Programación en C para principiantes

Gorka Urrutia

Table of Contents

Capítulo 1. Introducción. 1
- Sobre el libro 1
- Cómo resolver tus dudas 1
- El lenguaje C 1
- Peculiaridades de C 1
- Compiladores de C 1
- El editor de código fuente 2
- IDE: Entorno de desarrollo integrado 2
- El primer programa: Hola Mundo 2
- ¿Cómo se hace? 4
- Nota adicional sobre los comentarios 5
- ¿Qué sabemos hacer? 5
- Ejercicios 5

Capítulo 2. Mostrando Información por pantalla. 6
- Printf: Imprimir en pantalla 6
- Gotoxy: Posicionando el cursor (requiere conio.h) 8
- Clrscr: Borrar la pantalla (requiere conio.h) 9
- Borrar la pantalla (otros métodos) 9
- ¿Qué sabemos hacer? 9
- Ejercicios 10

Capítulo 1. Introducción.

Sobre el libro

Este es un curso para principiantes así que intentaré que no haga falta ningún conocimiento anterior para seguirlo. Muchos otros cursos suponen conocimientos previos pero voy a intentar que eso no suceda aquí.

NOTA IMPORTANTE: Si te pierdes no te desanimes, ponte en contacto conmigo y consúltame (al final del libro tienes varias formas para contactarme). Puede que alguna sección esté mal explicada. De esta forma estarás colaborando a mejorar el libro.

Cómo resolver tus dudas

En la última sección del libro podrás encontrar varias formas de contactar conmigo (email, Twitter, mi blog, etc).

El lenguaje C

El lenguaje C es uno de los más rápidos y potentes que hay hoy en día. Hay quien dice que está desfasado. No se si tendrá futuro pero está claro que presente si tiene. No hay más que decir que el sistema operativo Linux está desarrollado en C en su práctica totalidad. Así que creo que no sólo no perdemos nada aprendiéndolo sino que ganamos mucho. Para empezar nos servirá como base para aprender C++ e introducirnos en el mundo de la programación Windows. Si optamos por Linux existe una biblioteca llamada gtk (o librería, como prefieras) que permite desarrollar aplicaciones estilo Windows con C.

No debemos confundir C con C++, que no son lo mismo. Se podría decir que C++ es una extensión de C. Para empezar en C++ conviene tener una sólida base de C. Existen otros lenguajes como Visual Basic que son muy sencillos de aprender y de utilizar. Nos dan casi todo hecho. Pero cuando queremos hacer algo complicado o que sea rápido debemos recurrir a otros lenguajes (C++, Delphi,...).

Peculiaridades de C

Una de las cosas importantes de C que debes recordar es que es Case Sensitive (sensible a las mayúsculas o algo así). Es decir que para C no es lo mismo escribir Printf que printf. Conviene indicar también que las instrucciones se separan por ";".

Compiladores de C

Un compilador es un programa que convierte nuestro código fuente en un programa ejecutable (me imagino que la mayoría ya lo sabéis pero más vale asegurar). El ordenador trabaja con 0 y 1. Si escribiéramos un programa en el lenguaje del ordenador nos volveríamos locos. Para eso están lenguajes como el C. Nos permiten escribir un programa de manera que sea fácil entenderlo por una persona (el código fuente). Luego es el compilador el que se encarga de convertirlo al

complicado idioma de un ordenador.

En la practica a la hora de crear un programa nosotros escribimos el código fuente, en nuestro caso en C, que normalmente será un fichero de texto normal y corriente que contiene las instrucciones de nuestro programa. Luego se lo pasamos al compilador y este se encarga de convertirlo en un programa.

Si tenemos el código fuente podemos modificar el programa tantas veces como queramos (sólo tenemos que volver a compilarlo), pero si tenemos el ejecutable final no podremos cambiar nada (realmente sí se puede pero es mucho más complicado y requiere más conocimientos).

Existen multitud de compiladores. Yo suelo recomendar el Geany y Code::Blocks, que tiene versiones tanto para Linux como para Windows. Estos programas usa el compilador GNU GCC (http://gcc.gnu.org) y se pueden descargar aquí:

- Geany - http://www.geany.org/
- Code::Blocks - http://www.codeblocks.org/

Nota: Cuando comencé a escribir el curso solía usar el DJGPP en Windows, sin embargo, ahora me decanto más bien por el Geany por la comodidad y facilidad que supone para los principiantes.

El editor de código fuente

El compilador en sí mismo sólo es un programa que traduce nuestro código fuente y lo convierte en un ejecutable. Para escribir nuestros programas necesitamos un editor. La mayoría de los compiladores al instalarse incorporan ya un editor; es el caso de los conocidos Turbo C, Borland C, Code::Blocks, Visual C++,... Pero otros no lo traen por defecto. No debemos confundir por tanto el editor con el compilador. Estos editores suelen tener unas características que nos facilitan mucho el trabajo: permiten compilar y ejecutar el programa directamente, depurarlo (corregir errores), gestionar complejos proyectos, etc. Si nuestro compilador no trae editor la solución más simple usar un editor de texto plano (sin formato).

IDE: Entorno de desarrollo integrado

Para la comodidad de los desarrolladores se crearon lo que se llaman Entornos de Desarrollo Integrado (en inglés IDE). Un IDE es un software que incluye todo lo necesario para la programación: un compilador (con todos sus programas accesorios), un editor con herramientas que ayudan en la creación de programas, un depurador para buscar errores, etc... Es la solución más completa y recomendada.

Existen multitud de IDE que puedes utilizar. Geany y Code::Blocks anteriormente mencionados son muy recomendables en entornos MS Windows, para Linux tenemos montones de opciones, como el Geany, Anjuta o el Kdevelop.

El primer programa: Hola Mundo

En un alarde de originalidad vamos a hacer nuestro primer programa: hola mundo. Nadie puede llegar muy lejos en el mundo de la programación sin haber empezado su carrera con este original y

funcional programa. Allá va:

#include <stdio.h>

int main() { /* Aquí va el cuerpo del programa */ printf("Hola mundo\n"); return 0; }

Nota: Hay mucha gente que programa en Windows que se queja de que cuando ejecuta el programa no puede ver el resultado. Para evitarlo se puede añadir antes de return 0; la siguiente línea:

system("PAUSE");

Si esto no funciona prueba a añadir getch();

Otra nota: En compiladores MS Windows, para poder usar la función system() debes añadir al principio del fichero la línea:

#include <windows.h>

¿Qué fácil eh? Este programa lo único que hace es sacar por pantalla el mensaje:

Hola mundo

Vamos ahora a comentar el programa línea por línea (Esto no va a ser más que una primera aproximación).

#include <stdio.h>

#include es lo que se llama una directiva. Sirve para indicar al compilador que incluya otro archivo. Cuando en compilador se encuentra con esta directiva la sustituye por el archivo indicado. En este caso es el archivo stdio.h que es donde está definida la función printf, que veremos luego.

int main()

Es la función principal del programa. Todos los programas de C deben tener una función llamada main. Es la que primero se ejecuta. El int (viene de Integer=Entero) que tiene al principio significa que cuando la función main acabe devolverá un número entero. Este valor se suele usar para saber cómo ha terminado el prorama. Normalmente este valor será 0 si todo ha ido bien, o un valor distinto si se ha producido algún error (pero esto lo decidimos nosotros, ya lo veremos). De esta forma si nuestro programa se ejecuta desde otro el programa *padre* sabe como ha finalizado, si ha habido errores o no.

Se puede usar la definición *void main()*, que no necesita devolver ningún valor, pero se recomienda la forma con *int* que es más correcta. Es posible que veas muchos ejemplos que uso *void main* y en los que falta el return 0; del final; el código funciona correctamente pero puede dar un *warning* (un aviso) al compilar dado que no es una práctica correcta.

{

Son las llaves que indican, entre otras cosas, el comienzo de una función; en este caso la función main.

/* Aquí va el cuerpo del programa */

Esto es un comentario, el compilador lo ignorará. Sirve para describir el programa a otros desarrolladores o a nosotros mismos para cuando volvamos a ver el código fuente dentro de un tiempo. Conviene acostumbrarse a comentar los programas pero sin abusar de ellos (ya hablaremos sobre esto más adelante).

Los comentarios van encerrados entre /* y */.

Un comentario puede ocupar más de una línea. Por ejemplo el comentario:

/* Este es un comentario que ocupa dos filas */

es perfectamente válido.

printf("Hola mundo\n");

Aquí es donde por fin el programa hace algo que podemos ver al ejecutarlo. La función printf muestra un mensaje por la pantalla.

Al final del mensaje "Hola mundo" aparece el símbolo \n; este hace que después de imprimir el mensaje se pase a la línea siguiente. Por ejemplo:

printf("Hola mundo\nAdiós mundo");

mostrará:

Hola mundo Adiós mundo

Fíjate en el ";" del final. Es la forma que se usa en C para separar una instrucción de otra. Se pueden poner varias en la misma línea siempre que se separen por el punto y coma.

return 0;

Como he indicado antes el programa al finalizar develve un valor entero. Como en este programa no se pueden producir errores (nunca digas nunca jamás) la salida siempre será 0. La forma de hacer que el programa devuelva un 0 es usando return. Esta línea significa 'finaliza la función main haz que devuelva un 0.

}

...y cerramos llaves con lo que termina el programa. Todos los programas finalizan cuando se llega al final de la función main.

¿Cómo se hace?

Primero debemos crear el código fuente del programa. Para nuestro primer programa el código fuente es el del listado anterior. Arranca tu compilador de C, sea cual sea. Crea un nuevo fichero y copia el código anterior. Llámalo por ejemplo primero.c. Ahora, tenemos que compilar el programa para crear el ejecutable. Si estás usando un IDE busca una opción llamada "compile", o make, build o algo así. Si estamos usando GCC sin IDE tenemos que llamarlo desde la línea de comando:

gcc primero.c -o primero

Nota adicional sobre los comentarios

Los comentarios se pueden poner casi en cualquier parte. Excepto en medio de una instrucción. Por ejemplo lo siguiente no es válido:

pri/* Esto es un comentario */ntf("Hola mundo");

No podemos cortar a printf por en medio, tendríamos un error al compilar. Lo siguiente puede no dar un error, pero es una fea costumbre:

printf(/* Esto es un comentario */ "Hola mundo");

Y por último tenemos:

printf("Hola/* Esto es un comentario */ mundo");

Que no daría error, pero al ejecutar tendríamos:

Hola /* Esto es un comentario */ mundo

porque /* Esto es un comentario */ queda dentro de las comillas y C lo interpreta como texto, no como un comentario.

¿Qué sabemos hacer?

Pues la verdad es que todavía no hemos aprendido mucho. Lo único que podemos hacer es compilar nuestros programas. Pero paciencia, en seguida avanzaremos.

Ejercicios

Busca los errores en este programa:

int main() { /* Aquí va el cuerpo del programa */ Printf("Hola mundo\n"); return 0; }

Solución:

Si lo compilamos obtendremos un error que nos indicará que no hemos definido la función *Printf*. Esto es porque no hemos incluído la dichosa directiva *#include <stdio.h>*. (En algunos compiladores no es necesario incluir esta directiva, pero es una buena costumbre hacerlo).

Si lo corregimos y volvemos a compilar obtendremos un nuevo error. Otra vez nos dice que desconoce *Printf*. Esta vez el problema es el de las mayúsculas que hemos indicado antes. Lo correcto es poner *printf* con minúsculas. Parece una tontería, pero seguro que nos da más de un problema.

Capítulo 2. Mostrando Información por pantalla.

Printf: Imprimir en pantalla

Siempre he creído que cuando empiezas con un nuevo lenguaje suele gustar el ver los resultados, ver que nuestro programa hace *algo*. Por eso creo que el curso debe comenzar con la función ***printf***, que sirve para sacar información por pantalla. Para utilizar la función *printf* en nuestros programas debemos incluir la directiva:

```
#include <stdio.h>
```

al principio de programa. Como hemos visto en el programa hola mundo. Si sólo queremos imprimir una cadena basta con hacer (no olvides el ";" al final):

```
printf( "Cadena" );
```

Esto resultará por pantalla:

```
Cadena
```

Lo que pongamos entre las comillas es lo que vamos a sacar por pantalla. Si volvemos a usar otro *printf*, por ejemplo:

```
#include <stdio.h>
int main() {
    printf( "Cadena" );
    printf( "Segunda" );
    return 0;
}
```

Obtendremos:

```
CadenaSegunda
```

Este ejemplo nos muestra cómo funciona printf. Para escribir en la pantalla se usa un cursor que no vemos. Cuando escribimos algo el cursor va al final del texto. Cuando el texto llega al final de la fila, lo siguiente que pongamos irá a la fila siguiente. Si lo que queremos es sacar cada una en una línea deberemos usar "\n". Es el indicador de retorno de carro. Lo que hace es saltar el cursor de escritura a la línea siguiente:

```
#include <stdio.h>

int main()
{
    printf( "Cadena\n" );
    printf( "Segunda" );
    return 0;
}
```

y tendremos:

```
Cadena
Segunda
```

También podemos poner más de una cadena dentro del printf:

```
printf( "Primera cadena" "Segunda cadena" );
```

Lo que no podemos hacer es meter cosas entre las cadenas:

```
printf( "Primera cadena" texto en medio "Segunda cadena" );
```

esto no es válido. Cuando el compilador intenta interpretar esta sentencia se encuentra *"Primera cadena"* y luego texto en medio, no sabe qué hacer con ello y da un error. Pero ¿qué pasa si queremos imprimir el símbolo " en pantalla? Por ejemplo imaginemos que queremos escribir:

```
Esto es "raro"
```

Si hacemos:

```
printf( "Esto es "raro"" );
```

obtendremos unos cuantos errores. El problema es que el símbolo " se usa para indicar al compilador el comienzo o el final de una cadena. Así que en realidad le estaríamos dando la cadena "Esto es", luego extraño y luego otra cadena vacía "". Pues resulta que *printf* no admite esto y de nuevo tenemos errores.

La solución es usar \". Veamos:

```
printf( "Esto es \"extraño\"" );
```

Esta vez todo irá como la seda. Como vemos la contrabarra | sirve para indicarle al compilador que

escriba caracteres que de otra forma no podríamos. Esta contrabarra se usa en C para indicar al compilador que queremos meter símbolos especiales. Pero ¿Y si lo que queremos es usar | como un carácter normal y poner por ejemplo Hola\Adiós? Pues muy fácil, volvemos a usar |:

```
printf( "Hola\\Adiós" );
```

y esta doble | indica a C que lo que queremos es mostrar una |. He aquí un breve listado de códigos que se pueden imprimir:

Código Nombre Significado \a alert Hace sonar un pitido \b backspace Retroceso \n newline Salta a la línea siguiente (salto de línea) \r carriage return Retorno de carro (similar al anterior) \t horizontal tab Tabulador horizontal \v vertical tab Tabulador vertical \\ backslash Barra invertida \? question mark Signo de interrogación \' single quote Comilla sencilla \" double quote Comilla doble

Es recomendable probarlas para ver realmente lo que significa cada una.

Esto no ha sido mas que una introducción a printf. Luego volveremos sobre ella.

Gotoxy: Posicionando el cursor (requiere conio.h)

Esta función sólo está disponible en compiladores de C que dispongan de la biblioteca <conio.h>, de hecho, en la mayoría de compiladores para Linux no viene instalada por defecto. No debería usarse aunque se menciona aquí porque en muchos cursos de formación profesional y en universidades aún se usa. Hemos visto que cuando usamos printf se escribe en la posición actual del cursor y se mueve el cursor al final de la cadena que hemos escrito.

Vale, pero ¿qué pasa cuando queremos escribir en una posición determinada de la pantalla? La solución está en la función gotoxy. Supongamos que queremos escribir *Hola* en la fila 10, columna 20 de la pantalla:

```
#include <stdio.h>
#include <conio.h>

int main()
{
    gotoxy( 20, 10 );
    printf( "Hola" );
    return 0;
}
```

para usar gotoxy hay que incluir la biblioteca conio.h).

Fíjate que primero se pone la columna (x) y luego la fila (y). La esquina superior izquierda es la posición (1, 1).

Clrscr: Borrar la pantalla (requiere conio.h)

Ahora ya sólo nos falta saber cómo se borra la pantalla. Pues es tan fácil como usar:

```
clrscr();
```

(clear screen, borrar pantalla).

Esta función nó solo borra la pantalla, sino que además sitúa el cursor en la posición (1, 1), en la esquina superior izquierda.

```
#include <stdio.h>
#include <conio.h>

int main()
{
    clrscr();
    printf( "Hola" );
    return 0;
}
```

Este método sólo vale para compiladores que incluyan el fichero conio.h. Si tu sistema no lo tiene puedes consultar la sección siguiente.

Borrar la pantalla (otros métodos)

Existen otras formas de borrar la pantalla aparte de usar conio.h.

Si usas DOS:

system ("cls"); //Para DOS

Si usas Linux:

system ("clear"); // Para Linux

Otra forma válida para ambos sistemas:

char a[5]={27,*[*,*2*,*J*,0}; /* Para ambos (en DOS cargando antes ansi.sys) */ printf("%s",a);

¿Qué sabemos hacer?

Bueno, ya hemos aprendido a sacar información por pantalla. Si quieres puedes practicar con las instrucciones printf, gotoxy y clrscr. Lo que hemos visto hasta ahora no tiene mucho secreto, pero ya veremos cómo la función printf tiene mayor complejidad.

Ejercicios

Ejercicio 1: Busca los errores en el programa (este programa usa conio.h, pero aunque tu compilador no la incluya aprenderás algo con este ejercicio).

```
#include <stdio.h>
int main()
{
    ClrScr();
    gotoxy( 10, 10 )
    printf( Estoy en la fila 10 columna 10 );
    return 0;
}
```

Solución:

ClrScr está mal escrito, debe ponerse todo en minúsculas, recordemos una vez más que el C diferencia las mayúsculas de las minúsculas. Además no hemos incluído la directiva #include <conio.h>, que necesitamos para usar clrscr() y gotoxy(). Tampoco hemos puesto el punto y coma (;) después del gotoxy(10, 10). Después de cada instrucción debe ir un punto y coma. El último fallo es que el texto del printf no lo hemos puesto entre comillas. Lo correcto sería: printf("Estoy en la fila 10 columna 10");

Ejercicio 2: Escribe un programa que borre la pantalla y escriba en la primera línea tu nombre y en la segunda tu apellido:

Solución:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    clrscr();
    printf( "Gorka\n" );
    printf( "Urrutia" );
    return 0;
}
```

También se podía haber hecho todo de golpe:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    clrscr();
    printf( "Gorka\nUrrutia" );
    return 0;
}
```

Ejercicio 3: Escribe un programa que borre la pantalla y muestre el texto "estoy aqui" en la fila 10, columna 20 de la pantalla.

Solución:

```
#include <stdio.h>
#include <conio.h>
int main() {
    clrscr();
    gotoxy( 20, 10 );
    printf( "Estoy aqui" );
    return 0;
}
```