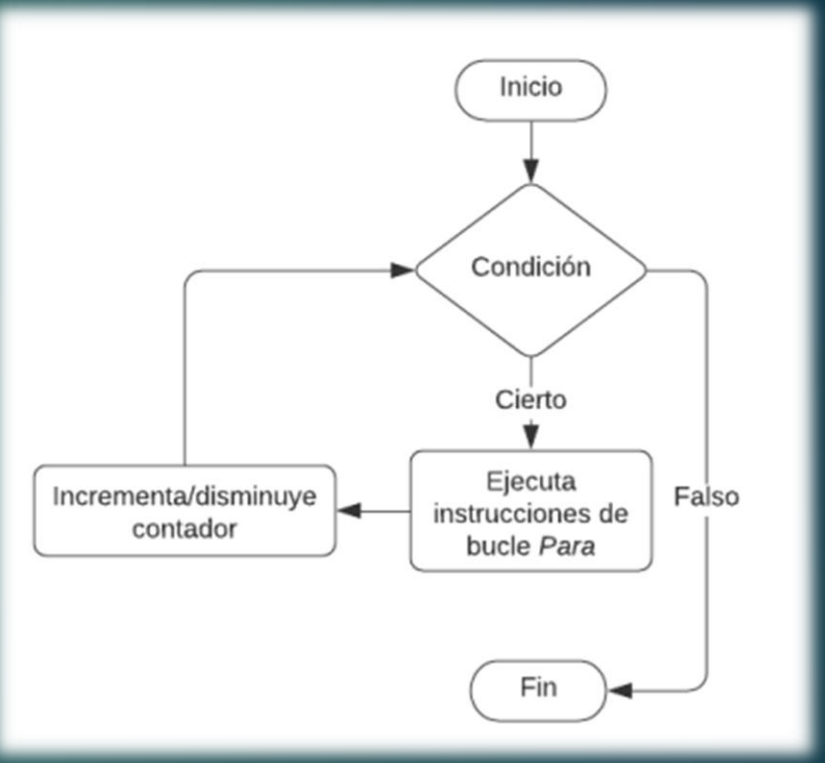
Múltiple



Las ***estructuras iterativas*** te permiten ejecutar un conjunto de instrucciones las veces que sea necesario mientras se cumpla una condición, es decir, realizar repeticiones o bucles. Cuando un ciclo se completa se comprueba nuevamente la condición y si es falsa el bucle se detiene.

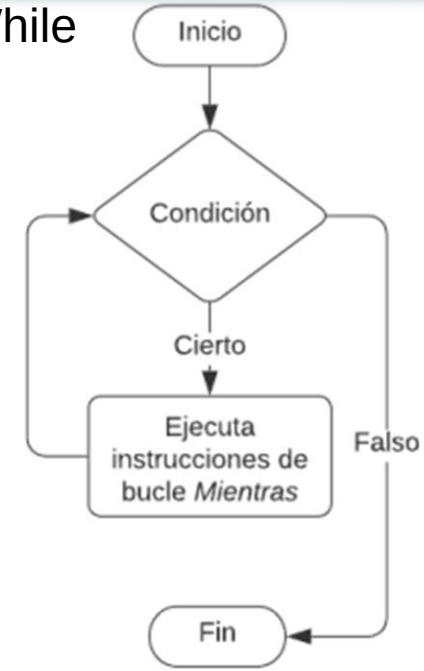
Existen distintos tipos de estructuras como son:

- Para (for). Con ella se establece el número de veces que una serie de instrucciones debe repetirse, lo cual determinas como parte de la solución a un problema en un algoritmo. Sus componentes son:
  - Expresión inicial. Indica con qué valor numérico inicia el ciclo.
  - Condición. Es la expresión relacional o lógica por evaluar, con ella se determina cuándo se detendrá el ciclo
  - Incremento. Indica el valor numérico que se le sumará a la expresión inicial tras completar el ciclo.

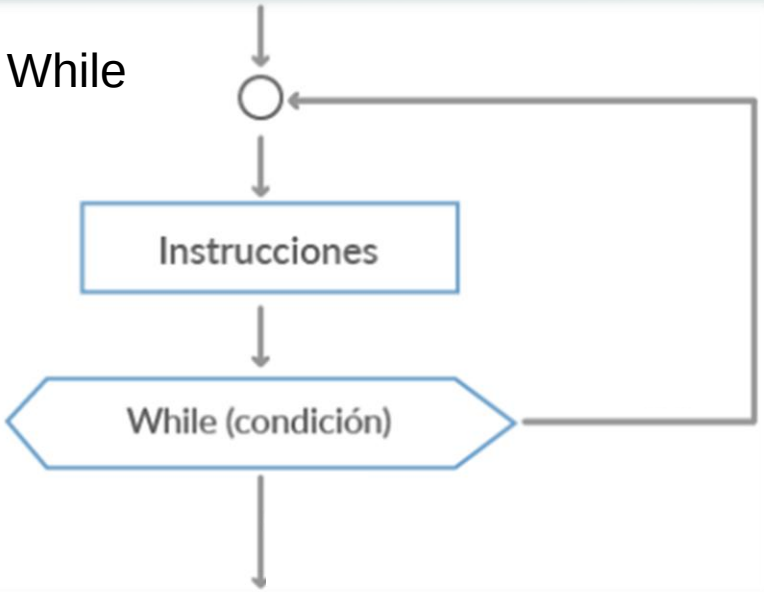


- Mientras (while). Se emplea para ejecutar un bloque de instrucciones en un ciclo sin necesidad de establecer el número de veces que lo hará. Se compone por una expresión lógica, relacional o aritmética, que es evaluada en una condición.

While



Do While



- Hacer Mientras (do...while). Se usa cuando el problema a resolver requiere que se ejecute por lo menos una vez el ciclo. Se compone por una condicional cuya expresión se evalúa después de ejecutar el bloque de instrucciones. El ciclo finaliza cuando la condición se vuelve falsa.

Pseudocódigo

De las raíces Pseudo (Supuesto) y Código (Instrucción). El ***pseudocódigo*** es un lenguaje para las especificaciones de algoritmos. Permite realizar la narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado.

Incluye una serie de convenciones léxicas y gramaticales parecidas a la mayoría de los lenguajes de programación, pero sin llegar a la rigidez de sintaxis de estos ni a la fluidez del lenguaje coloquial. A pesar de que las convenciones no cuentan con un estándar, no afecta la utilidad de la herramienta, que es una opción ágil para el estudio y diseño de soluciones.

Los beneficios que proporciona son:

- Representar de forma fácil operaciones repetitivas complejas.
- Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.
- Si se siguen las reglas de alineación, pueden observarse claramente los niveles en la estructura del programa.

Pseudocódigo

La relación de Convenciones empleadas en el pseudocódigo es la siguiente:

- Asignar un nombre al algoritmo.
- Tener un único punto de inicio.
- Contemplar un número finito de posibles puntos de término.
- Contemplar un número finito de caminos, entre el punto de inicio y los posibles puntos de término.
- Mostrar las palabras reservadas del pseudocódigo en negritas.
- Alinear los bloques de código de acuerdo al nivel de la instrucción en la estructura del programa.
- Emplear oraciones en lenguaje natural, donde cada una se refiere a una actividad general o específica.
- Utilizar lenguaje común.
- Evitar los errores gramaticales, abreviaciones y puntuaciones.
- Cuando exista la posibilidad de elegir algún elemento a partir de un conjunto de elementos, éstos se enlistarán separados por el símbolo pipe ("|").

Pseudocódigo

Estructura	Descripción	
	Convención	Ejemplo
Inicio Fin Secuencial	Inicio Instrucción 1 Instrucción 2 Instrucción 3 . . . . . Instrucción N Fin	Inicio Recibe calificaciones parciales Suma calificaciones Calcula promedio Entrega promedio Fin

Selección Simple	Si (condición)
	entonces {
	Instrucciones
	}
		Si (numero < 0) entonces {
		Negativo
		}

Selección
Doble
	Si (condición) entonces {
	Instrucciones
	}
	Si no {
	Instrucciones
	}
		Si (calificación =>6) entonces {
		Acreditado
		}
		Si no {
		No Acreditado
		}

Pseudocódigo

Iteración (mientras)	Mientras (condición) hacer {
	
	Instrucciones
	
	}
		Mientras (dinero >0) hacer {
		comprar
		dinero=dinero-montoCompra
		}

Iteración (Desde/hasta)	para contador = número hasta N hacer {
	Instrucciones
	Incrementar contador
	}
		para abdominal= 0 hasta 10 hacer {
		Hacer abdominal
		abdominal= abdominal+1
		}

Iteración (Hacer mientras)	Hacer { Instrucciones } mientras (condición)	Hacer { Calificar alumno siguiente } mientras (¿hay alumnos sin calificar?)

Pseudocódigo

Para ejemplificar un pseudocódigo, se ha construido la solución al problema que consiste en calcular el monto total de la compra de libros en una librería, el número de libros comprados es variable, se lee el precio de cada libro, se suma y se despliega el total de la compra, si el monto excede de $1000.00 pesos se otorga un descuento del 20%.

```
Inicio
   Recibe Numero_ de_ libros
   para libro = 1 hasta Numero_de_libros hacer
   {
      recibe precio_libro
      suma = precio_libro+suma
      monto_total=suma
    }
      si (monto_total >=1000) entonces {
      descuento = monto_total*0.2
      monto a pagar = monto_total – descuento
      entrega Monto_total, descuento, monto a pagar
      }
      si no {
      entrega “No hay descuento”, monto_total
      }
}
Fin
```