Referencia bibliográfica

García-Bermejo Giner, José R. (2008). *Programación estructurada en C*. Madrid: Pearson Educación. Pp. 1-28.

Programación estructurada en C

José R. García-Bermejo Giner

PEARSON
Prentice Hall

Programación estructurada en C

José R. García-Bermejo Giner
Departamento de Informática y Automática
Universidad de Salamanca

PEARSON
Prentice
Hall

Madrid • México • Santa Fe de Bogotá • Buenos Aires • Caracas • Lima • Montevideo
San Juan • San José • Santiago • São Paulo • White Plains

Datos de catalogación bibliográfica

PROGRAMACIÓN ESTRUCTURADA EN C
García-Bermejo Giner, José R.

PEARSON EDUCACIÓN, S.A., Madrid, 2008
ISBN: 978-84-8322-423-6

Materia: Informática, 004

Formato: 195 × 250 mm Páginas: 296

PROGRAMACIÓN ESTRUCTURADA EN C
García-Bermejo Giner, José R.

ISBN: 978-84-8322-423-6

Depósito Legal: M. 6.091-2008

PEARSON PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

Equipo editorial:
Editor: Miguel Martín-Romo
Técnico editorial: Marta Caicoya
Equipo de producción:
Director: José A. Clares
Técnico: José A. Hernán
Diseño de cubierta: Equipo de diseño de PEARSON EDUCACIÓN, S. A.
Composición: JOSUR TRATAMIENTOS DE TEXTOS, S.L.
Impreso por: Lavel, S. A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

Índice

CAPÍTULO 1

C: Un lenguaje de programación

✦SUMARIO✦

Introducción

(Véase http://maxus.fis.usal.es/FICHAS_C.WEB/00xx_PAGS/0000.html)

La programación es una actividad eminentemente práctica, vaya esto por delante. Los conceptos teóricos pueden ser difíciles de aprehender, pero no son largos, ni constituyen el núcleo de la programación. Más bien, solo es posible comprender realmente las descripciones teóricas mediante un uso extenso e intenso de la práctica. Por esta razón, presentaremos brevemente los conceptos teóricos, apoyándonos en maxus para aportar conocimientos adicionales, y pasaremos con la mayor rapidez posible a la práctica.

1.1. ¿Cómo se estudia este curso?

La forma de asimilar los conocimientos que aporta este curso es similar a la de un curso de matemáticas; a fin de cuentas, la programación es la expresión de conceptos en un lenguaje que se

asemeja mucho al algebraico por su precisión y falta de ambigüedad. El desarrollo y estudio de los conceptos que se muestran tiene siempre tres fases:

a) Comprensión teórica del concepto, que se consigue estudiando detalladamente uno o más ejemplos resueltos y procurando comprender cómo funcionan. Estos ejemplos deben ser compilados y ejecutados para verificar su corrección, esto es, que hacen lo que deben hacer. Al haber sido probados en distintas plataformas, su ejecución no debe plantear problemas; si los hubiera, o bien no se ha escrito correctamente el programa, o no se ha instalado correctamente el entorno.

b) Constatación de haber comprendido los conceptos expuestos. En esta fase, es preciso cerrar el libro, poner en marcha el entorno de programación seleccionado y rehacer los ejercicios vistos en la primera fase. Si falla algo, es preciso determinar qué fue y subsanar el error. Esta fase no puede darse por concluida mientras no sea posible alcanzar resultados correctos aplicando los conceptos expuestos, aunque muy posiblemente se llegue a una solución (a un programa, esto es) que sea distinta de la expuesta en el libro.

c) Realización de ejercicios adicionales (tanto en el libro como en maxus se proponen numerosos ejercicios) para desarrollar y afianzar los conceptos estudiados. Esta fase es "abierta"; finalizará cuando se considere que se ha comprendido correctamente el concepto estudiado.

Téngase en cuenta que la programación tiene una estructura que podríamos denominar "circular", lo cual quiere decir que la construcción de programas suele requerir todos los aspectos del lenguaje. Una vez más, aparece el carácter matemático de la programación: no se puede olvidar la suma y esperar que en un problema aparezcan únicamente multiplicaciones; lo normal es que el programa haga uso de muchos de los recursos del lenguaje.

Téngase en cuenta también que la primera dificultad que se hallará en el estudio de la programación es la sintaxis del lenguaje. Esto es, hay que aprender dónde se precisa una coma, y dónde hace falta un punto y coma, y los nombres exactos de unas cuantas palabras mediante las cuales se construyen todos los programas. Estas reglas se memorizan fácilmente con la práctica, pero no son en modo alguno el aspecto más difícil ni el más importante de la programación.

El verdadero núcleo de la programación, sin embargo, lo constituyen dos elementos bastante alejados de la sintaxis:

— Los algoritmos, y

— Las estructuras de datos.

Los algoritmos son los métodos que se emplean para obtener resultados a partir de los datos disponibles. Un algoritmo bueno o malo es más importante que una buena o mala implementación de ese algoritmo. Esto se constatará con la práctica.

Las estructuras de datos son los artificios del lenguaje mediante los cuales se representa la información. Estos mecanismos van desde sencillos (listas y tablas, por ejemplo) hasta otros como las pilas, colas y árboles cuyo estudio es tan complejo que posee una disciplina propia, la de Estructuras de Datos.

Consúltese "Algoritmos + Estructuras de datos = Programas", escrito por Niklaus Wirth.

1.2. Programas y lenguajes de programación

Un programa es un texto, una colección de líneas que se escriben empleando un programa editor. Ese texto se llama código fuente, y no puede ser (en general) ejecutado por la computadora.

El código fuente se escribe mediante un editor en un fichero de texto y queda almacenado en el disco de nuestra computadora en un fichero que va a tener la extensión .c. Ese fichero de texto que reside en nuestra computadora se podría llamar, por ejemplo, HolaMundo.c.

Los programas son la descripción de un algoritmo o conjunto de algoritmos. Ciertos algoritmos, como el de impresión en pantalla o del de lectura de datos del teclado, son tan importantes que el propio lenguaje los ofrece ya encapsulados, y se pueden invocar mediante una sola palabra. Otros, los más, deben ser descritos por el programador en forma de líneas de texto. Obsérvese que nada impide que esa descripción se apoye en otros algoritmos (encapsulados, como decimos) ya existentes.

Consideremos el texto siguiente (*véase http://cm.bell-labs.com/cm/cs/cbook/*):

HolaMundo.c

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    printf("%s", "Hola, Mundo!\n");
    printf("%s", "Hoy es Jueves\n");
    return 0;
}
```

Esto es un ejemplo de "programa". Lo que se ve sobre estas líneas es un "código fuente", un pequeño programa escrito en un lenguaje de programación que se llama C. En este caso concreto, el programa se ha escrito con ayuda de un editor llamado BBEdit. Existe una versión gratuita (TextWrangler) en la página *www.barebones.com.* Evidentemente, cualquier editor de texto sirve para escribir programas, aunque las facilidades que ofrecen los editores especializados los hacen más recomendables que un editor de textos de propósito general.

Un programa es una colección de instrucciones que, debidamente traducidas, serán ejecutadas por una computadora para realizar una tarea. En otras palabras, un programa es la descripción de un algoritmo escrita en un cierto lenguaje de programación.

La tarea que se pretende realizar, en este caso, consiste en escribir el mensaje siguiente en la pantalla:

```
Hola, Mundo!

Hoy es Jueves
```

Para llevar a cabo esta tarea vamos a emplear el algoritmo de escritura que se ofrece (encapsulado) en el lenguaje de programación C. Este algoritmo se invoca escribiendo la palabra

`printf` (print formatted o escribir con formato), que va acompañada por lo que se quiere escribir, entre paréntesis. Obsérvese la presencia de ciertos fragmentos de código anteriores (`int main...`) y posteriores (`return ...`) cuya funcionalidad no se comentará de momento.

1.3. ¿Cómo es realmente un programa?

Consideremos ahora otro texto de funcionalidad idéntica:

HolaMundo.java

```
public class HolaMundo {
    public static void main(String[] args)  {
        System.out.printf("Hola, Mundo!%n");
        System.out.printf("Hoy es Jueves%n");
    }
}
```

Esto también es un ejemplo de programa; de hecho, realiza la misma tarea que el anterior. El texto impreso es el mismo, y la convención seguida para saltar a la línea siguiente es parecida. Sin embargo, se aprecian diferencias entre ambos textos. Esto se debe a que el primer programa está escrito en un lenguaje llamado C (*véase http://en.wikipedia.org/wiki/C_programming_language*), mientras que el otro está escrito en otro lenguaje distinto, llamado Java (*véase http://en.wikipedia.org/wiki/Java_programming_language*).

Los programas, como hemos dicho, se escriben en ficheros con formato de texto. Los ficheros en que se escriben los programas no tienen la extensión .txt, sino otras (como .c o .java) que denotan el lenguaje de programación empleado para escribir el programa.

Un lenguaje (de programación) es una colección de reglas que especifican la forma en que deben escribirse sentencias para que sea posible traducirlas a otras comprensibles para el procesador en que se ejecutarán. Esas sentencias sirven para describir la colección de algoritmos mediante los cuales va a resolver una cierta tarea la computadora.

Aún hay más: estos programas no se pueden utilizar directamente. La computadora no entiende estas expresiones, aunque están escritas en un lenguaje "de alto nivel", lo cual significa que es comprensible para operadores humanos. En efecto, puede llegar a entenderse el propósito del programa en ambos casos. Esto se debe a que la computadora tiene otra visión del mundo.

¿Por qué no se puede ejecutar directamente el código fuente?

Simplemente, porque las computadoras entienden únicamente instrucciones que en nada se parecen a las que se emplean en estos lenguajes de programación (denominados de alto nivel).

Las instrucciones aceptables para la computadora (lenguaje máquina) no son fácilmente interpretables para un operador humano. Así que tenemos una situación en que aparecen dos len-

guajes: el lenguaje de programación (de alto nivel), en que escribiremos nosotros el programa, y el lenguaje máquina, propio de la máquina. Obviamente, hace falta traducir el texto creado por nosotros (ese fichero de texto que contiene el código fuente) a instrucciones en código máquina. Esa tarea la va a realizar un programa denominado compilador.

Veamos un par de ejemplos concretos. El resultado de traducir HolaMundo.c al lenguaje ensamblador (un precursor del lenguaje máquina) para un procesador de la familia PowerPC es como sigue:

```
        .section __TEXT,__text,regular,pure_instructions
        .section __TEXT,__picsymbolstub1,symbol_stubs,pure_instruc-
            tions,32
        .machine ppc
        .cstring
        .align 2
LC0:
        .ascii "Hola, Mundo!\12\0"
        .align 2
LC1:
        .ascii "Hoy es Jueves\12\0"
        .text
        .align 2
        .globl _main
_main:
        mflr r0
        stmw r30,-8(r1)
        stw r0,8(r1)
        stwu r1,-80(r1)
        mr r30,r1
        bcl 20,31,"L00000000001$pb"
"L00000000001$pb":
        mflr r31
        stw r3,104(r30)
        stw r4,108(r30)
        addis r2,r31,ha16(LC0-"L00000000001$pb")
        la r3,lo16(LC0-"L00000000001$pb")(r2)
        bl L_printf$LDBLStub$stub
        addis r2,r31,ha16(LC1-"L00000000001$pb")
        la r3,lo16(LC1-"L00000000001$pb")(r2)
        bl L_printf$LDBLStub$stub
        li r0,0
        mr r3,r0
        lwz r1,0(r1)
        lwz r0,8(r1)
        mtlr r0
        lmw r30,-8(r1)
        blr
        .section __TEXT,__picsymbolstub1,symbol_stubs,pure_instruc-
            tions,32
        .align 5
L_printf$LDBLStub$stub:
```

```
        .indirect_symbol _printf$LDBLStub
        mflr r0
        bcl 20,31,"L00000000001$spb"
"L00000000001$spb":
        mflr r11
        addis r11,r11,ha16(L_printf$LDBLStub$lazy_ptr-
            "L00000000001$spb")
        mtlr r0
        lwzu r12,lo16(L_printf$LDBLStub$lazy_ptr-
            "L00000000001$spb")(r11)
        mtctr r12
        bctr
        .lazy_symbol_pointer
L_printf$LDBLStub$lazy_ptr:
        .indirect_symbol _printf$LDBLStub
        .long   dyld_stub_binding_helper
        .subsections_via_symbols
```

No resulta especialmente comprensible, en verdad. Y esto está escrito en lenguaje ensamblador, que (teóricamente) es comprensible para seres humanos. El mismo programa HolaMundo.c, traducido a ensamblador para Intel, tiene este otro aspecto:

```
  .cstring
LC0:
  .ascii "Hello, Mundo!\0"
LC1:
  .ascii "Hoy es Jueves!\0"
  .text
.globl _main
_main:
  pushl %ebp
  movl  %esp, %ebp
  pushl %ebx
  subl  $20, %esp
  call  ___i686.get_pc_thunk.bx
"L00000000001$pb":
  leal  LC0-"L00000000001$pb"(%ebx), %eax
  movl  %eax, (%esp)
  call  L_puts$stub
  leal  LC1-"L00000000001$pb"(%ebx), %eax
  movl  %eax, (%esp)
  call  L_puts$stub
  movl  $0, %eax
  addl  $20, %esp
  popl  %ebx
  popl  %ebp
  ret
  .section __IM
      PORT,__jump_table,symbol_stubs,self_modifying_code+pure_ins-
      tructions,5
L_puts$stub:
```

```
  .indirect_symbol _puts
  hlt ; hlt ; hlt ; hlt ; hlt
  .subsections_via_symbols
  .section __TEXT,__textcoal_nt,coalesced,pure_instructions
.weak_definition ___i686.get_pc_thunk.bx
.private_extern ___i686.get_pc_thunk.bx
___i686.get_pc_thunk.bx:
  movl (%esp), %ebx
  ret
```

Mostramos este código, que no se va a ver ni a utilizar en este curso, para poner de manifiesto cosas importantes:

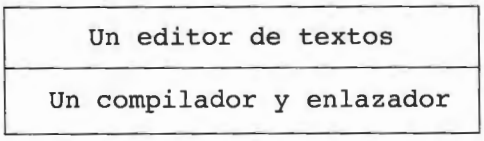
a) El código en ensamblador es más largo que el código en C, debido a que las instrucciones de programación empleadas en C se convierten en muchas más instrucciones de programación en ensamblador. Por tanto, requerirá mucho más tiempo para su elaboración.

b) El código en ensamblador es distinto para distintos procesadores. Esto se debe a que cada fabricante dota al procesador de distintos juegos de instrucciones. Aunque algunas instrucciones son comunes, sus nombres y expresiones no lo son. Se dice entonces que el lenguaje ensamblador no es compatible entre procesadores.

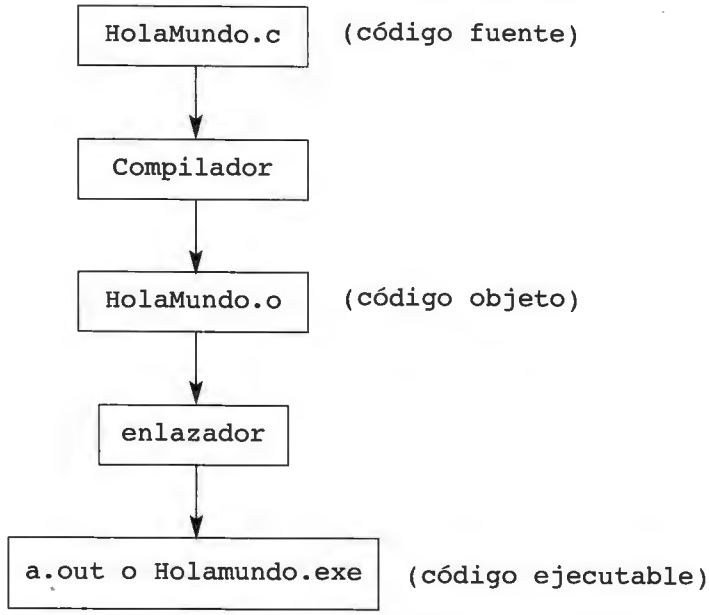
Se observa entonces una idea interesante: un mismo código fuente escrito en C se puede compilar para dos procesadores distintos. El resultado será un programa ejecutable en distintos procesadores, incluso en distintos sistemas operativos, pero con igual funcionalidad. Hará lo mismo. Esto es sumamente interesante desde el punto de vista de la creación de programas, puesto que sugiere la posibilidad de tener acceso a distintas plataformas (distintos mercados) con un mismo programa.

1.4. Proceso de creación de un programa ejecutable

Volviendo a nuestro programa, sabemos que un programa escrito en código fuente en C no se puede ejecutar sin más. Si hacemos doble clic en HolaMundo.c, se abrirá el fichero con un editor de textos. Y si acudimos a la línea de órdenes y tecleamos el nombre de este fichero, tampoco conseguiremos gran cosa: el sistema operativo nos indicará que HolaMundo.c no se reconoce como una instrucción. Lo que ocurre es que para escribir programas se precisan al menos estas herramientas:



El editor de textos permite que el operador humano escriba instrucciones como las vistas anteriormente (HolaMundo.c), y el compilador y el enlazador se encargarán de generar el código ejecutable; el esquema siguiente muestra el proceso:



El proceso de compilación, mediante el cual se pasa de un fichero de texto (código fuente) a un fichero ejecutable, puede realizarse empleando programas basados en la línea de órdenes, y también programas basados en una interfaz gráfica de usuario o IDE.

Las imágenes siguientes muestran paso a paso el proceso anterior, realizado a través de una terminal de UNIX.

Figura 1.1. Un fichero de código fuente.

Figura 1.2. Un fichero de código fuente y el correspondiente fichero de código objeto.

Figura 1.3. Estado final: código fuente y código ejecutable.

Téngase en cuenta que el proceso normal consiste en emplear la instrucción `cc HolaMundo.c`, que genera directamente el código ejecutable a partir del código fuente. Es como saltar de la primera figura a la última, sin que resultara visible el código objeto que se crea en la fase intermedia.

1.5. Descarga e instalación de un IDE

Los entornos integrados de desarrollo o IDE coordinan en un único paquete el editor de textos, el compilador, el enlazador y un depurador, como mínimo. Además, se encargan de organizar los distintos ficheros de código fuente de que constan normalmente los programas, agrupándolos en una estructura denominada "proyecto" que aglutina todos los elementos necesarios para la generación del programa ejecutable final.

El lenguaje C, objeto de este libro, está disponible en las principales plataformas del mercado. Existen herramientas de tipo IDE, gratuitas, que se pueden descargar en Internet, o se adjuntan directamente en los discos de instalación del sistema operativo seleccionado. Describiremos a continuación la forma de obtener estas herramientas para tres plataformas de amplia difusión.

1.5.1. Linux

Este sistema operativo, basado en UNIX, ofrece las herramientas necesarias para compilar programas en C sea cual fuere la distribución seleccionada. Salvo que se indique lo contrario, se instalan las herramientas de compilación de forma predeterminada, para que así sea posible instalar otros paquetes. Para comprobar la existencia de un compilador de C, basta abrir una terminal y teclear:

```
$ gcc -v
```

El sistema mostrará la versión de GCC presente en el sistema. Si la terminal indica que gcc no se reconoce como una orden válida, indicando "command not found" o un mensaje equivalente, será preciso utilizar las herramientas de instalación de paquetes que ofrezca la distribución que se haya instalado.

El IDE KDevelop permite desarrollar cómodamente aplicaciones escritas en C, y se puede obtener gratuitamente en la siguiente página:

http://www.kdevelop.org

Una posibilidad interesante que ofrece Linux es el uso de una distribución basada en un CD "vivo", que permite arrancar directamente desde él, sin necesidad de modificar el sistema operativo instalado en nuestra computadora. Por ejemplo, considérese Ubuntu, que se puede obtener de forma gratuita en la dirección:

http://www.ubuntu.com/

http://www.ubuntu-es.org/index.php?q=ubuntu/conseguir

Y también se puede considerar la posibilidad de utilizar Knoppix:

http://www.knoppix.com

que es igualmente un "CD vivo" desde el cual se arranca un sistema operativo de tipo Linux sin necesidad de efectuar una instalación en nuestra computadora.

Aun cuando no se trata de una instalación definitiva, el sistema operativo que se ofrece permitirá al alumno compilar programas y ejercitarse cómodamente. Si se opta por instalar alguna

versión de Linux, se pueden conseguir los discos de instalación de muchas distribuciones de forma también gratuita. Considérese, por ejemplo, la página siguiente:

http://www.hungrypenguin.net/downloads.html

en la que se hallarán enlaces para descargar diferentes versiones de Linux.

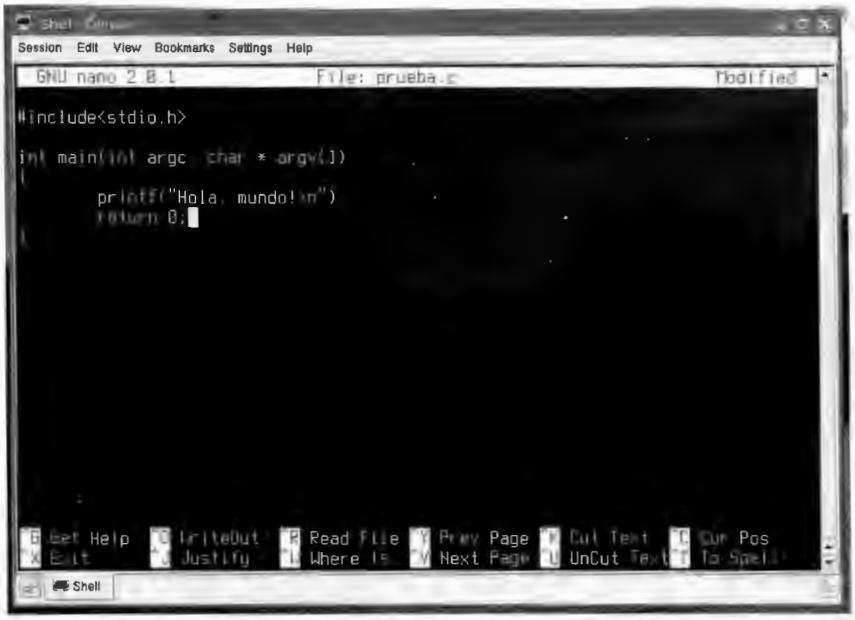


Figura 1.3.1. Creación de un programa desde Knoppix.

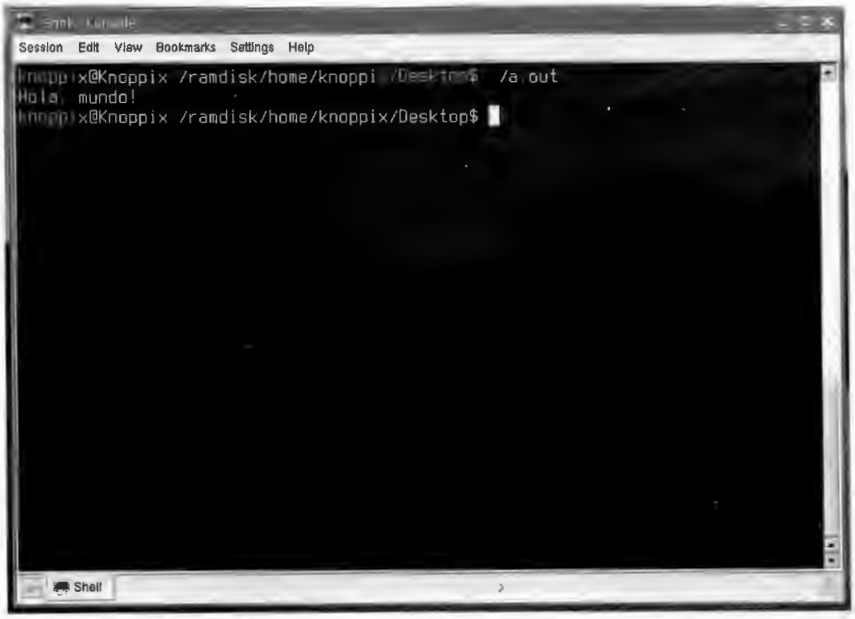


Figura 1.3.2. Ejecución de un programa desde Knoppix.

1.5.2. Mac OS X

Se trata de un sistema operativo basado en UNIX, con especial atención a la facilidad de manejo e instalación. Las herramientas de desarrollo se proporcionan junto con el sistema operativo, pero no forman parte de la instalación predeterminada.

Se puede verificar la disponibilidad del compilador de C abriendo una terminal (/Aplicaciones/Utilidades/Terminal) y tecleando:

```
$ cc -v
```

El sistema mostrará la versión de GCC presente en el sistema. En la actualidad se ofrece la versión 2.4 de XCode, que incluye una versión 4.x de GCC (aunque la versión 3.0 estará disponible cuando este libro sea publicado).

Si la terminal indicase que la orden cc no es conocida, será preciso recurrir al DVD del sistema operativo para instalar las herramientas de desarrollo. El DVD contiene un instalador denominado Developer Tools, que permite añadir al sistema operativo básico una extensa colección de herramientas de desarrollo. Se recomienda aceptar la instalación predeterminada. También es posible descargar gratuitamente las últimas versiones de estas herramientas en *http://connect.apple.com*; requiere registro pero es gratuito.

El IDE XCode es un entorno integrado que permite crear proyectos de distintos tipos; en particular, la variante Standard Tool corresponde a un programa en C estándar adecuado para su ejecución en una terminal.

1.5.3. Windows

Las herramientas de desarrollo no forman parte de la instalación estándar. Existe un gran número de entornos de desarrollo; de entre ellos consideramos una posibilidad: es el denominado DevCpp, que se puede descargar en la dirección

http://www.bloodshed.net

Este entorno es compacto, permite crear fácilmente programas en C o C++, y su instalación y manejo es simple. Resulta muy adecuado para hacer nuestras primeras armas en C. En la actualidad, se ofrece la versión 5 beta, que contiene la versión 3.x de GCC.

Otro entorno de desarrollo agradable es Eclipse, cuya página principal en Internet se hallará en la siguiente dirección:

http://www.eclipse.org

La página de descarga del entorno para C/C++ es la siguiente:

http://www.eclipse.org/downloads/

Este entorno admite la instalación de perspectivas adecuadas para trabajar en C/C++, y también en distintas versiones de Java. Es más potente que el anterior, y por ende tiene una curva de aprendizaje más elevada.

Instalación y actualización de DevCpp

El proceso de instalación es sencillo, pero véanse las imágenes siguientes, en las cuales se detalla el proceso.

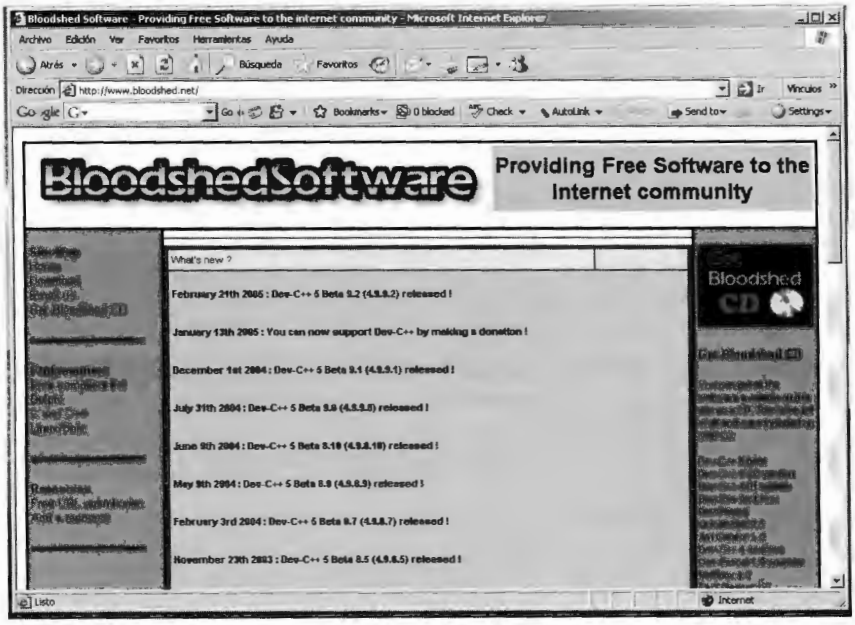


Figura 1.4. Sitio web de DevCpp, situado en *http://www.bloodshed.net.*

Figura 1.5. Página de descargas de DevCpp, situada en *http://www.bloodshed.net/dev/devcpp.**html.***

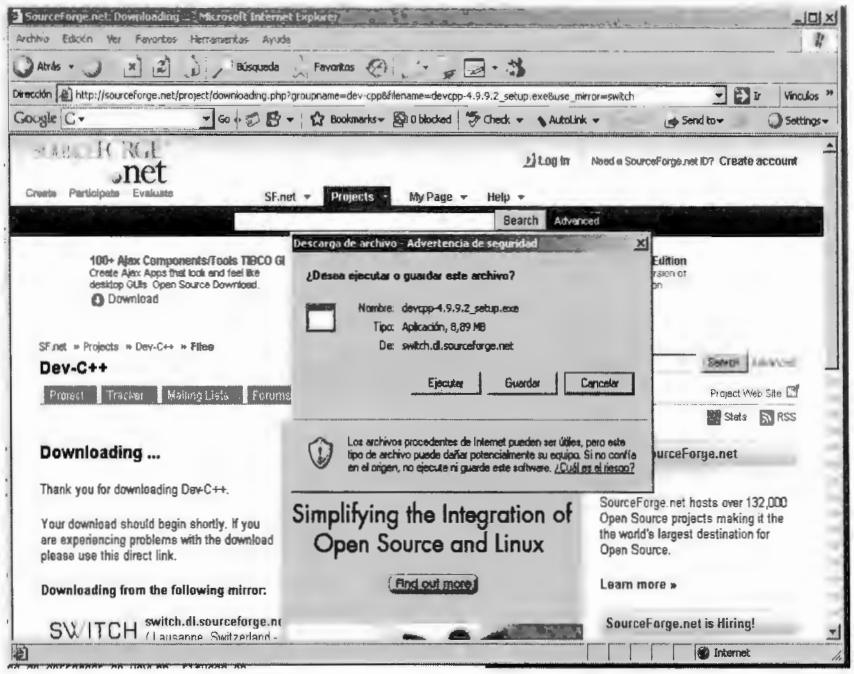


Figura 1.6. Página de descarga de SourceForge. Obsérvese que se ha descargado el bloque formado por DevCpp y el entorno de ejecución. De otro modo no funcionará el programa. El fichero de instalación ocupa 8.98 Mb para la versión 4.9.9.2.

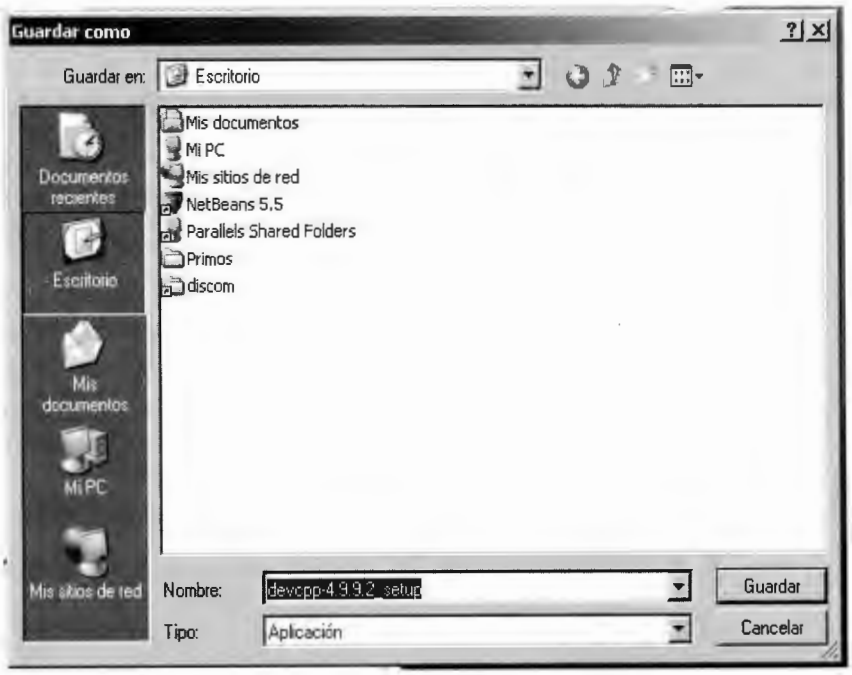


Figura 1.7. Se almacena el instalador en el Escritorio, para después encontrarlo con facilidad. Una vez descargado, se ejecuta el instalador.

Figura 1.8.a)

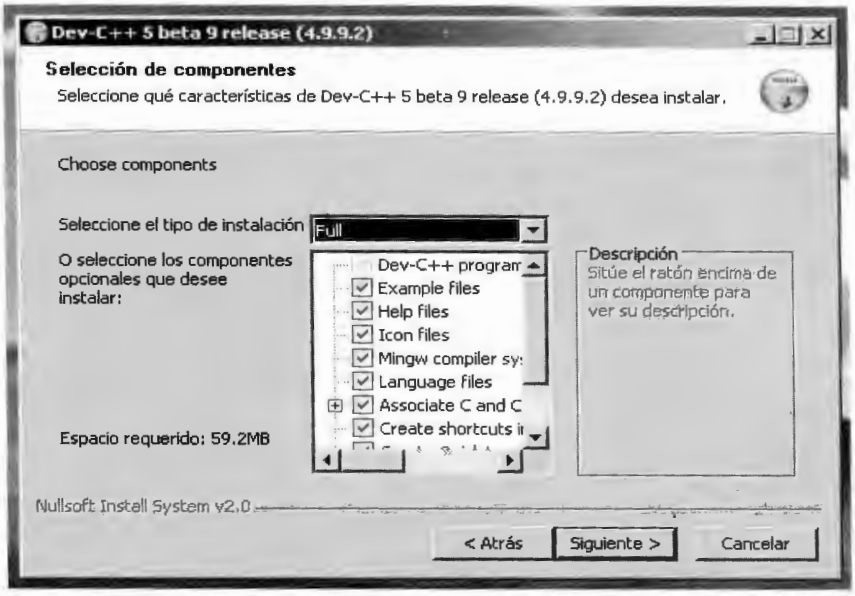


Figura 1.8.b)

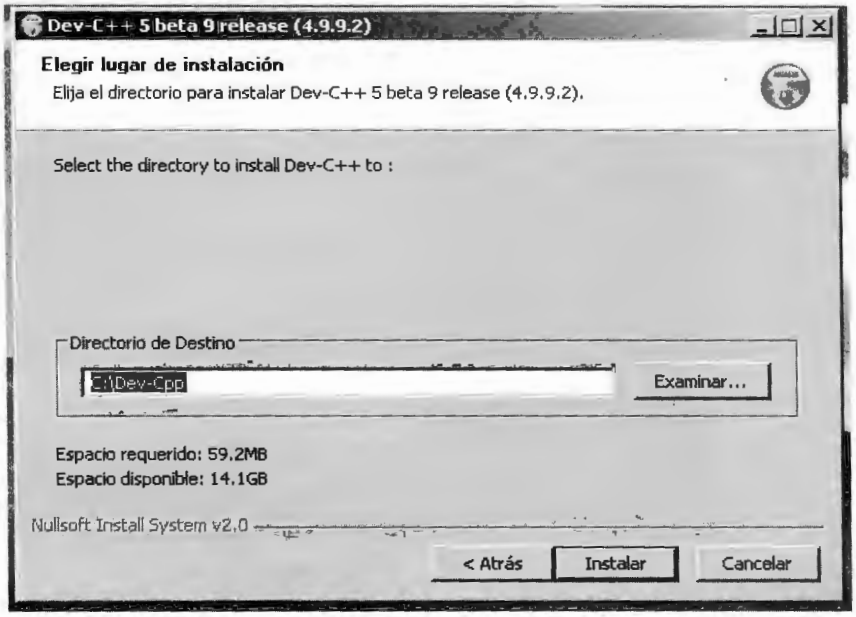


Figura 1.8.c)

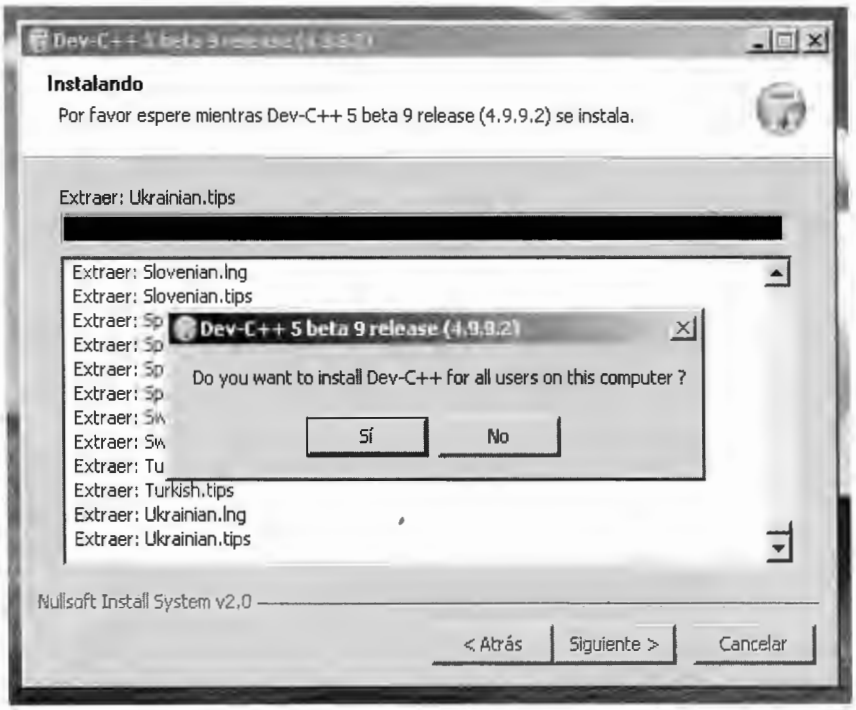


Figura 1.8.d)

Figura 1.8.e)

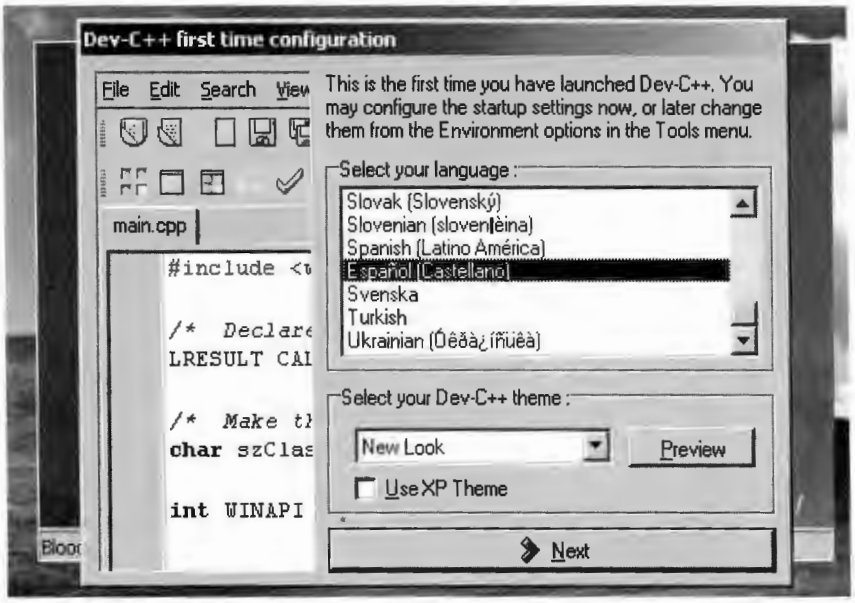


Figura 1.8.f)

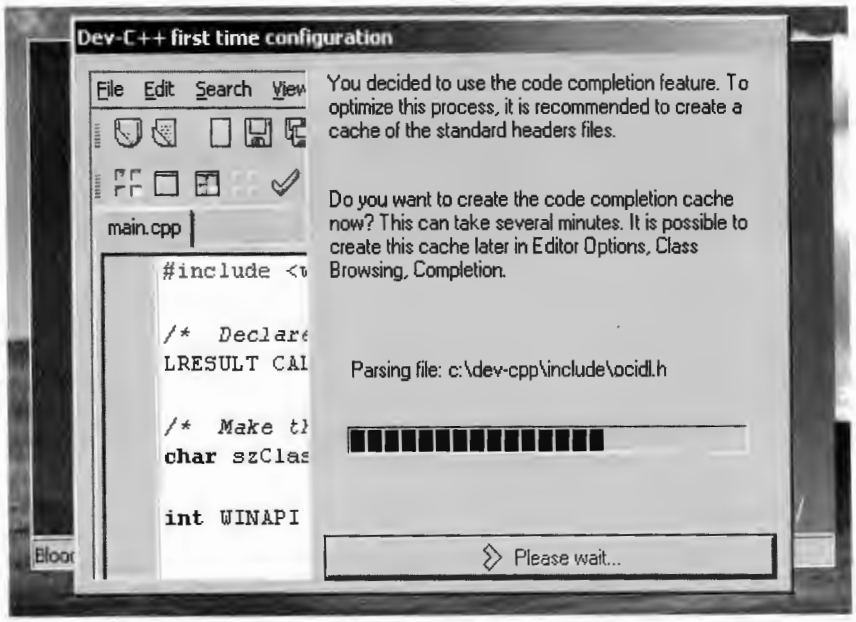


Figura 1.8.g). Se acepta la instalación por omisión propuesta por el instalador.

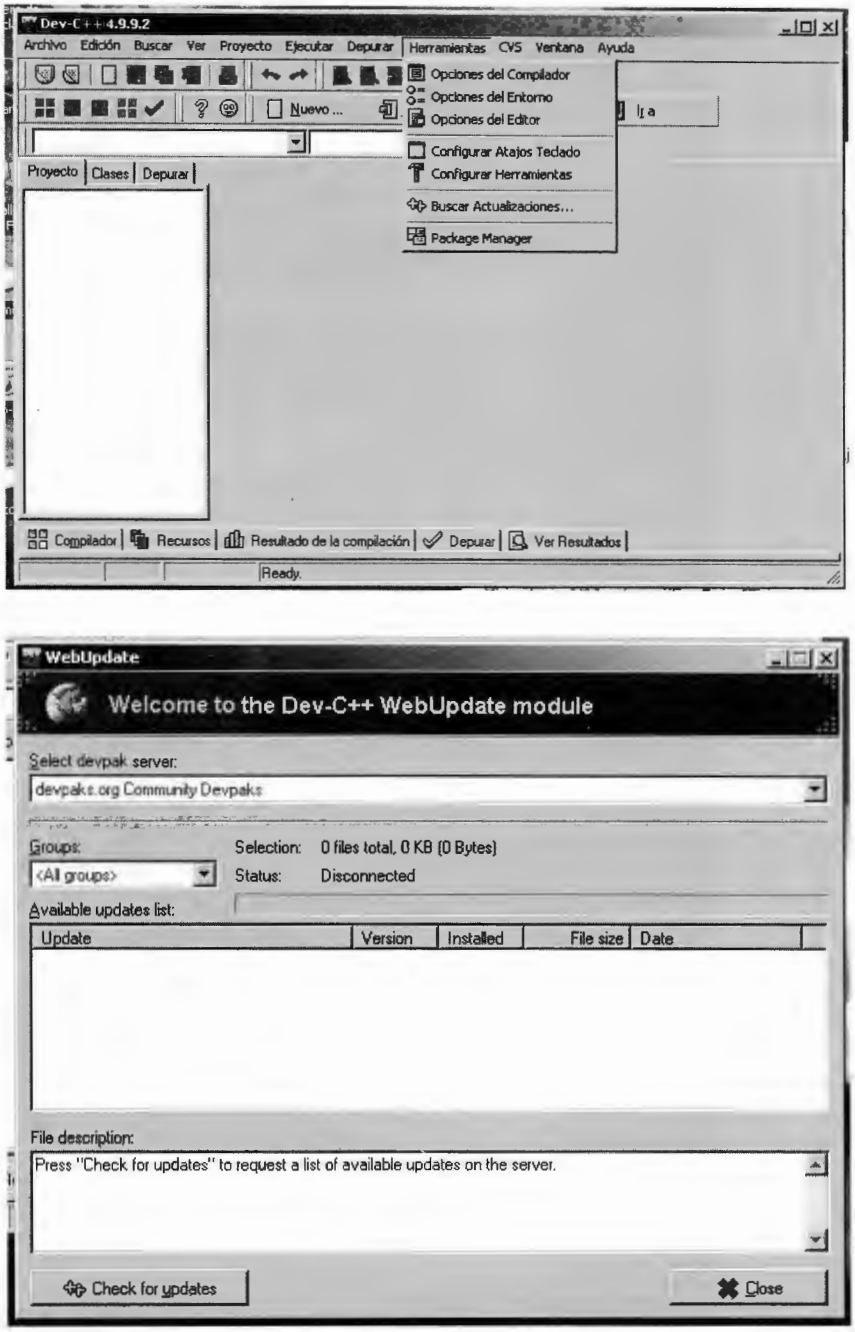


Figura 1.9. Se buscan e instalan las posibles actualizaciones.

Si no es posible conectar con PlanetMirror, será preciso utilizar el servidor de actualizaciones secundario, devpaks.org. Esto plantea un problema, porque no hay garantías de seguridad en este servidor. El modo de resolver este problema consiste en instalar una actualización de la colección de servidores que ofrecen actualizaciones. Esto se hace descargando la lista de actualizaciones de devpaks.org e instalando *únicamente* la actualización denominada WebUpdate Mirrors List, como se ve en la figura siguiente.

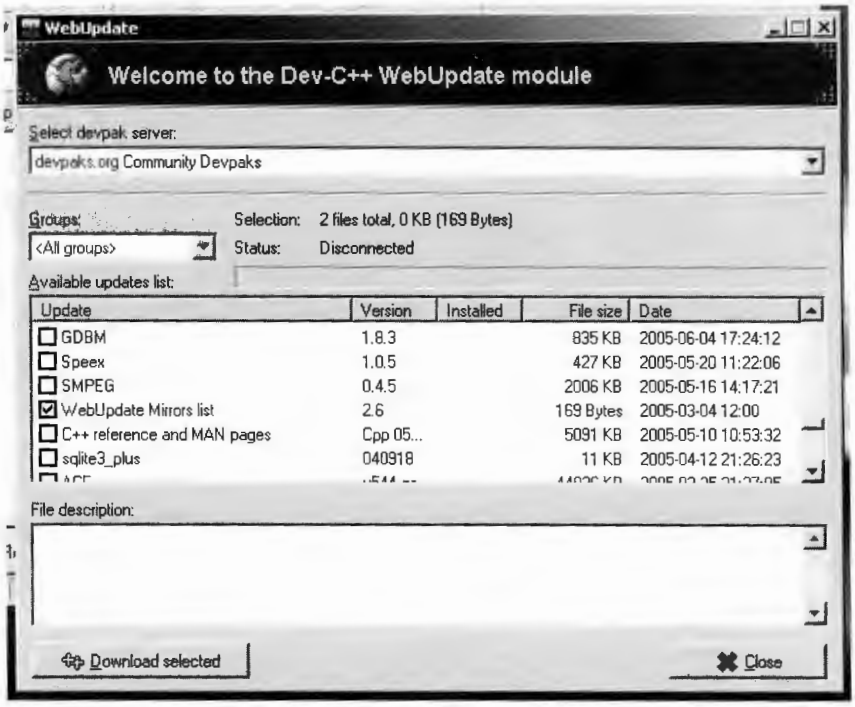


Figura 1.10. Instalación de la nueva lista de servidores de actualizaciones.

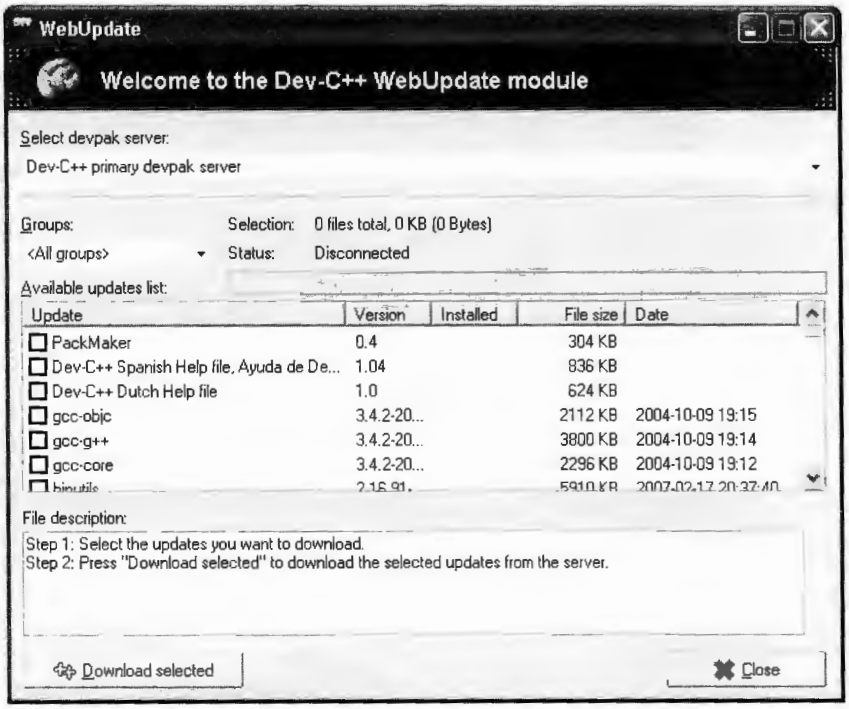


Figura 1.11. En esa nueva lista aparece el Dev-C++ primary devpak server. Es conveniente no instalar todas las posibles actualizaciones, sino tan solo los grupos de Application y Dev-C++ MinGW System, como se ve en la figura siguiente.

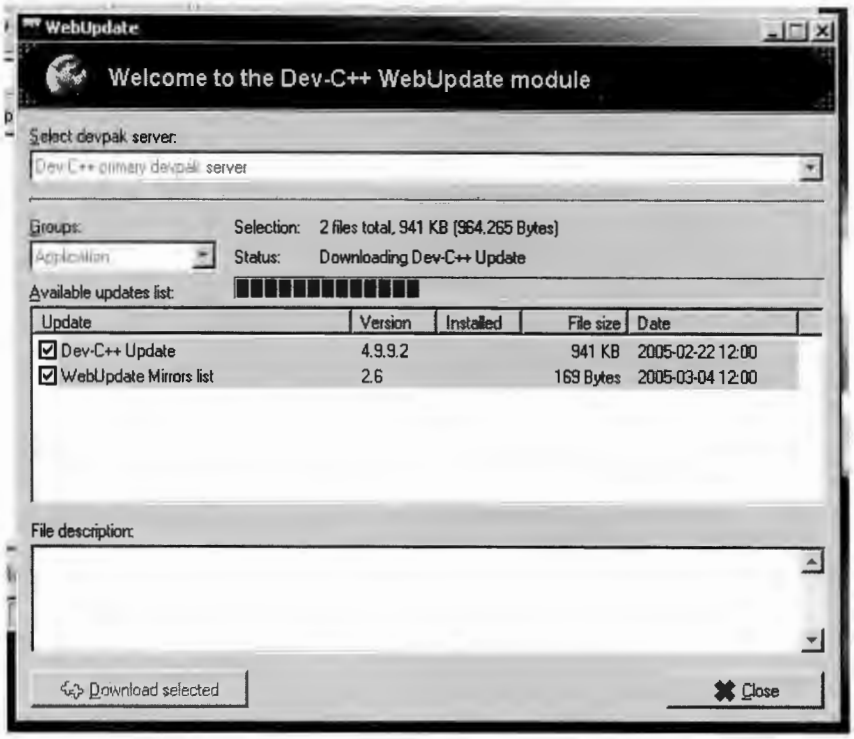


Figura 1.12.a)

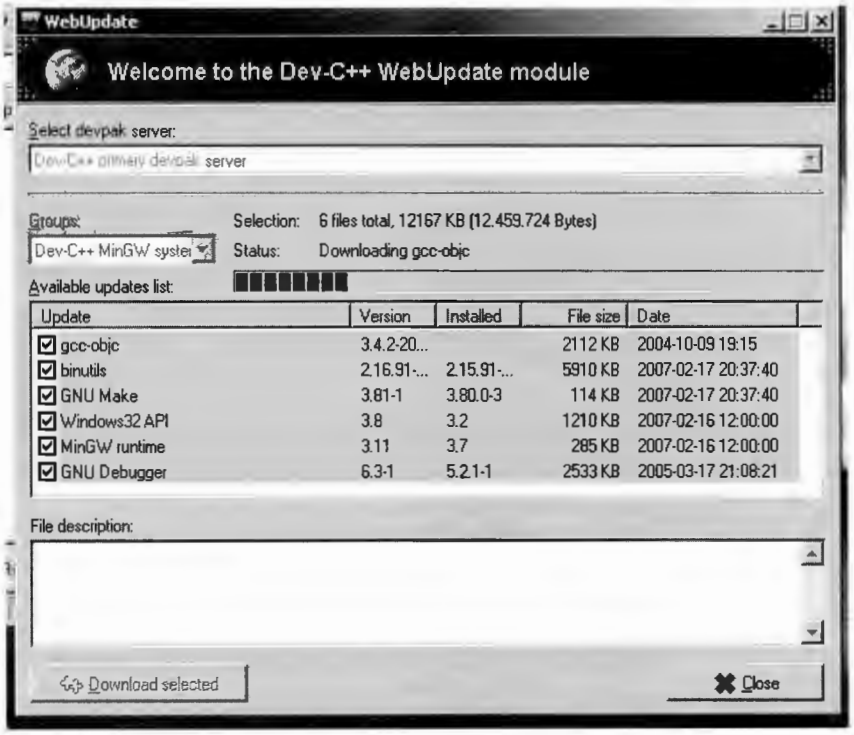


Figura 1.12.b)

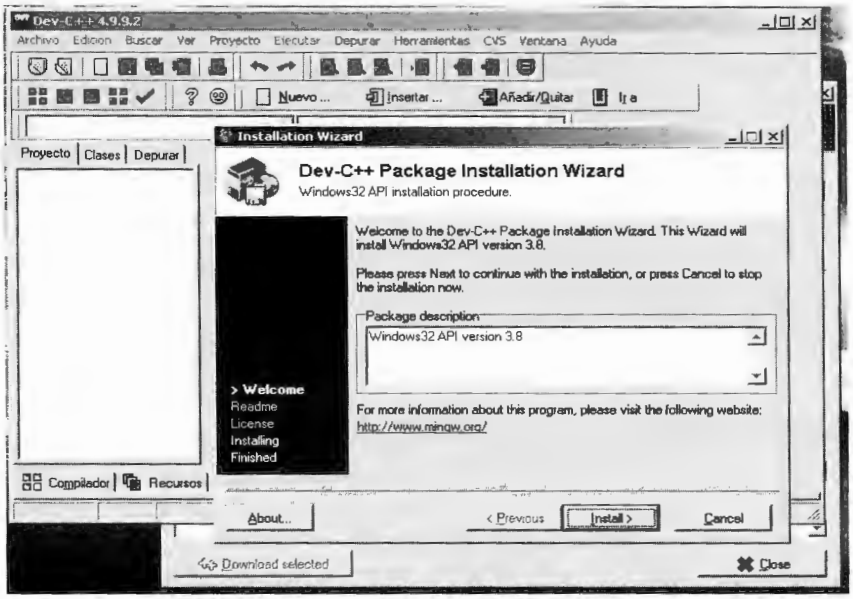


Figura 1.12.c). Se instalan los dos grupos de actualizaciones recomendados, y DevCpp queda dispuesto para su utilización.

El entorno DevCpp funciona en cualquier versión de Windows, desde Windows 95 en adelante. Las imágenes mostradas en el libro corresponden a DevCpp ejecutándose bajo Windows XP, que a su vez funcionaba en Parallels Desktop bajo Mac OS X 10.4.9. *Véase www.parallels.com.*

Figura 1.13.a)

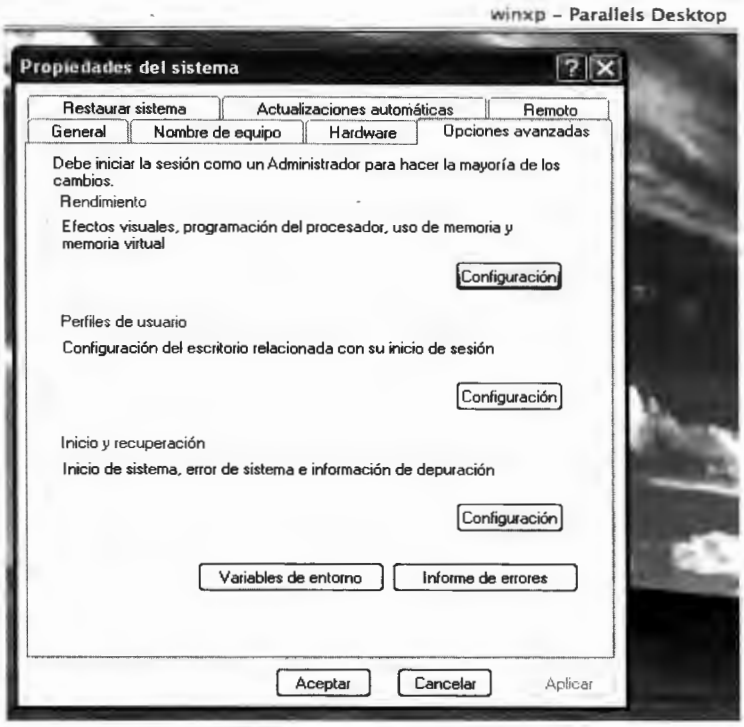


Figura 1.13.b)

Figura 1.13.c). Configuración de la variable de entorno PATH para ejecutar gcc en DevCpp.

1.6. Herramientas de compilación y edición

Los programas encargados de generar un fichero de código ejecutable partiendo de un fichero de código fuente (el compilador y el enlazador) son utilidades que, en su origen, se gobiernan a través de la línea de órdenes de una terminal. La ejecución más eficiente posible de la utilidad gcc, que como se ha visto es la encargada de producir un ejecutable, es la que se consigue a través de una terminal. Todas las instalaciones de EID o IDE mencionadas hasta el momento conllevan la instalación de gcc.

En el caso de un sistema operativo basado en UNIX, como Linux o Mac OS X, la instalación de un IDE, como KDevelop o XCode deja configurada correctamente la terminal para utilizar directamente gcc. Por tanto, se pueden utilizar las órdenes gcc (Linux) y cc (Mac OS X) para construir programas; por supuesto, el resto de las utilidades de desarrollo (make, por ejemplo) también estará disponible.

En el caso de DevCpp, es preciso configurar manualmente la variable de entorno PATH, añadiendo la ruta siguiente:

```
C:\DevCpp\bin
```

y de este modo la terminal "verá" a gcc y demás utilidades instaladas por DevCpp, haciendo posible utilizar la terminal para construir y ejecutar programas.

Evidentemente, para escribir el código se necesita un editor de textos. Estos pueden ser basados en texto (del tipo de los que se ejecutan en el interior de una terminal) o de interfaz gráfica de usuario (del tipo habitual en la actualidad, con posibilidad de emplear distintos tamaños y tipos de letra, etc.). Sea cual fuere el editor empleado, es preciso guardar el programa con formato de texto, y con la extensión .c. Esto es importante, porque el comportamiento del compilador depende de la extensión utilizada. Nuestros programas deben almacenarse siempre en archivos con la extensión .c, no con la extensión .cc ni .cpp, por ejemplo. Esto daría lugar a la compilación del programa como si estuviese escrito en el lenguaje C++, que es algo completamente distinto. Es un error frecuente aceptar la extensión correspondiente a C++ en lugar de admitir la extensión correspondiente a C. Esto da lugar a errores de difícil solución, al corresponder estos a un lenguaje como C++, que es bien distinto, insistimos, del lenguaje C. Si el lector no quiere problemas, hará bien en utilizar siempre la extensión .c para sus archivos de código fuente.

Instalación de Eclipse

El primer paso consiste en acceder a la página de descargas y hacer clic en el botón "Download Eclipse". También, se puede acceder en directo a la página

http://www.eclipse.org/downloads/

Allí es preciso hacer clic en el enlace "Eclipse IDE for C/C++ Developers", lo cual dará lugar a la descarga de un archivo. Este archivo, una vez descomprimido, produce una carpeta llamada eclipse, que contiene todo el entorno de desarrollo. La carpeta puede albergarse en cualquier lugar del disco, y conviene crear un acceso directo en el escritorio (Windows) o situar un icono de Eclipse en el Dock (Mac OS).

Eclipse está basado en Java, una potente plataforma de programación [*véase,* por ejemplo, *JAVA se6 & SWING,* del mismo autor, publicado por Pearson Educación en la colección

"Manual de Aprendizaje" (octubre 2007)]. Si se utiliza la plataforma Windows o Linux, es preciso acudir a *http://java.sun.com* e instalar Java 2 SE siguiendo las instrucciones que allí se muestran. Si se utiliza Mac OS X, Java se habrá instalado automáticamente junto con el sistema operativo).

Este instalador no incluye un compilador de C; únicamente contiene un entorno de desarrollo. Las plataformas basadas en UNIX ofrecen automáticamente un sistema de desarrollo en C, que normalmente es una variante de GCC (*véase* la Sección 1.5). Se puede emplear como compilador el ofrecido por MinGW, que se puede descargar en la página siguiente:

http://sourceforge.net/project/showfiles.php?group_id=2435

Es conveniente descargar la versión marcada como "current" para el API de Windows. Al hacer clic en este enlace ("current") aparecerá una segunda página, desde la cual se efectúa la descarga en sí. El paquete se encuentra en el enlace "MinGW". Es posible que el navegador impida inicialmente la descarga del instalador; de ser así, será preciso indicar que realmente se desea descargar el compilador. Una vez descargado, será preciso descomprimirlo (el archivo tiene el formato .tar.gz) antes de comenzar a instalarlo. Una posibilidad es emplear el descompresor gratuito Stuffit Expander, que se puede obtener en la dirección *http://www.stuffit.com.* Por supuesto, se puede emplear cualquier otro programa capaz de descomprimir archivos de los formatos .tar y .gz.

Lo que se instala es un programa que permite seleccionar las partes de MinGW que se quieren instalar realmente. Se puede admitir la configuración por omisión, aunque los más osados pueden descargar también el compilador de Objective-C.

Se recomienda instalar primero el entorno MinGw y después el entorno Eclipse.

1.6.1. Editores para su uso en la terminal

Las herramientas habituales de los entornos Unix (vi, vim, emacs, pico/nano) son perfectamente adecuadas para construir programas. Se han mencionado en primer lugar las más potentes, aunque la última (pico) es sin duda la de uso más sencillo. También existen versiones de estas herramientas para MS-DOS, aunque resulta preferible emplear alguna herramienta basada en una IGU.

Por su parte, los editores de MS-DOS (por ejemplo, edit) también son viables para construir programas, aunque reconocidamente no son los más potentes.

1.6.2. Editores con interfaz gráfica de usuario

Estos editores de texto se caracterizan por facilitar mucho la tarea del programador, ofreciendo una interfaz gráfica de usuario de agradable aspecto. Sus capacidades son, a nuestros efectos, análogas o superiores a las de los editores basados en una terminal, por ser más intuitivos.

1.6.2.1. Linux

Posiblemente kate (KDE) y gedit (GNOME) sean los más usados, y resultan suficientes para construir programas, aunque sin llegar a la perfección de los editores de textos que ofrece un IDE.

1.6.2.2. Mac OS X

Resulta muy agradable el editor TextWrangler (versión reducida de BBEdit), que es gratuito y puede obtenerse en la dirección *http://www.barebones.com.*

1.6.2.3. Windows

Recomendamos el programa TextPad (*http://www.textpad.com*), un programa shareware muy probado. También resulta agradable Crimson Editor, que se puede descargar en la dirección *http://www.crimsoneditor.com.*

Véase además la página

http://www.thefreecountry.com/programming/editors.shtml

que muestra una gran número de editores para distintas plataformas. Es conveniente estudiar las posibilidades que ofrecen los editores, puesto que su conocimiento puede dar lugar a notables ahorros de tiempo y esfuerzo. En particular, se recomienda encarecidamente estudiar el uso de Expresiones Regulares, presentes en todos los buenos editores, y cuyo uso correcto es la marca del buen programador y usuario de sistemas informáticos. Igualmente, el uso de macros puede simplificar ciertas tareas repetitivas.

1.7. El programa HolaMundo. Escritura, compilación y ejecución

Una vez instalado cualquier sistema de desarrollo, y con objeto de probarlo, debe escribirse un primer programa que después se compilará y ejecutará. Este programa se denomina "Hola, Mundo" y posiblemente apareciera por primera vez en la obra "The C Programming Language", escrita por Brian Kernighan y Dennis Ritchie (*véase http://cm.bell-labs.com/cm/cs/cbook/*). El programa es el mostrado como primer ejemplo de código:

HolaMundo.c

```
#include <stdio.h>

int main(int argc, char * argv[])
{
    printf("%s", "Hola, Mundo!\n");
    printf("%s", "Hoy es Jueves\n");
    return 0;
}
```

Las figuras siguientes muestran la forma de crear un proyecto, escribir el código fuente, compilarlo y ejecutarlo. La tarea es bastante sencilla, aunque hay que prestar atención a cosas como la dirección de las barras ("\" no es lo mismo que "/") y también hay que incluir tres veces el signo de punto y coma. Si se reproduce fielmente el programa que se muestra, funcionará a la primera.

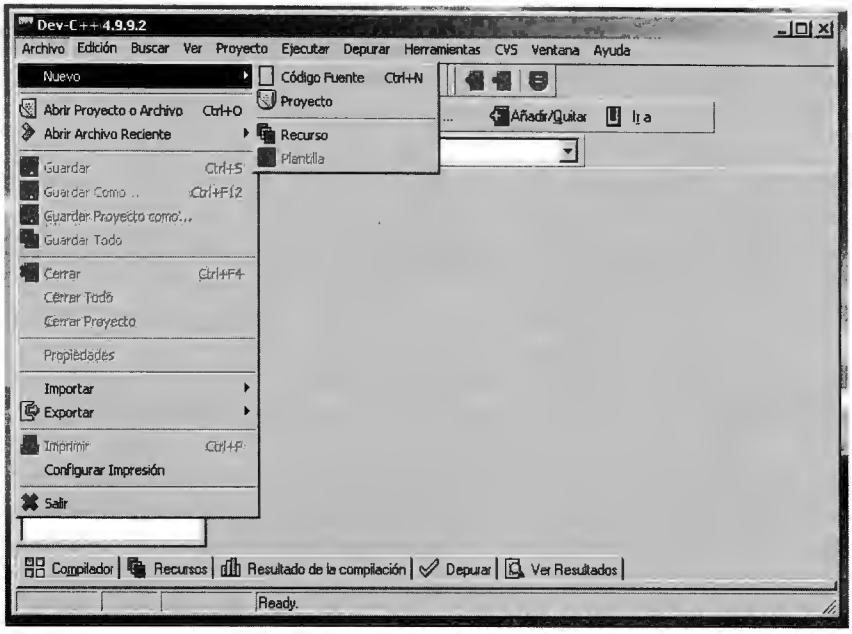


Figura 1.14.a)

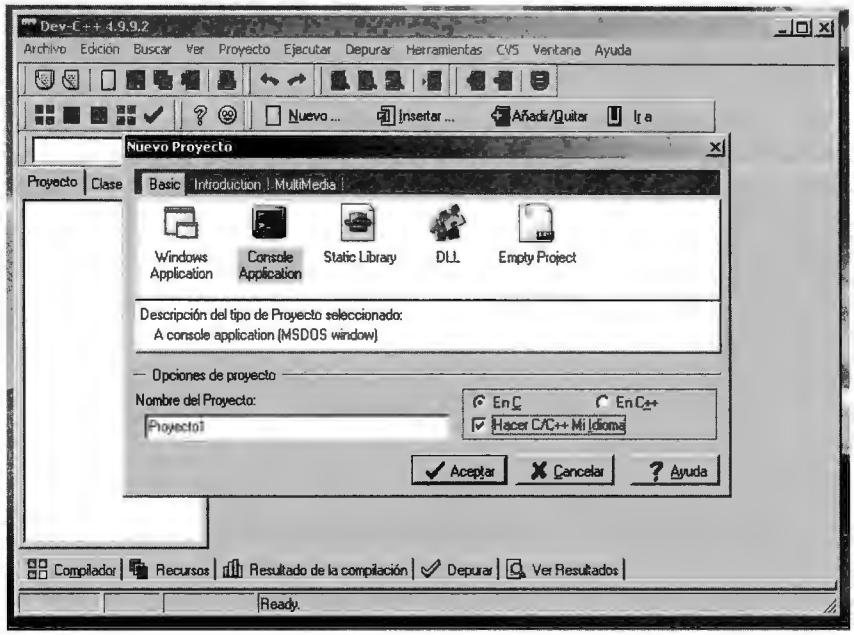


Figura 1.14.b)

Figura 1.14.c). Creación de un proyecto desde DevCpp.

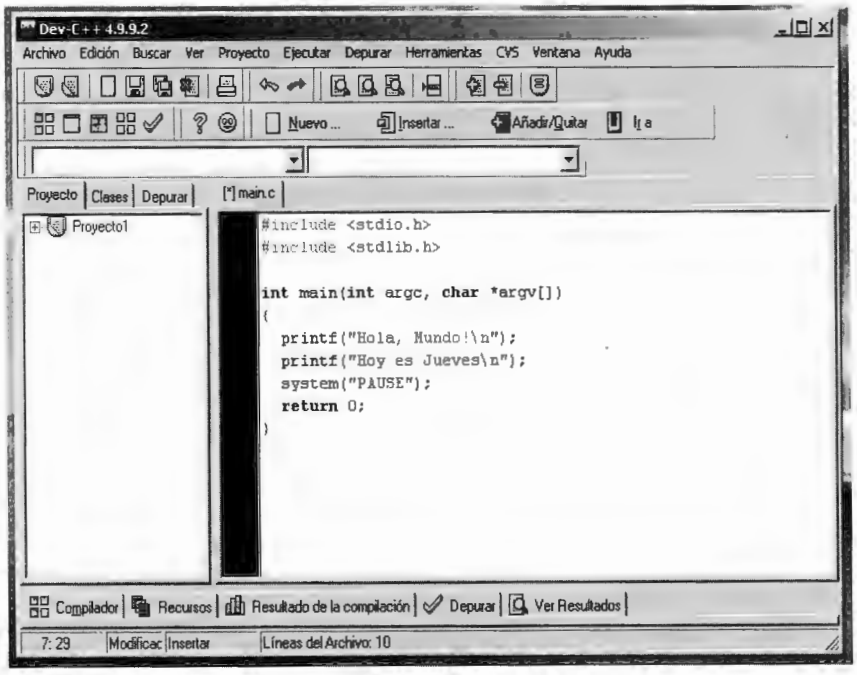


Figura 1.15. Creación del código fuente desde DevCpp.

Figura 1.16. Ejecución del código desde DevCpp.

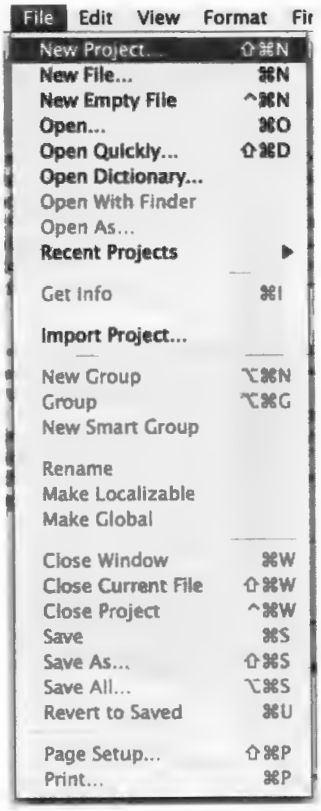


Figura 1.17.a)

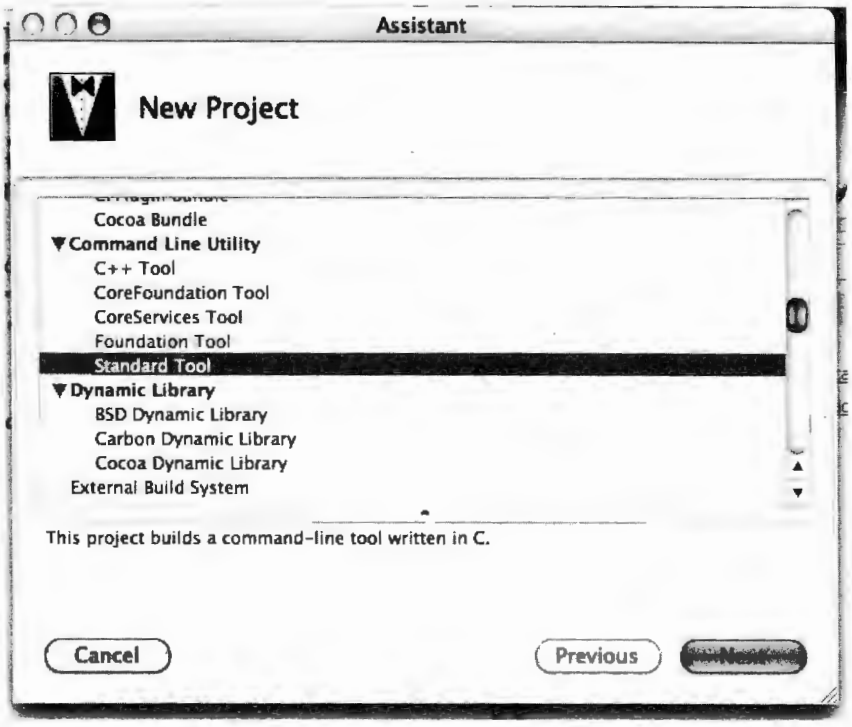


Figura 1.17.b)

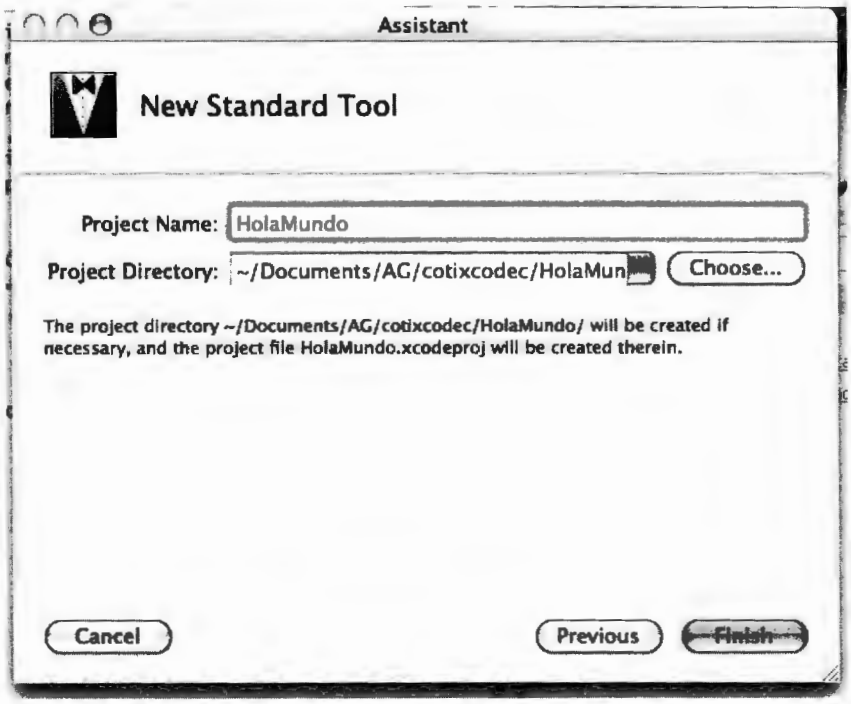


Figura 1.17.c). Creación de un proyecto desde XCode.

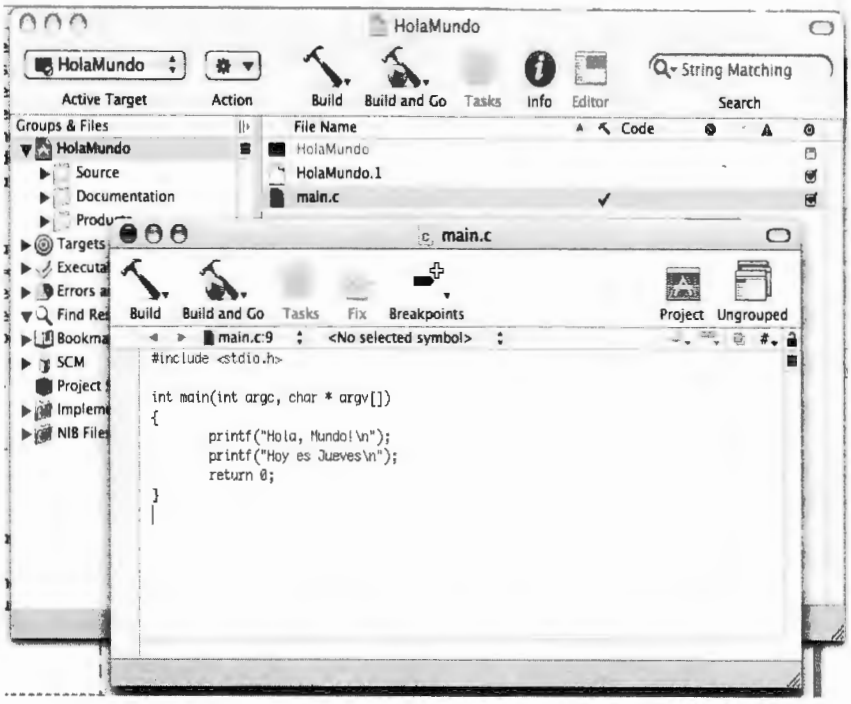


Figura 1.18. Creación del código fuente desde XCode.

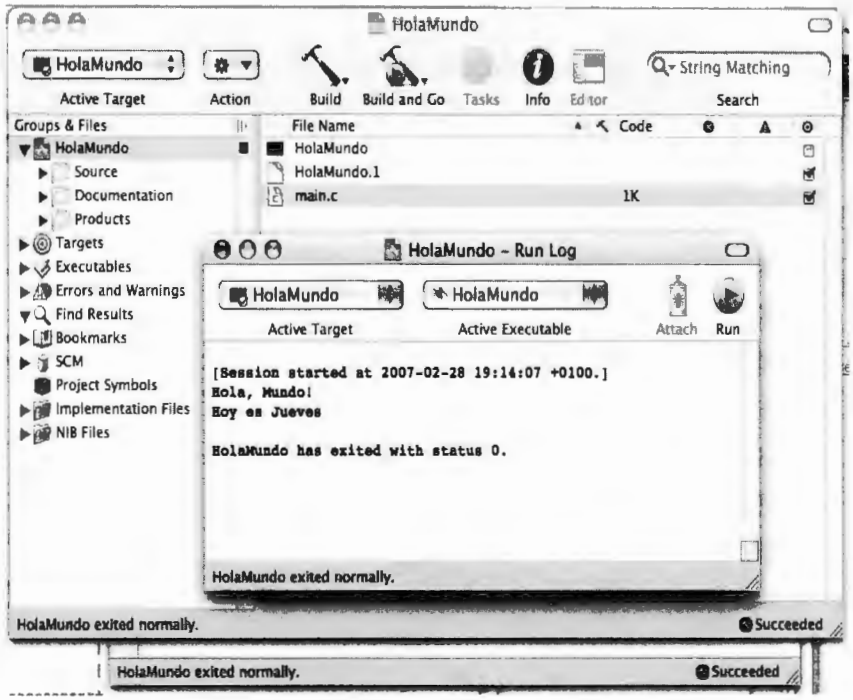


Figura 1.19. Ejecución del código desde XCode.