

Contenido

Prefacio	VII
-----------------------	-----

CAPÍTULO 1

Instalando un entorno de desarrollo para el libro	1
----------------------------------------------------------------	---

1.1. ¿Qué es un entorno de desarrollo?.....	1
---------------------------------------------	---

1.1.1. ¿Qué es un entorno de desarrollo?.....	1
-----------------------------------------------	---

1.1.2. ¿Qué nos proporciona un entorno de desarrollo?	1
-------------------------------------------------------------	---

1.1.3. Compiladores de C/C++	2
------------------------------------	---

1.1.4. Herramientas de virtualización	3
---------------------------------------------	---

1.1.4.1. Configuración del networking entre sistemas virtuales y físicos	3
--------------------------------------------------------------------------------	---

1.1.5. Escoger herramientas para cada lector.....	4
---------------------------------------------------	---

1.1.6. La herramienta de Visual Studio Code	5
---------------------------------------------------	---

1.2. Instalación del entorno de desarrollo aconsejado	6
-------------------------------------------------------------	---

1.2.1. Orden de instalación de los elementos de entorno de desarrollo	7
-----------------------------------------------------------------------	---

1.2.2. Paso 1: descargar los ejemplos de código fuente del libro .	7
--------------------------------------------------------------------	---

1.2.3. Paso 2: instalar el compilador MinGW	8
---------------------------------------------------	---

1.2.3.1. Comprobar que MinGW está bien instalado.....	10
-------------------------------------------------------	----

1.2.4. Paso 3: instalar Visual Studio Code (VSCode).....	10
----------------------------------------------------------	----

1.2.4.1. Acoplado los <i>plugins</i> a VSCode	12
-----------------------------------------------------	----

1.2.5. Comprobando el funcionamiento de VSCode	13
------------------------------------------------------	----

1.2.6. Paso 4: instalar VirtualBox..	14
--------------------------------------	----

1.2.7. Paso 5: obtener una versión de Ubuntu Linux.....	15
---------------------------------------------------------	----

1.2.7.1. Comprobar que funciona la máquina virtual Linux.....	16
---------------------------------------------------------------	----

1.2.8. Paso 6: instalar Eclipse.....	17
--------------------------------------	----

1.2.9. Paso 7: instalando Visual Studio 2022 Community Edition	20
----------------------------------------------------------------------	----

1.3. Conclusión del capítulo 1	21
--------------------------------------	----

CAPÍTULO 2

Nuestro primer programa en lenguaje C	23
----------------------------------------------------	----

2.1. HolaMundo: nuestro primer programa en C	23
----------------------------------------------------	----

2.1.1. Las librerías de include.....	23
--------------------------------------	----

2.1.2. Sumario de la función main	24
-----------------------------------------	----

2.1.3. La función printf	25
--------------------------------	----

2.1.4. ¿Qué cambios tenemos en este nuevo código fuente?	27
----------------------------------------------------------------	----

2.1.5. Cómo imprimir los parámetros de línea de comando.....	30
--------------------------------------------------------------	----

2.1.5.1. ¿Qué novedades vemos?..	30
----------------------------------	----

2.1.5.2. ¿Qué es el código <i>spaghetti</i> ?	31
-----------------------------------------------------	----

2.1.6. Depurar programas en C.....	33
------------------------------------	----

2.1.7. En depuración, ¿qué significa un <i>breakpoint</i> ?	35
-------------------------------------------------------------------	----

2.1.8. ¿Cómo se pone en marcha la depuración (<i>debugging</i>)?	35
2.1.9. ¿Cómo se configura la depuración en VSCode?	35
2.1.9.1. Arrancando el modo depuración.....	36
2.1.10. La barra de herramientas de <i>debug</i>	36
2.1.11. La configuración de ventanas del modo <i>debug</i>	37
2.1.12. Cambios en el punto del <i>breakpoint</i>	39
2.1.13. Cómo enviar argumentos de línea de comando al depurar	43
2.1.14. En el próximo capítulo	46

CAPÍTULO 3

Variables alfanuméricas	47
3.1. Tipos de datos alfanuméricos	47
3.1.1. El tipo de variable <i>char</i>	47
3.1.2. Cadenas de caracteres y sus punteros.....	48
3.1.2.1. Errores típicos y catastróficos con las cadenas de texto.....	49
3.1.2.2. Cómo inicializar cadenas de texto	49
3.1.2.3. ¿Qué novedades tenemos en este programa?	51
3.1.2.4. Un programa para pasar de mayúsculas a minúsculas	53
3.1.2.5. Una página de códigos equivale a ¡una margarita!	58
3.1.2.6. Las impresoras de margarita y las páginas de códigos.....	58
3.1.2.7. Trabajar con caracteres en chino con UTF-16	59
3.1.3. Idiomas y culturas: el ejemplo chino	62

3.1.4. Representar pictogramas con caracteres occidentales.....	64
3.1.4.1. La escritura Wade-Giles	64
3.1.4.2. El sistema de escritura Pinyin	65
3.1.5. Ventajas del sistema de pictogramas	67
3.1.6. El área CJKV.....	67
3.1.6.1. La escritura en Japón	68
3.1.6.2. La escritura en Corea	69
3.1.6.3. La escritura en Vietnam	69
3.1.6.4. Las diásporas históricas desde Asia.....	70
3.1.7. La simplificación o reforma de los caracteres chinos	73
3.1.8. Y llegó la tecnología Unicode con los 32 bits	74
3.1.8.1. Diferencias prácticas con Unicode entre chino simplificado y chino tradicional	74
3.1.9. Culturas y programación informática	79
3.1.10. ¿En qué año vives?	79
3.1.11. La función <i>setlocale</i>	81
3.1.11.1. Conversión de UTF-16 a UTF-8	84
3.1.11.2. Llegaron las cadenas de caracteres Unicode	85
3.1.11.3. Recapitulando sobre los punteros a caracteres	87
3.1.11.4. <i>Stack</i> y <i>heap</i> , ¿dónde guardamos nuestras variables?	87
3.1.11.5. Pilas contra montones	88
3.1.12. ¿Veis cómo los montones son mejores?	89
3.1.12.1. Cómo declarar, inicializar y asignar cadenas de caracteres dinámicas.....	90

3.1.12.2. Dos formas de recorrer una cadena de texto 92
 3.1.12.3. Función de búsqueda de texto mediante puntero-cursor 96
 3.1.13. *Stack* y *heap*: funciones y procesadores..... 96

CAPÍTULO 4

Tipos de datos numéricos. Coma flotante y binarios..... 101
 4.1. Los tipos básicos numéricos 101
 4.1.1. Enteros, punteros a enteros y copias de enteros.....102
 4.1.1.1. Límites de cálculo de los números enteros de diferentes formatos 102
 4.1.1.2. El tipo numérico entero sin signo 104
 4.1.1.3. Anécdota de un regulador de velocidad. ¿Con signo o sin signo? 106
 4.1.2. Modificadores.....108
 4.1.2.1. El modificador *unsigned* .108
 4.1.2.2. El modificador *const* 109
 4.1.2.3. El modificador *constexpr* 109
 4.1.2.4. El modificador *static* 109
 4.1.3. El tipo *float* y los números decimales 109
 4.1.3.1. Coma fija y coma flotante 110
 4.1.3.2. Estructura de un coma flotante en memoria 110
 4.1.4. El tipo *long double* o doble precisión 112
 4.1.5. El tipo *long double* o cuádruple precisión 113

4.1.6. Anécdota histórica: fallo en el cálculo de coma flotante del Pentium de Intel 113
 4.1.7. Tipos numéricos máscara binaria: *BYTE*, *WORD* y *DWORD*.... 115
 4.1.7.1. El sistema binario de numeración 116
 4.1.7.2. Conversiones a hexadecimal..... 118
 4.1.7.3. Álgebra de Bool para números binarios..... 119
 4.1.8. La operación *NOT* 120
 4.1.9. La operación *OR* 120
 4.1.10. La operación *NOR*..... 121
 4.1.11. La operación *AND*..... 122
 4.1.12. La operación *NAND* 122
 4.1.13. La operación *XOR* 123
 4.1.14. Aplicación de puertas *XOR* y *AND* para un semisumador de 1 bit 124
 4.1.14.1. La operación de desplazamiento *SHL* (multiplicación por 2) 125
 4.1.14.2. La operación de desplazamiento *SHR* (división por 2) 126
 4.1.15. Transformación de puertas *OR* en puertas *AND* y viceversa 126
 4.1.15.1. Puerta *NOR* construida con transistores 127
 4.1.15.2. Puerta *AND* construida con transistores 128
 4.1.16. La tecnología *TTL* y cómo esto nos influyó desde los años 60..... 128
 4.1.17. Orden *Endian* de los bytes 132
 4.1.17.1. *Endianidad* en el procesador 133

4.1.18. Cómo dividir un WORD en byte alto y byte bajo	134
4.1.19. La resta, el signo y el complemento a 1	137
4.1.19.1. Investigando cómo se hace la resta	138
4.1.20. El complemento a 1	140
4.1.21. Operadores de máscara de bits en C	141
4.1.22. Función en C para hallar el complemento a 1	142
4.1.22.1. El programa de resta binaria con máscaras binarias en lenguaje C	145
4.1.23. La directiva asm y las máscaras binarias	149
4.1.23.1. La arquitectura escalada de Intel y su influencia en los sistemas operativos	153
4.1.23.2. Memoria y multitarea mejoradas	154

CAPÍTULO 5

Structs, unions y typedefs	157
5.1. El tipo de dato struct y su complementario union	157
5.1.1. Ejemplo básico de un struct	157
5.1.2. Ejemplo de creación de un nuevo tipo de dato con typedef struct	158
5.1.3. Ejemplo de ordenación de una matriz de clientes	162
5.1.3.1. Ahora, ¿qué vamos a hacer con esta matriz de 10 clientes?....	162
5.1.3.2. ¿Cómo queda la función de ordenación?.....	164

5.1.3.3. Función de liberación de cadenas de texto	165
5.1.3.4. Función de impresión de listado	167
5.1.3.5. El cuerpo de programa principal	167
5.1.4. Ejemplo de struct y unión: un emulador de 2 procesadores a la vez	169
5.1.4.1. ¡Que vuelven los indios! .	174
5.1.4.2. La instrucción MOV en ensamblador	175
5.1.4.3. Codificando MOV en lenguaje máquina	176
5.1.4.4. Simulando un MOV paso a paso	177
5.1.4.5. ¿Qué hemos conseguido?.....	179
5.1.4.6. El programa de prueba de structs y unions	180
5.1.4.7. La tentación vive abajo ..	182
5.1.4.8. Importancia de unions y emuladores en sistemas operativos	183
5.1.4.9. Del sistema operativo a la virtualización y la nube (<i>Cloud Computing</i>).....	184
5.1.5. El struct como base para crear las clases en C++	185
5.1.5.1. La necesidad era manejar ordenadamente la creciente complejidad	185

CAPÍTULO 6

Creando funciones: ámbitos, parámetros y retorno	187
6.1. ¿Para qué sirven las funciones?	187

6.1.1. Librerías de funciones	187	6.1.11. Pasando parámetros por referencia	223
6.1.2. ¿Qué es un prototipo de función?	188	6.1.12. La función ObtenerTextoldiomaExt arreglada	225
6.1.2.1. El orden de las funciones sí que altera el producto	189	6.1.13. ¿Cómo quedaría ahora el programa principal?	226
6.1.2.2. Declaraciones de prototipo	191		
6.1.3. Creando una librería externa	192		
6.1.4. ¿Cómo fusionar todos estos archivos en un proyecto?	193		
6.1.4.1. Solucionando errores de gestión de proyecto.....	197		
6.1.4.2. Cómo pasar parámetros por valor a las funciones.....	200		
6.1.5. Dos enfoques para el valor de retorno de una función	205		
6.1.5.1. Bajando hasta los infiernos de la excepción	213		
6.1.5.2. ¿Cómo llamar al desensamblador de Visual Studio Code?	215		
6.1.6. Conclusión sobre las funciones que retornan punteros nulos	218		
6.1.7. Protegiendo las funciones que reciben punteros	220		
6.1.8. Aclaración sobre los valores de retorno válidos	221		
6.1.9. Prácticas poco ortodoxas para salvar un puntero loco	221		
6.1.9.1. La función definida como static	222		
6.1.9.2. Cómo asustar al compilador para que no libere una variable local	222		
6.1.10. Arreglando el diseño de la función ObtenerTextoldioma	223		

CAPÍTULO 7

Librerías de funciones clásicas de entrada/salida (I/O).....

7.1. Tres grupos de funciones de entrada/salida	229
7.1.1. ¿Por qué estos tipos de funciones de entrada/salida?	229
7.1.2. ¿Dónde se encuentran las librerías de estas funciones?.....	231
7.1.3. Un ejemplo de entrada/salida para pantalla y teclado.....	231
7.1.4. Función printf (#include <stdio.h>).....	232
7.1.5. La función scanf	233
7.1.5.1. Controlar la longitud de una cadena de texto	235
7.1.6. Funciones de entrada de caracteres individuales	236
7.1.7. Entrada de caracteres dobles	237
7.1.7.1. Captura de las teclas de función	237
7.1.8. La función getch().....	239
7.1.9. La función getche().....	240
7.1.10. La función putchar()	241
7.1.11. El teclado	241
7.1.11.1. El <i>buffer</i> (o memoria) de teclado	241
7.1.11.2. Características del <i>buffer</i> de teclado	242

7.1.11.3. El error de <i>buffer</i> <i>overflow</i>	243
7.1.12. Las lecturas de teclado a muy bajo nivel desaparecen	244

CAPÍTULO 8

Blockchain como librería de funciones externas

245

8.1. Un ejemplo de librería de
funciones Blockchain..... 245

8.1.1. Introducción

8.1.2. ¿Vamos a vender
criptomonedas?

8.1.3. ¿Qué es Blockchain?

8.2. Características de
Blockchain

8.2.1. Características de las
Blockchain a nivel político.....

8.2.2. El dilema de las
modificaciones y cómo se
soluciona

8.3. El universo Blockchain

8.3.1. Activos digitales,
criptoactivos, tokens fungibles
y NFT

8.4. Uso de los criptoactivos a nivel
financiero y de negocio.....

8.4.1. ¿Por qué son interesantes a
nivel financiero/contable?

8.4.2. A nivel de contabilización y
control de activos.....

8.4.3. Como moneda de cambio
entre empresas de grupos y de
terceros.....

8.4.3.1. Como incremento de solidez
financiera de filiales.....

8.4.3.2. Ejemplo de desacople
financiero mediante activos
digitales

8.5. De vuelta al mundo
tecnológico.....

8.5.1. La encriptación como
requisito de ciberseguridad

8.6. *Hashes* y la librería externa de
encriptación SSL.....

8.6.1. La librería OpenSSL como
librería externa.....

8.6.2. ¿Qué es un *hash*? ¿Qué
es un bloque?

8.6.3. Similitudes y relaciones entre
hashes y encriptaciones

8.6.4. Comprobación de propiedad:
demostrar ser el *owner*

8.6.5. *Hashes* truculentos o *Hecha
la ley, hecha la trampa*

8.6.6. ¿Qué es un bloque?

8.6.7. Concatenación o
encabalgamiento de *hashes*

8.6.8. Blockchain es relativamente
caro

8.6.9. Tipos de cadenas Blockchain
y costes por transacción

8.6.10. Sobre el volumen de
transacciones

8.6.11. La figura del minador de
criptoactivos.....

8.6.12. Minadores y Fintechs
diversas.....

8.6.13. Empresas más tranquilas
basadas en Blockchain

8.6.14. Tamaños de redes Blockchain
y recursos de implementación

8.6.14.1. Recursos necesarios en
empresas Fintech globales.....

255

258

258

258

258

259

260

261

261

262

263

264

264

265

266

266

267

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

268

8.6.14.2. Recursos necesarios en energéticas y telecos269

8.6.14.3. Recursos necesarios en empresas logísticas grandes269

8.6.14.4. Oficinas de notarías y servicios legales269

8.7. Los objetivos del capítulo269

8.8. Ejercicio único de este capítulo: la blockchain270

8.8.1. Entendiendo las listas dinámicas270

8.8.2. De elementos a bloques de Blockchain276

8.8.3. Funcionamiento del programa de ejemplo de Blockchain278

8.8.3.1. Niveles de funciones:.....279

8.8.3.2. Reparto de las funciones por niveles280

8.8.4. Repaso de las opciones de programa281

8.8.4.1. Dar de alta nuevo bloque281

8.8.4.2. Generar N bloques aleatorios.....282

8.8.4.3. Contar bloques283

8.8.4.4. Falsificar bloque N284

8.8.4.5. Listado de todos los bloques286

8.8.4.6. Verificar blockchain.....287

8.8.4.7. Reconstruir todo el blockchain recalculando bloques falsos290

8.8.4.8. Imprimir el *hash* de un bloque292

8.8.4.9. Imprimir solo los bloques falsos293

8.8.4.10. Salir (y, sobre todo, liberar bloques con *free*)295

8.8.4.11. Indiana Jones en busca de la memoria perdida 295

8.8.4.12. La función *LiberarBlockchain*..... 297

8.8.5. Conclusión..... 299

CAPÍTULO 9

Librerías de funciones de entrada/salida para consola de Windows 301

9.1. Aplicación de ejemplo para pantallas de texto avanzadas..... 301

9.2. Funciones de entrada/salida propias de la Consola de Windows 302

9.2.1. MS-DOS ya no existe: es una emulación de consola 303

9.2.2. ¿Cuándo programar con *printf* y cuándo usar consola de Windows?..... 305

9.2.3. Construyendo aplicaciones de Consola de Windows 306

9.2.4. La tabla de caracteres semigráficos..... 306

9.2.5. Escribiendo una función para entrada de texto: *EntraString* 309

9.2.5.1. Sintaxis de *EntraString*... 312

9.2.6. Uso de *EntraString* en el contexto del programa 318

9.2.6.1. El resto de las funciones auxiliares, de mayor a menor 320

9.2.6.2. La función *GestionarTeclasEspeciales*..... 320

9.2.6.3. La función *GoToXY*..... 324

9.2.6.4. ¿Y cómo se obtiene el handle del *buffer* de consola? 325

9.2.6.5. La función SetFGColor para cambiar colores de texto	326
9.2.7. Funciones de transformación de códigos de página de texto	327
9.2.7.1. Código fuente de la función UnicodetoAnsi	327
9.2.7.2. Código fuente de la función AnsiToUnicode.....	329
9.2.7.3. La función PrintCPLabel..	330
9.2.8. Gestionar ventanas indoloras y pintar recuadros	332
9.2.8.1. Cómo pintar una ventana con la función PintarVentana.....	332
9.2.8.2. La función PintarVentana.....	336
9.2.8.3. Implementando ventanas indoloras (<i>painless Windows</i>)	339
9.2.9. Posibles mejoras para copiar y restaurar ventanas.....	345
9.3. Lectura de ratón y eventos especiales.....	346
9.3.1. El bucle de mensajes.....	348
9.3.2. Los manejadores de eventos	349
9.3.2.1. El manejador de teclado.	349
9.3.2.2. El manejador de ratón	349
9.3.2.3. El manejador de redimensionamiento de ventana..	351
9.3.2.4. El manejador de eventos de paso de foco.....	351
9.3.2.5. El manejador de eventos de menú	352
9.3.3. El programa principal del testeador de eventos.....	353
9.3.4. Descargar el código fuente del testeador de eventos.....	355
9.3.5. En el siguiente capítulo	355

CAPÍTULO 10

Librerías de funciones de entrada/salida para ficheros (archivos)	357
10.1. Añadiendo funciones para gestión de ficheros	357
10.2. Construyendo la parte de grabación de archivos	358
10.2.1. La persistencia	358
10.2.2. Persistencia mediante bases de datos	358
10.2.3. Persistencia mediante archivos.....	359
10.2.4. Archivos secuenciales.....	360
10.2.5. Registros y campos	360
10.2.6. Algoritmo típico para un sistema de acceso secuencial.....	361
10.2.7. ¿Se usan mucho los ficheros secuenciales hoy día?	362
10.2.7.1. Ficheros secuenciales como archivos de configuración...	362
10.2.7.2. El formato CSV: un archivo secuencial de datos compatible con Excel	363
10.2.7.3. Los archivos de Log como archivos de acceso secuencial.....	364
10.2.7.4. El formato XML como archivo secuencial.....	365
10.2.7.5. El tipo JSON como archivo secuencial.....	367
10.2.7.6. El formato DBF de dBase III, Clipper y FoxPro	368
10.2.7.7. Más y más formatos secuenciales	371
10.2.8. Los archivos de acceso aleatorio (<i>random</i>).....	371
10.2.8.1. Cálculo del posicionamiento <i>random</i>	373

11.2. Dos nuevas formas de hackear Blockchain	438
11.2.1. El <i>hacking</i> por inserción de punteros.....	438
11.2.2. El <i>hacking</i> por escritura directa o recifrado del fichero de Blockchain	438
11.2.3. Analizando el problema de la inserción de un bloque falso.....	439
11.3. Funciones de la capa de persistencia Blockchain	441
11.3.1. ExistsFile.....	442
11.3.2. CrearFicheroBlockchain ...	443
11.3.3. ContarBloquesDesdeFichero	444
11.3.4. LeerBloque	445
11.3.5. LeerCadenaBloquesEnteraDesdeFichero	447
11.3.6. EscribirBloque	452
11.3.7. EscribirCadenaBloquesEnteraHaciaFichero	454
11.4. Código fuente comentado del programa principal.....	457
11.5. Conclusión.....	464

CAPÍTULO 12

Recursividad, árboles binarios, traversas y punteros a funciones 467

12.1. Introducción a los procesos recursivos	467
12.1.1. ¿Todos los procesos se pueden hacer recursivos?.....	467
12.1.2. El cálculo de factorial como ejemplo básico	468
12.1.3. Código fuente de la factorial de coma flotante, 64 bits	471
12.1.4. Dónde se encuentra el código fuente de las factoriales.....	472

12.2. Programación recursiva de determinantes	475
12.2.1. ¿Qué es un determinante?	475
12.2.2. Organizar las tareas del determinante	478
12.2.2.1. El acceso bidimensional a los elementos de la matriz.....	480
12.2.3. El programa principal	485
12.3. Introducción a los árboles binarios	488
12.3.1. ¿Qué son los árboles binarios?	488
12.3.2. Comparación de listas con árboles	488
12.3.3. Listas doblemente enlazadas.....	490
12.3.4. El árbol binario de búsqueda (BST o Binary Search Tree)	492
12.3.5. Los árboles en Prolog.....	495
12.4. Un ejemplo de diseño de árbol en C.....	496
12.5. Insertando datos en un árbol	497
12.5.1. Preparar el árbol.....	497
12.5.2. Insertamos una clave y un valor en un árbol vacío.....	498
12.5.3. Insertamos clave y valor en un árbol con datos preexistentes	498
12.5.3.1. Creamos la función de inserción de tipo recursivo.....	499
12.5.4. Experimentando con la secuencia de inserción	505
12.6. Introducción a las rotaciones	507
12.6.1. Paso 1	507
12.6.2. Paso 2	508
12.6.2.1. Transición del paso 2 al paso 3	508

12.6.3. Paso 3.....	509
12.7. Insertando todos los elementos de prueba.....	509
12.8. Buscando elementos dentro del árbol.....	511
12.8.1. Código fuente de BuscarNodo.....	512
12.8.2. Comprobación del número de comparaciones.....	516
12.9. Traversa In Order y punteros a funciones.....	516
12.9.1. Los métodos de traversas (traversal).....	516
12.9.2. La traversa in order.....	517
12.9.3. Punteros a funciones.....	518
12.9.3.1. Cómo declarar un prototipo de puntero a función.....	519
12.9.3.2. Asignar una función genérica a este puntero.....	522
12.9.4. Función para contar elementos basada en traversa.....	524
12.10. Función para liberar nodos y el árbol en sí.....	525
12.11. La traversa post order para eliminar el árbol.....	527
12.11.1. Código fuente de la traversa post order.....	527
12.11.2. La función LiberarNodo	528
12.12. Código fuente final.....	529
12.13. Conclusión.....	529