Introducción a lenguaje C

Javier Fernández Rivera - www.aurea.es

Que es el lenguaje C?

El lenguaje C es un lenguaje de programación estructurado. Lo que quiere decir que el código o algoritmo esta ordenado o estructurado. Así pues es fácilmente diferenciable de un lenguaje orientado a objetos.
El lenguaje C es uno de los lenguajes mas rapidos y potentes que existen hoy en día con una sintaxis sumamente compacta y de alta portabilidad.
Es común leer que se lo caracteriza como un lenguaje de "bajo nivel". No debe confundirse el término "bajo" con "poco", ya que el significado del mismo es en realidad "profundo", en el sentido que C maneja los elementos básicos presentes en todas las computadoras: caracteres, números y direcciones .
Además este lenguaje sirve de puente a otros, lo que quiere decir que sabiendo lenguaje C, sabrás defenderte fácilmente con cualquier otro lenguaje de programación estructurada, solo tendrás que cambiar las palabras reservadas y poco mas.
Para verificar la importancia y magnitud de este lenguaje solo hay que observar que el sistema operativo **Linux** el cual ha sido desarrollado en su practica totalidad con el C. Además vemos que en las universidades y ciclos formativos referentes a la informática exigen el aprendizaje absoluto de este lenguaje, en muchos casos por encima del resto.
El lenguaje C no es lo mismo que el lenguaje C++ como en muchos casos se piensa. Para controlar este ultimo hace falta tener una buena base de C. Se podría decir que C++ es una extensión del C y con el podríamos llegar a introducirnos en la programación basada en windows. C++ incorpora el potencial de C al servicio de una programación basada en el SO de Microsoft.
Pero no todo es oro lo que reluce en este lenguaje, es lógico que con tanto potencial halla determinados puntos “oscuros”,nos referimos pues a que el lenguaje en C es de “caso sensible”, lo que quiere decir que distingue entre mayúsculas y minúsculas y otros casos. Volviéndonos locos por ser un lenguaje pelin quisquilloso. Pero su potencial obliga a ello.
Por otro lado, todas las instrucciones que damos en C, se separan o finalizan con el “**;**” (punto y coma).
La descripción del lenguaje se realiza siguiendo las normas del ANSI C.

El lenguaje C ofrece a los usuarios ventajas y desventajas, dependiendo de su nivel de conocimiento del mismo.

1. Es un lenguaje complicado para el usuario novel (el usuario requiere cierta experiencia para empezar a trabajar).
2. Suministra una vision de conjunto.
3. Eficacia.
4. Nos ofrece libertad para organizar el trabajo.
5. Lenguaje de alto nivel, se asemeja al lenguaje humano (normalmente el ingles).
6. Es un lenguaje diseñado para la resolución de problemas, independientes de las características del computador.
7. En ocasiones determinadas nos permite programar como lenguaje de bajo nivel, consiguiendo una mayor eficacia.
8. Flexibilidad
9. Muy POTENTE
10. Es muy usado en aplicaciones científicas, industriales, simulaciones de vuelo; es decir, se aplica en áreas desconocidas por gran parte de los usuarios.
11. No es un lenguaje muy estructurado como ocurre en lenguajes como ALGOL 68 o PASCAL.

Compilador de lenguaje C

Antes de todo explicare que es un compilador. Un compilador dicho sencillamente, seria un programa capaz de hacer otros programas.
Profundizando mas:
El ordenador trabaja unica y exclusivamente con combinaciones de ceros y unos. Por decirlo de alguna manera, solo entiende el llamado código binario. Ahora bien, si nosotros quisiéramos desarrollar un programa para algún ordenador, deberíamos hacerlo a base de combinaciones de ceros y unos. Deberíamos escribirlo en código binario. Esto seria una labor muy tediosa, prácticamente imposible, tarea reservada a gurus de las matemáticas jejeje. En el mundo podrían contarse con los dedos de las manos los que saben programar en lenguaje binario, son los encargados de programar a mas bajo nivel las funciones de los microprocesadores.
Ahora es aquí donde entran los lenguajes como el C, delphi, basic, ect. Estos lenguajes lo que hacen es poner a nuestro alcance una serie de instrucciones que nosotros podemos entender con facilidad. Funciones racionales para el hombre. Aquí se encuentran las palabras reservadas (if, goto, else, while...). Una vez escrito todo el programa mediante estas reglas racionales que nos sirve el lenguaje que usemos, tendremos lo que se llama el código fuente. Pero este solo lo entienden los programadores no el ordenador, que como habíamos visto solo entendía ceros y unos. Es aquí donde actúa el compilador. Este es el encargado de pasar nuestro código fuente (ordenes racionales) a código binario (lenguaje que entiende el ordenador). Y es así como de nuestro código fuente podemos obtener un ejecutable. Un fichero que el ordenador podrá interpretar y ejecutar, los llamados (*.exe), otros ficheros ejecutables serian (*.bat, *.con).
Teniendo el código fuente podremos modificar nuestro programa y compilarlo para generar otro ejecutable tantas veces como queramos. Pero de no tener el código fuente no podremos modificar nada.

A menudo se confunden los distintos tipos de compiladores. Existen 3 tipos de compiladores, cuya función es la misma. Pasar de código fuente a maquina.

- **Compiladores:** Propiamente llamados, se dedican a traducir el código fuente a código maquina.
- **Interpretes:** Va cogiendo cada instrucción cuando la precise y la va convirtiendo en su instrucción correspondiente en codigo maquina. Para aquellos que programeis en scrpitting, el cliente mIRC para IRC nos facilita un interprete para elaborar lo que se llama un script (subprograma). Mas información en: www.ircorion.com
- **Entornos de desarrollo:** La mezcla de los dos sistemas, primero interpreta para posteriormente compilar sin errores.

Ventajas y desventajas

Ventaja de un compilador: Una vez compilado su ejecución y tiempo de proceso es mas rapido.
Ventaja del interprete: Se puede depurar el código de forma mas rapida y sencilla.

Durante este curso se usara el archifamoso compilador de C para modo bajo MSDOS: DJGPP. (Con lo que nuestra programación durante este curso estará bajo MSDOS.). Podeís conseguir este compilador en: http://www.delorie.com/djgpp/
Aquellos usuarios de Linux podrán utilizar el GNU.

Para poder usar el DJGPP, acuerdate de insertar en el fichero autoexec.bat Estas líneas:

```
set DJGPP=C:\DJGPP\DJGPP.ENV
set PATH=C:\DJGPP\BIN;%PATH%
```

El editor

Un editor es simplemente el lugar donde editamos o escribimos nuestro código fuente. Podríamos hacerlo simplemente en el notepad de windows ☺ Y luego compilarlo con algún compilador. Pero el DJGPP, ya tiene su propio editor integrado, el llamado RHIDE.
Un buen editor es importante en cualquier lenguaje de programación. Puesto que mediante técnicas puede clarificar mucho nuestro código fuente. Técnicas como pueden ser el uso de colores, esto es que cuando escribimos una palabra reservada sale en un color distinto que cuando escribimos un dato o una cadena de caracteres a mostrar por pantalla. Existen también editores que van ordenando nuestro código fuente, estructurándolo.
Es importante escribir un código fuente claro y bien organizado o estructurado y con comentarios. Para que nuestros superiores o nosotros mismos podramos entenderlo con facilidad en posteriores revisiones.

Compilar y ver los resultados en DJGPP.

- **Como compilar en DJGPP?**

Una vez que tengamos nuestro código fuente, nuestro código del programa, debemos compilarlo, para ello acudimos al menú del DJGPP **COMPILE**, posteriormente marcamos sobre **BUILD ALL** (construir todo). Si a la hora de compilar el DJGPP no nos ha dado ningún error, nuestro programa quedara listo para ser ejecutado. En caso contrario a romperse el coco buscando el posible error.

- **Como ejecutar programas desde DJGPP?**

Suponiendo que tenemos un programa ya editado en el RHIDE (editor del DJGPP). Y que esta compilado. Debemos acudir al menú **FILE** y a continuación a DOS **SHELL**. De esta forma pasaremos al entorno bajo MSDOS y desde la línea de comandos (C.\DJGPP\bin\) escribimos el nombre del programa compilado. Y este será ejecutado bajo MSDOS.

Comentarios en el código fuente.

Muchas veces los buenos programadores hacen pequeños comentarios en partes de su código para clarificarlo. Esto se hace con la única función de que si al día siguiente o al mes siguiente queremos modificar alguna parte de nuestro código sepamos con rapidez y claridad las partes de este y para ello se especifican comentarios puestos por el propio programador para entender que es lo que hace una parte del código donde ha puesto el comentario. Esto también se suele hacer por si hace falta que otro programador necesite ver el código o revisarlo y así facilitarle la tarea de comprensión del mismo.

A la hora de compilar un código el compilador una vez que detecta los cometarios, (los detecta por la inclusión de unos signos específicos) lo que hace es pasar de largo seguir leyendo o procesando.

Para hacer comentarios en C se usan los signos de introducción /* y de finalización */.

Ejemplo:

```
#include stdio.h
main () {
printf ("OrioN ScripT by Quasi en http://www.ircorion.com") /*esto imprime en pantalla el script para IRC y la web*/
}
```

En este ejemplo observamos que no haría falta poner el comentario puesto que ya es claro la función del printf pero hay otros casos en los que se realizan bucles o ciertas operaciones en las que si seria de mucha utilidad poner comentarios aclaratorios.

Nuestro primer programa

Vamos a exponer el mitico programa que se suele hacer como primera aproximación a cualquier lenguaje de programación.
El, Hola Mundo!!!

Codigo fuente (ejemplo):

```
#include <stdio.h>
main () {
printf("Hola mundo!!!");
}
```

Ahora voy a comentar paso a paso el programa.

Comenzamos con: #include <stdio.h>
Include es una directriz que se encarga de meter librerías de funciones. En este caso metemos la librería "**stdio**" Esta librería contiene funciones de consola. Funciones de entrada y salida de datos, es donde se encuentra la función **printf**, posteriormente usada. Es la librería por excelencia de C. Y la que siempre nos veremos obligados a incluir.
Si no habeis entendido bien el concepto no os preocupar, mas adelante se habla de ello en profundidad.
A continuación viene la función **main ()**. Esta función es el cuerpo principal del programa como su propio nombre indica. Es el punto de partida y de salida de un programa.
Las ordenes en C es bueno agruparlas todas entre llaves.
Dentro del main metemos la función "**printf**" función que esta en la biblioteca "**stdio**" y lo que hace es mostrar por pantalla el texto que tenemos entre comillas. No olvidemos terminar la instrucción de la función con un "**;**". Y cerramos la llave que finaliza el conjunto de ordenes a realizar por el main. Las ordenes que se encuentran dentro del main siempre se ejecutaran puesto que es el cuerpo principal del programa. Vendría a ser como el tronco de un árbol, las ramificaciones de ese árbol seria otras funciones fuera del main.
El lenguaje C se caracteriza entre otras cosas por seguir un orden muy claro y estricto. Todos los programas deben presentar una anatomia comun.

1. Añadir todas las directrices o directivas que incluyan ficheros que contengan otros funciones que necesitemos para nuestro programa. Siempre o casi siempre deberemos insertar la librería "stdio", puesto que es la que contiene las funciones principales de toma (scanf) y muestreo de datos (printf). A la hora de añadir una librería siempre se siguen la misma sintaxis: **#include <NombreDeLaLibreria>**
2. Podemos obtener el nombre de las funciones o librerías que contienen las funciones que necesitemos acudiendo al menú **HELP** del compilador DJGPP.
3. Se declaran funciones o variables globales. Funciones externas creadas por nosotros mismos y definidas al final del main. O variables globales que tendrán vida dentro y fuera del main (cuerpo principal).
4. Se define la función main, se especifican todas las ordenes y se delimitan entre llaves tal y como se expone en el ejemplo anterior.
5. Se definen las funciones si es que las hay, anteriormente declaradas antes del main.

En todo programa que realicemos sobre este lenguaje C. Debemos tener muy en cuenta, el principio y el final de las cosas, y su orden y prioridad de ejecución. En C las ordenes se compilan de forma lineal y secuencial (de arriba a abajo y de una en una). En la programación esta mal visto que se pierda el ciclo de proceso de un programa, siempre debemos saber por que punto pasa el programa, como varia y cuando finaliza.

Estructura de datos

Esto que se explica a continuación si no lo entendeis no le deis mas importancia, pasar al capitulo siguiente.

Se dice que: **programa = algoritmo + estructura de datos**

Dependiendo la forma en la que se almacenan los datos tendremos lo que se conoce como una estructura de datos.
Las estructuras de datos vienen predeterminadas por la disposición y los valores en que vienen dadas.
Normalmente las estructuras habituales son:

Matrices: Es una secuencia de elementos del mismo tipo relacionados unos con otros por el orden en el que están definidos
Las matrices pueden ser: unidimensionales, bidimensionales y tridimensionales. Dependiendo claro esta de su dimensión.
Cada departamento o valor de una matriz viene identificado por un índice.
Listas: Conjunto de elementos en el que cada uno esta relacionado con el elemento anterior y siguiente. Los hay circulares.
Colas: Es un tipo especial de listas llamada FIFO (first imput first out)
Pilas: Conocidas como LIFO (last imput firts out)
Arbol: Es una estructura en la que un elemento va a estar relacionado con otro o con varios mas.
Fichero: Es una estructura que permite almacenar información en un dispositivo de almacenamiento. Las filas cuando almacenan datos están constituidas por una estructura que se llama registro. A cada parte de un registro se le llama campo.

Bueno ahora entremos ya en materia de C. ☺

Recordemos siempre que el C distingue entre nomenclaturas, o sea diferencia entre minúsculas y mayúsculas.
En C, es necesario poner punto y coma “;” al final de una orden, pero no se debe poner al final de una condición o estructura condicional tampoco se debe poner en otro tipo de ordenes como algunos bucles.

.