Introducción a C

Primera parte

Sofía Vitale

1. ¿Qué es un programa?

Un programa está constituído por un conjunto de instrucciones que se ejecutan de modo secuencial.

1.a Lenguajes de programación:

Las instrucciones que un procesador digital es capaz de comprender están constituídos por conjuntos de unos y ceros. Por lo tanto debemos diferenciar diveros lenguajes:

Lenguaje de programación de bajo nivel:

- lenguaje de máquina
- lenguaje ensamblador

Lenguaje de alto nivel: ejemplo C, C++, Java, entre otros.

Lenguaje de máquina o binario: consiste en cadenas de números que finalmente se reducen a unos y ceros. La CPU es capaz es capaz de interpretar el significado de cada una de esas secuencias de bits y llevar a cabo las acciones que codifican.

Lenguaje ensamblador: códigos mnemotécnicos, abreviaturas que representan las operaciones básicas de la computadora.
Estos códigos mnemotécnicos sólo pueden ser interpretados por la máquina gracias a programas llamados ensambladores, que convierten programas en lenguaje ensamblador a lenguaje de máquina.

La programación en lenguaje ensamblador es sumamente complicada, y no existe un lenguaje ensamblador universal, por lo que debería diseñarse un programa diferente para cada ordenador; esto se soluciona con el elnguaje de alto nivel.

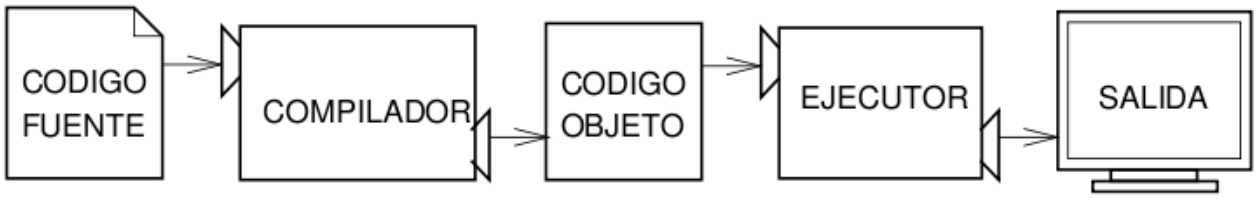
Lenguaje de alto nivel: señala la independencia de un ordenador concreto, este lenguaje permite escribir instrucciones mucho más entendibles para los programadores, como por ejemplo C, C++, Java.

Estos lenguajes de alto nivel deben traducirse a código de máquina, hay 2 tipos diferentes de traductores de pendiendo de su modo de funcionamiento: compiladores e intérpretes.

1.b Compliladores e intérpretes

Intérprete:
Un intérprete lee un programa y lo ejecuta, lleva a cabo lo que indica el programa. Traduce el programa poco a poco, leyendo y ejecutando cada comando.
Cada vez que ejecutamos un programa con un intérprete se repite el proceso de traducción y ejecución, ya que ambos son simultáneos. Lee un programa instrucción a instrucción y lo va traduciendo al código de máquina equivalente:

Compilador:
Un compilador lee el programa y lo traduce todo al mismo tiempo, antes de ejecutar cualquiera de las instrucciones. En este caso, al programa de alto nivel se le llama el código fuente, y al programa traducido el código de objeto o el código ejecutable. Una vez compilado el programa, puede ejecutarlo repetidamente sin volver a traducirlo, puede ejecutarse repetidamente sin necesidad de volver a traducir el programa original:



Compiladores e intérpretes... de idiomas

Puede resultarte de ayuda establecer una analogía entre compiladores e intérpretes de lenguajes de programación y traductores e intérpretes de idiomas.

Un compilador actúa como un traductor que recibe un libro escrito en un idioma determinado (lenguaje de alto nivel) y escribe un nuevo libro que, con la mayor fidelidad posible, contiene una traducción del texto original a otro idioma (código de máquina). El proceso de traducción (compilación) tiene lugar una sola vez y podemos leer el libro (ejecutar el programa) en el idioma destino (código de máquina) cuantas veces queramos.

Un intérprete de programas actúa como su homónimo en el caso de los idiomas. Supón que se imparte una conferencia en inglés en diferentes ciudades y un intérprete ofrece su traducción simultánea al castellano. Cada vez que la conferencia es pronunciada, el intérprete debe realizar nuevamente la traducción. Es más, la traducción se produce sobre la marcha, frase a frase, y no de un tirón al final de la conferencia. Del mismo modo actúa un intérprete de un lenguaje de programación: traduce cada vez que ejecutamos el programa y además lo hace instrucción a instrucción.

Ventajas y desventajas de intérpretes y compiladores:

Por regla general los intérpretes (a diferencia de los compiladores) ejecutan los programas más lentamente, pero son más flexibles, más portables, pueden ejecutarse en tipos diferentes de computadores sin modificación alguna o con pocas modificaciones.

Lenguaje compilado: las órdenes son transformadas a lenguaje de máquina que se almacena en un archivo ejecutable. Ese archivo puede ejecutarse luego, sin recurrir al compilador. Ejemplo C

Lenguaje interpretado: lenguajes que son interpretados, cuyas órdenes pasan a través de un intérprete que se encarga de ejecutarlas (a partir del código fuente) en el mismo momento en que están siendo leídas. Ejemplo Phyton, Perl

2. Algoritmos y programación estructurada

2.a Algoritmos:

Un algoritmo es una secuencia de instrucciones que permiten obtener un resultado en particular.
Es un método para resolver un problema mediante una serie de pasos: *precisos, definidos y finitos.*

2.b Programación estructurada:

La programación estructurada sigue 3 reglas:

- secuencia: indica que las instrucciones del código se leerán de principio a fin
- iteración: según cierta condición, un número de instrucciones podrían repetirse un número determinado de veces
- decisión: indica que según unas ciertas condiciones se ejecutarán o no un conjunto de instrucciones.

Se utiliza la indentación que indica cuales instrucciones son englobadas dentro de otras.

Ejemplo:

```
mientras haya platos
    tomar plato
    mientras haya suciedad
       echar jabón
       pasar el paño por el plato
    si plato es azul
       ponerlo con los azules
```

En código no estructurado, quedaría algo más “complicado”:

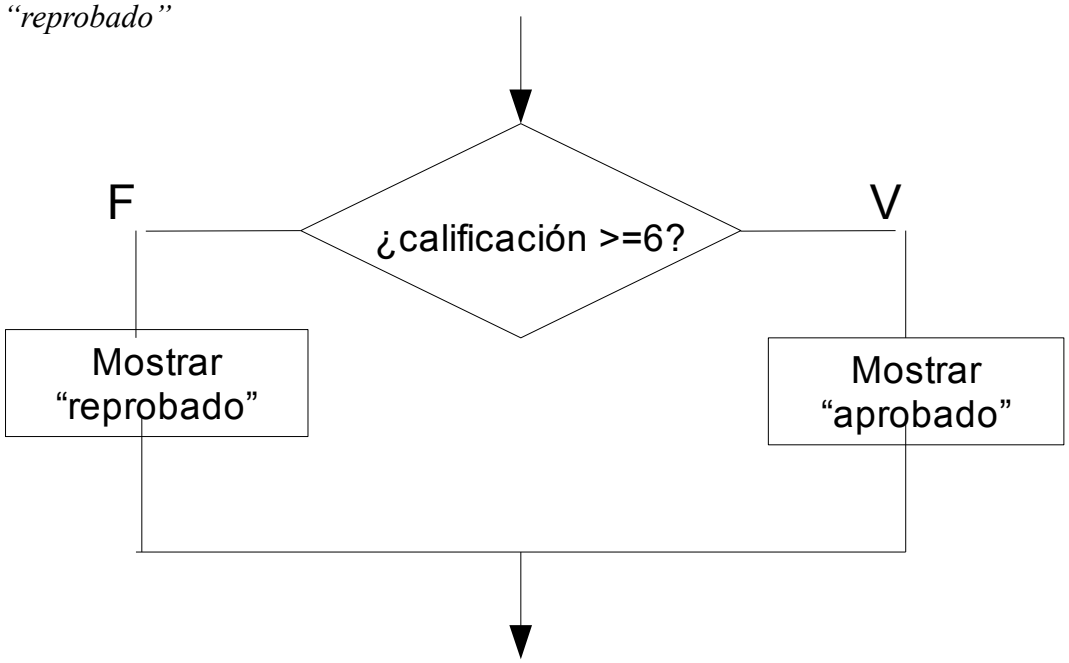
```
1 tomar plato
2 echar jabón
3 pasar el paño por el plato
4 si hay suciedad ir a la instrucción 2
5 si el plato no es azul ir a la instrucción 7
6 ponerlo con los azules
7 si hay más platos ir a la instrucción 1
```

3. Pseudocódigo y diagrama de flujo:

Permite entender de forma más sencilla un algoritmo, es el lenguaje que utilizamos para crear algoritmos de manera clara.
Se pueden representar mediante un diagrama de flujo (representación gráfica de un algoritmo); en el diagrama se pueden observar claramente las ideas principales y los pasos a seguir, es transferible a cualquier lenguaje de programación.

Ejemplo de pseudocódigo y diagrama de flujo para una sentencia selectiva (condicional):

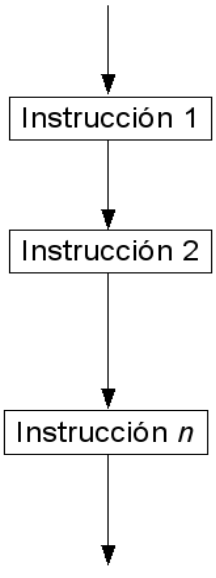
Si calificación del estudiante es mayor o igual a 6
mostrar "aprobado"
sino
mostrar "reprobado"



Estructuras de control:

a) Estructuras secuenciales:

instrucción 1
instrucción 2
instrucción 3
....
instrucción n



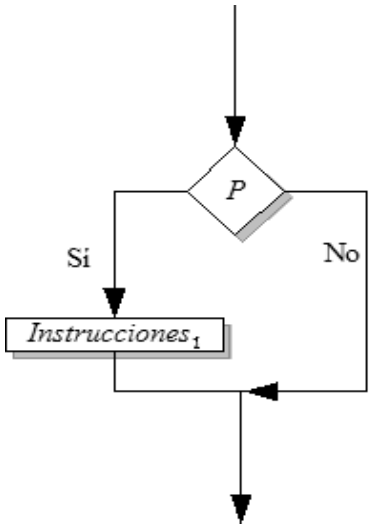
b) Estructuras selectivas (condicionales):

Las instrucciones selectivas representan instrucciones que pueden o no ejecutarse, según el cumplimiento de una condición.
Hay tres tipos principales:

- selección simple *(if)*
- selección doble *(if …else)*
- selección múltiple *(switch)*

Sectiva simple (if)
Si (if) condición entonces
instrucciones
fin si

Si la condición es verdadera se ejecuta la instrucción, sino no.

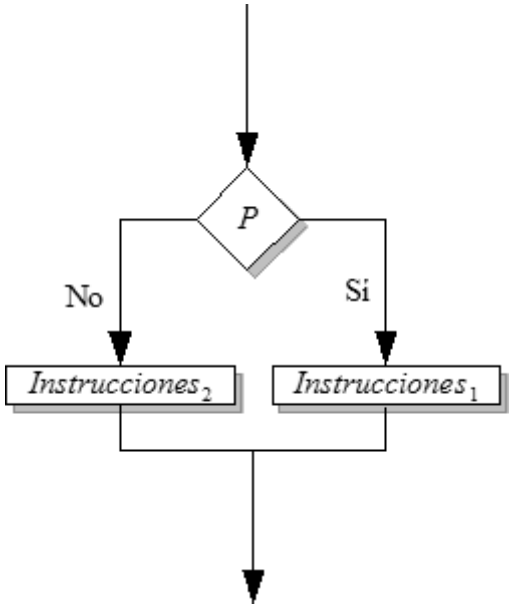


Ejemplo:

si calificación mayor o igual a 6 (condición)
mostrar aprobado (instrucción)
fin si

Selectiva o condicional doble (if... else):
Se realizan acciones diferentes según si la condición es verdadera o falsa.

Si (if) condición entonces
instrucción 1
sino entonces (else)
instrucción 2
fin si



Ejemplo:

Si calificación del estudiante es mayor o igual que 6
mostrar "aprobado"
sino
mostrar reprobado

Otro ejemplo: if...else anidadas:
Supongamos que la calificación es de la categoría:

- A si es >= 9
- B si es >=8
- C si es >=7
- D si es >=6
- F para todos los demás casos

Entonces así quedaría el pseudocódigo:

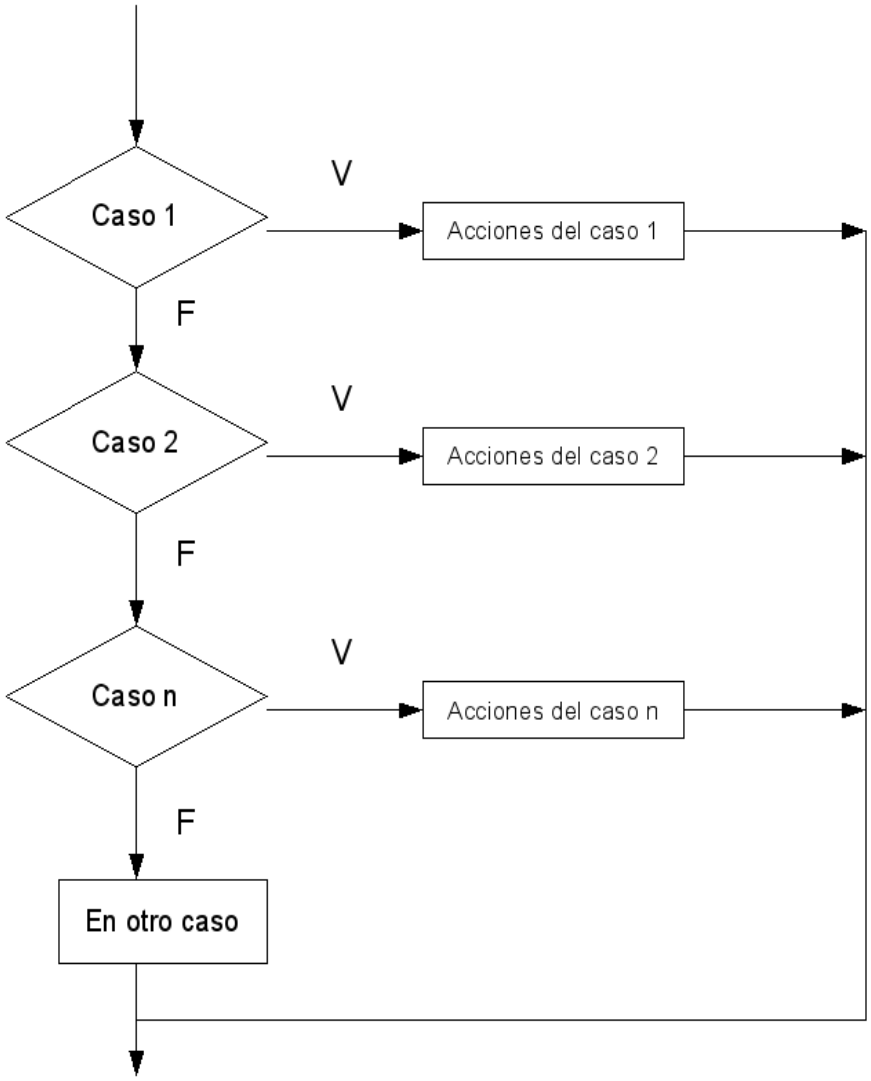
si calificación del estudiante es mayor o igual a 9
mostrar "A"
si no entonces
si calificación del estudiante es mayor o igual a 8
sino entonces
si calificación del estudiante es mayor o igual a 7
mostrar "C"
sino entonces
si calificación del estudiante es mayor o igual a 6
mostrar "D"
sino entonces
mostrar "F"

Selectiva múltiples casos: instrucción de selección múltiple Switch...case

Seleccionar indicador
caso valor 1
instrucciones 1
caso valor 2
instrucciones 2
caso valor 3
instrucciones 3
.....
en otro caso
instrucciones n
fin seleccionar

En este caso hay un *Indicador* que es una variable o una función cuyo valor es comparado en cada caso con los valores "$Valor_i$", si en algún caso coinciden ambos valores, entonces se ejecutarán las $Instrucciones_i$ correspondientes.

Diagrama:



Ejemplo:
Supongamos que según el nivel del trabajador calculamos el sueldo para los diferenetes niveles (1, 2 y 3):

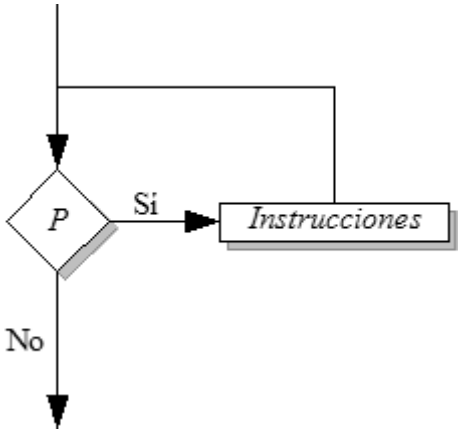
seleccionar (switch) nivel (indicador)
nivel 1 (case 1):
salario x 0,35
mostrar el resultado del salario
ir al fin de la instrucción switch (break)
nivel 2 (case 2):
salario x 0,50
mostrar el resultado del salario
ir al fin de la instrucción switch (break)
nivel 3 (case 3):
salario x 0,80
mostrar el resultado del salario
ir al fin de la instrucción switch (break)
en otro caso (default):
mostrar "error, debe ingresar el valor 1, 2 o 3"

c) Instrucciones iterativas (repetitivas)
Representan la ejecución de instrucciones en más de una vez.
Hay 3 tipos:
- instrucción de repetición *while*
- instrucción de repetición *do...while*
- instrucción der repetición *for*

Bucle mientras (instrucción repetitiva while): si la condición es falsa no se ejecuta ninguna vez la instrucción/es del bucle while

Mientras la condición es verdadera se realiza la acción

mietras (while) condición hacer
instrucción
fin mientras



Ejemplo:

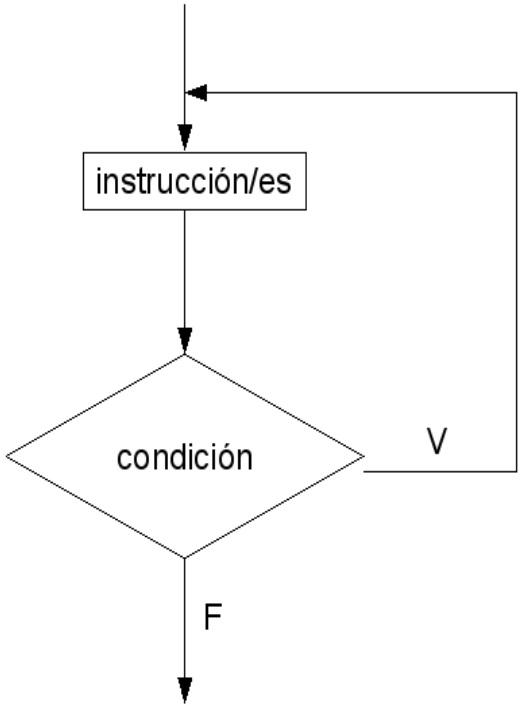
Mientras edad mayor a 18
mostrar "es mayor"
fin mientras

Bucle repetir: instrucción de repetición do ...while
Se utiliza cuando es necesario que el bucle se ejecute al menos una vez y hasta que se cumpla la condición.

Hacer instrucción
mientras condición

Ejemplo:

Supongamos que partimos de de una variable llamada contador que vale 1, a la que se le va sumando un número:
contador = 1
sumar 1
mientras la variable contador sea menor o igual a 10

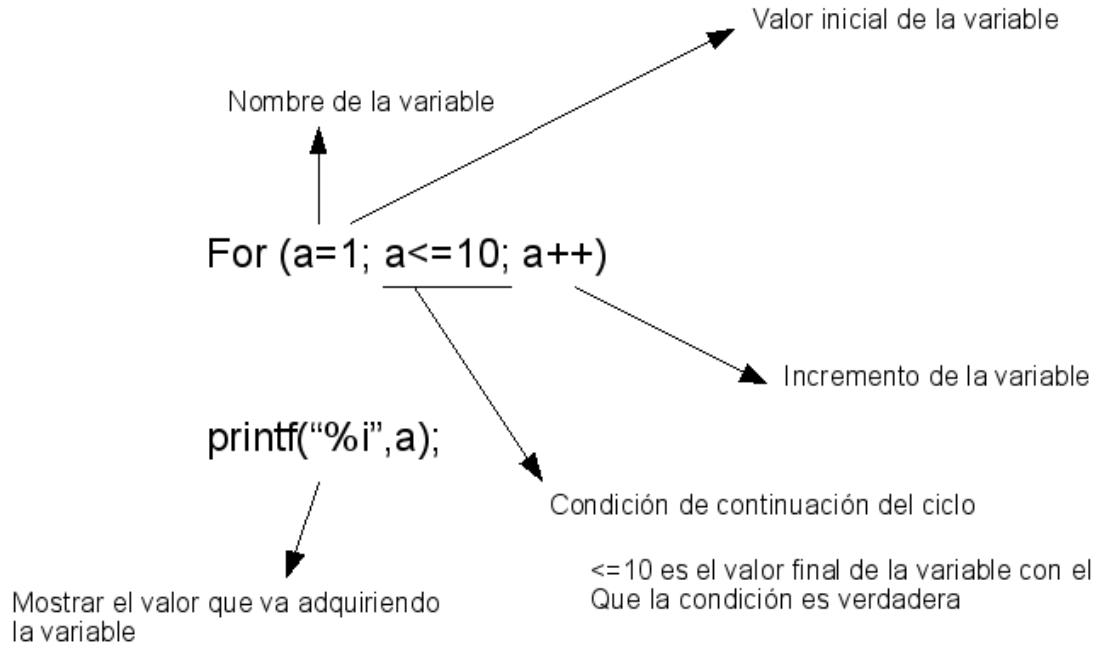


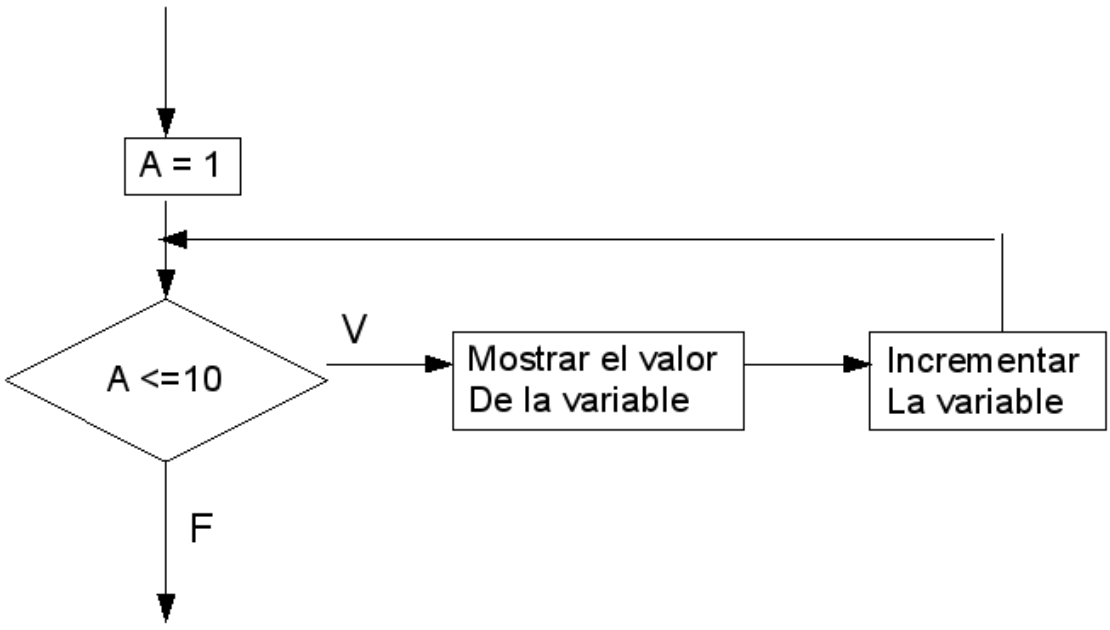
Bucle para: instrucción de repetición for
Se usa cuando se desea iterar un número conocido de veces, empleando un índice de variable que se incrementa o decrementa.

Para i <-------- x hasta n hacer
instrucción
fin para

Ejemplo:

Supongamos que quiero mostrar los números desde el 1 hasta el 10, supongo una variable llamada a, la cual parte del valor 1, a esta variable se le incrementa 1 hasta llegar al número 10:





4.Primer programa en C:

Este primer programa mostrará la palabra “Hola”; es una instrucción de tipo secuencial:

```
/*este programa muestra “hola”*/
#include <stdio.h>
void  main()
{
  printf(“Hola”);
}
```

/*: lo que escribimos entre */ serán los comentarios, para poder explicar que hace el programa, al compliar el programa es ignorado, por lo que puedes expresarte de la forma que quieras.

#include <stdio.h>: indica que el programa incluya el contenido del encabezado estandar, que utiliza funciones de la biblioteca estandar de C, como por ejemplo printf.

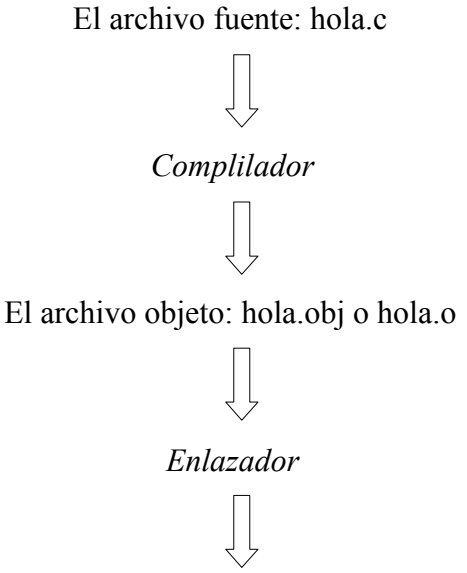
void main(): main es la función principal, forma parte de todos los programas en C, void se refiere a que no devuelve nada, void= vacío, no va a devolver ningún valor.

5. Crear un programa ejecutable:

Para escribir programas se necesitan estas herramientas:

- un editor de textos
- un complilador y enlazador

Una vez compilado el código fuente se genera un archivo llamado archivo objeto o programa objeto que es luego enlazado mediante el enlazador, para generar el archivo ejecutable.
Los compiladores actuales suelen hacer dos funciones de una vez, compilando y enlazando todo en una sola función, aunque es posible pedirles que no lo hagan mediante parámetros adicionales.
Según el compilador y la configuración utilizada, se obtendrán dos o tres archivos:



6. Tipos de datos

El C, como cualquier otro lenguaje de programación, tiene posibilidad de trabajar con datos de distinta naturaleza: texto formado por caracteres alfanuméricos, números enteros, números reales con parte entera y parte fraccionaria, etc.

Tipo	Bits	Rango
char	8	0 a 255
int	16	-32.768 a 32.767
float	32	3,4 E -38 a 3,4 E +38
double	64	1,7 E -308 a 1,7 E +308
void	0	sin valor

Enteros (int): Se usan para representar números enteros. Los tipos de datos enteros son: *short, int, long y long long.*

Flotantes (float): Se denomina flotantes a los tipos de datos que representan a los números reales.

El tipo de dato flotante en lenguaje C sólo tiene dos tamaños: el *float* y el *double*, que son 4 bytes y 8 bytes respectivamente. Se los puede utilizar tanto para representar números decimales, como para representar números enteros con un orden de magnitud muy grande.

Caracteres: (char): Este tipo se utiliza para representar los 255 caracteres de la tabla de caracteres del sistema. El tipo char es también un tipo entero, ya que puede tomar valores de 0 a 255.

El **void** se usa para declarar funciones que no devuelven ningún valor o para declarar funciones sin parámetros.

Modificadores de tipos

unsigned
signed
long
short

Los modificadores signed, unsigned, long y short se pueden aplicar a los tipos base entero y carácter. Sin embargo, long también se puede aplicar a double.

Tipo	Bits	Rango
char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
int	16	-32.768 a 32.767
unsigned int	16	0 a 65.535
signed int	16	-32.768 a 32.767
short int	16	-32.768 a 32.767
unsigned short int	16	0 a 65.535
signed short int	16	-32.768 a 32.767
long int	32	-2147483648 a 2147483647
signed long int	32	-2147483648 a 2147483647
float	32	3,4 E -38 a 3,4 E +38
double	64	1,7 E -308 a 1,7 E +308
long double	64	1,7 E -308 a 1,7 E +308

7. Variables y constantes:

7.a Variables

A las variables se les asigna un nombre identificador es un nombre simbólico que se refiere a un dato o programa determinado.

Un ejemplo: queremos sumar dos números. Nuestro programa tendrá que tener tres cajones: Uno para cada número y otro para el resultado. Cada cajón tiene un nombre en vez de una posición de memoria, de manera que sólo hay que nombrarlo:

```
Necesito cajones A, B y Resultado

Lee un número y guárdalo en A
Lee un número y guárdalo en B
Suma A y B y guárdalo en Resultado
Imprime el contenido de Resultado
```

Las posiciones de memoria A y B son variables.

Las variables pueden cambiar de valor a lo largo de la ejecución de un programa, o bien en ejecuciones distintas de un mismo programa.

7.b Constantes

Además de variables, un programa utiliza también constantes, es decir, valores que siempre son los mismos.
Un ejemplo típico es el número π, que vale 3.141592654. Este valor, con más o menos cifras significativas, puede aparecer muchas veces en las sentencias de un programa. En C existen distintos tipos de constantes:

Constantes numéricas. Son valores numéricos, enteros o de punto flotante.

Constantes caracteres. Cualquier carácter individual encerrado entre apóstrofos (tal como 'a', 'Y', ')', '+', etc.) es considerado por C como una constante carácter, o en realidad como un número entero pequeño (entre 0 y 255, o entre -128 y 127, según los sistemas). Existe un código, llamado código ASCII, que establece una equivalencia entre cada carácter y un valor numérico correspondiente.

Cadenas de caracteres. Un conjunto de caracteres alfanuméricos encerrados entre comillas es también un tipo de constante del lenguaje C, como por ejemplo: "espacio"

Constantes simbólicas. Las constantes simbólicas tienen un nombre (identificador) y en esto se parecen a las variables. Sin embargo, no pueden cambiar de valor a lo largo de la ejecución del programa.

8. Operadores:

Los operadores son signos especiales –a veces, conjuntos de dos caracteres– que indican determinadas operaciones a realizar con las variables y/o constantes sobre las que actúan en el programa.

En C hay tres clases de operadores: aritméticos, relacionales y lógicos, y a nivel de bits.

Aritmeticos

-	resta
+	suma
*	producto
/	división
%	módulo (resto de la división entera)
--	decrementar
++	incrementar

```
x=10;                    x=10;
y=++x;                   y=x++;
y=11                     y=10
```

Relacionales

En C cierto es cualquier valor distinto de cero. Falso es cero.

>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que
==	igual
!=	distinto

Lógicos

&&	y
||	o
!	no

9 . Bibliografía

- Cómo programar en C/C++ y Java, 4º edición, Deitel, Deitel
- Programación estructurada en C, José R. García-Bernejo Giner
- Introducción a la programación con Phyton, Andrés Marzal , Isabel Gracia
- Aprenda lenguaje ANSI C como si estuviera en primero, Javier García de Jalón de la Fuente, José Ignacio Rodríguez Garrido · Rufino Goñi Lasheras , Alfonso Brazález Guerra, Patxi Funes Martínez, Rubén Rodríguez Tamayo

Documentación web:
http://es.wikipedia.org/wiki/Pseudoc%C3%B3digo
http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C