ANTONIO GARRIDO CARRILLO

METODOLOGÍA DE LA PROGRAMACIÓN

de bits a objetos

GRANADA
2016

© Antonio Garrido Carrillo
© Universidad de Granada
Metodología de la Programación: de bits a objetos
ISBN: 978-84-338-5868-9.
Depósito legal: GR.160-2016
Edita: Editorial Universidad de Granada.
Campus Universitario de Cartuja. Granada.
Imprime: Gráficas La Madraza. Albolote. Granada.

Printed in Spain *Impreso en España*.

Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra sólo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley.

Índice general

1 Representación en binario 1

1.1 Introducción 1

1.2 Bases binaria, octal y hexadecimal 1

1.2.1 De base 10 a base b 2
1.2.2 Cambio agrupando dígitos 4
1.2.3 Números con parte fraccionaria 5

1.3 Enteros y reales en el ordenador 6

1.3.1 Representación de enteros 6
1.3.2 Representación de reales 8

1.4 Arquitectura del ordenador 10

1.4.1 Datos en memoria 11

1.5 Caracteres 12

1.5.1 Código ASCII: Extensiones 13
1.5.2 Unicode 14

1.6 Problemas 16

2 Tipos básicos: Bits 17

2.1 Introducción 17

2.1.1 Tipos y lenguaje de alto nivel 17

2.2 Tipos aritméticos 18

2.2.1 Tipo booleano 18
2.2.2 Tipos carácter 19

2.2.3 Tipos enteros . . . 19
2.2.4 Tipos en coma flotante . . . 20
2.2.5 Sinónimos con typedef . . . 21
2.3 Características de los tipos 22
2.3.1 Tamaños. El operador *sizeof* . . . 23
2.3.2 Límites numéricos . . . 23
2.4 Tipo enumerado 24
2.4.1 Representación del tipo enumerado . . . 25
2.5 Conversiones 26
2.5.1 Conversiones implícitas . . . 27
2.5.2 Conversiones explícitas . . . 29
2.6 Tipos de datos y bits 31
2.6.1 Operadores a nivel de bit . . . 32
2.6.2 Campos de bits . . . 35
2.6.3 Uniones . . . 36
2.7 Problemas 39

3 Vectores y matrices . . . 41
3.1 Introducción 41
3.2 Vectores 41
3.2.1 Declaración . . . 42
3.2.2 Vector de constantes . . . 44
3.2.3 Límites de un vector . . . 45
3.3 Paso de vectores a funciones 46
3.3.1 Vectores de constantes . . . 48
3.3.2 Tamaño delimitado por *centinela* . . . 52
3.4 Ejemplos: Búsqueda y ordenación 52
3.4.1 Algoritmos de búsqueda . . . 52
3.4.2 Algoritmos de ordenación . . . 54
3.5 Cadenas de caracteres 56
3.5.1 Paso de cadenas a funciones . . . 57
3.5.2 E/S de cadenas . . . 58
3.5.3 Cadenas de caracteres y tipo *string* . . . 60
3.6 Matrices 61
3.6.1 Declaración . . . 62
3.6.2 Almacenamiento y límites de matrices . . . 62
3.6.3 Paso de matrices a funciones . . . 63
3.7 Arrays multidimensionales 70
3.7.1 Paso de arrays a funciones . . . 71
3.8 Problemas 72

4 Estructuras y vectores-C 75

4.1 Introducción 75

4.2 Definición de estructuras 76

4.2.1 Declaración de variables 76

4.2.2 Definición en C y typedef 77

4.2.3 El operador punto 78

4.3 Mezclando estructuras y vectores-C 78

4.3.1 Asignación y copia de estructuras con vectores 81

4.3.2 Estructuras como miembro 84

4.4 Problemas 88

5 Punteros 89

5.1 Introducción 89

5.2 El tipo de dato *Puntero a T* 89

5.2.1 Operaciones básicas 91

5.2.2 Puntero a *void* 93

5.3 Estructuras 94

5.3.1 Puntero a un campo de la estructura 95

5.3.2 Punteros miembro a la propia estructura 96

5.3.3 Declaración adelantada 97

5.4 Funciones 98

5.4.1 Devolución de punteros a objetos locales 99

5.5 Vectores 100

5.5.1 Aritmética de punteros 101

5.6 Punteros y const 103

5.6.1 Punteros, funciones y const 105

5.6.2 Punteros, vectores y const 105

5.6.3 typedef y const 106

5.7 Cadenas de caracteres 107

5.8 Problemas 108

6 Memoria dinámica 113

6.1 Introducción 113

6.2 Estructura de la memoria 113

6.3 Reservar y liberar memoria dinámica 115

6.3.1 Reserva y liberación de espacio para un objeto 115

6.3.2 Reserva y liberación de vectores 119

6.3.3 Gestión automática de memoria 122

6.4 Ejemplos de estructuras de datos simples 122
6.4.1 Celdas enlazadas . . . 122
6.4.2 Vectores y Matrices . . . 126
6.4.3 Cursores . . . 129
6.5 Problemas 131

7 Funciones . . . 135
7.1 Introducción 135
7.2 La función *main* 135
7.2.1 Otras formas de finalización de *main* . . . 137
7.2.2 Parámetros de main y tipo *string* . . . 138
7.3 La Pila 139
7.3.1 Estructura de la memoria . . . 139
7.3.2 Variables locales a un bloque . . . 141
7.3.3 Resolviendo llamada y retorno . . . 142
7.4 Paso parámetros/devolución de resultados 143
7.4.1 Paso/devolución en C . . . 144
7.4.2 Paso/devolución en C++. Tipo referencia . . . 147
7.4.3 Parámetros y *const* . . . 153
7.4.4 Ejemplo de paso de parámetros . . . 155
7.5 Nuevas características 156
7.5.1 Funciones *inline* . . . 156
7.5.2 Sobrecarga de funciones . . . 157
7.5.3 Parámetros con valor por defecto . . . 159
7.5.4 Variables locales *static* . . . 159
7.6 Punteros a función 160
7.6.1 Nombrando el tipo puntero a función . . . 164
7.7 Problemas 165

8 Errores . . . 169
8.1 Introducción 169
8.1.1 Errores de compilación y ejecución . . . 170
8.1.2 Errores de origen endógeno y exógeno . . . 171
8.2 Localizar y depurar errores 174
8.2.1 Compilación vs ejecución . . . 174
8.2.2 Errores lógicos . . . 176
8.2.3 Invariantes . . . 177
8.2.4 Localización de errores . . . 180
8.3 Mecanismos de gestión de errores 184
8.3.1 ¿Qué errores gestionar? . . . 185
8.3.2 Sistemas clásicos de gestión . . . 188
8.3.3 Excepciones . . . 192

8.3.4 Excepciones estándar . . . 205
8.4 Problemas 208

9 Clases: conceptos básicos . . . 209
9.1 Introducción 209
9.1.1 Abstracción . . . 209
9.1.2 Usando abstracciones . . . 210
9.2 Diseño basado en la abstracción 212
9.2.1 El problema: sumar matrices . . . 212
9.2.2 Una solución directa sin módulos . . . 213
9.2.3 Abstracción funcional . . . 214
9.2.4 Abstracción de datos . . . 217
9.3 Diseño e implementación de clases simples 221
9.3.1 Ocultamiento de información . . . 221
9.3.2 Definición de una clase . . . 224
9.3.3 Separación en archivos . . . 235
9.4 Encapsulamiento: la clase *Matriz* 238
9.4.1 Representación en base a un vector de tamaño fijo . . . 238
9.4.2 Implementación con la *STL* . . . 246
9.5 Problemas 247

10 Clases: diseño . . . 249
10.1 Introducción 249
10.2 Diseñar la parte interna 249
10.2.1 Función de abstracción . . . 250
10.2.2 Invariante de la representación . . . 250
10.2.3 Métodos privados . . . 251
10.3 Diseñar la interfaz 252
10.3.1 Selección de las operaciones . . . 252
10.3.2 Niveles de encapsulación . . . 253
10.3.3 La interfaz implementa la abstracción . . . 257
10.3.4 Atributos públicos . . . 258
10.3.5 Orden de las secciones de una clase . . . 259
10.4 Atributos y métodos de clase 259
10.4.1 Miembros static constantes . . . 262
10.4.2 Atributos/métodos de clase vs variables/funciones globales . . . 263
10.5 Clases como atributos 265
10.5.1 Clases y métodos amigos . . . 266
10.5.2 Construir los atributos . . . 266
10.6 Diseño de nuevos tipos 269
10.6.1 Ejemplos de clases . . . 269
10.6.2 Estructuras y clases . . . 273

10.7 Problemas **275**

11 Clases: avanzado 277

11.1 Introducción **277**
11.1.1 Comportamiento como tipos predefinidos 277

11.2 Funciones básicas de una clase **279**
11.2.1 Construcción y destrucción 281
11.2.2 Copia y asignación 284
11.2.3 Funciones miembro generadas por defecto 291

11.3 Construcción, destrucción y asignación en detalle **294**
11.3.1 Llamadas a constructores y destructores 294
11.3.2 Aprovechando constructor y destructor en la asignación 302
11.3.3 Ejemplos de llamadas generadas 304
11.3.4 Eficiencia de las operaciones 309

11.4 Definiciones dentro de la clase **313**
11.4.1 Clases dentro de clases 314

11.5 Excepciones **316**
11.5.1 Clases y rebobinado de la pila 316
11.5.2 Excepciones vs devolución de errores 317
11.5.3 Código seguro ante excepciones 318

11.6 Problemas **326**

12 Sobrecarga de operadores 329

12.1 Introducción **329**
12.1.1 Sobrecarga de funciones/operadores 330

12.2 Mecanismo de sobrecarga de operadores **330**
12.2.1 Sobrecarga como función externa 330
12.2.2 Sobrecarga como función miembro 331
12.2.3 Operadores como funciones miembro o externas 332

12.3 Sobrecargando operadores **333**
12.3.1 Operador de indexación 334
12.3.2 Operadores de asignación 337
12.3.3 Operadores relacionales 339
12.3.4 Operadores de incremento y decremento 340
12.3.5 Operadores de E/S 342
12.3.6 Operador de *llamada a función* 345
12.3.7 Ejemplo: la clase *Complejo* 349

12.4 Problemas **354**

13 E/S. Acceso aleatorio 355

13.1 Introducción **355**

13.2 Flujos asociados a ficheros 356
13.2.1 Flujos y buffers . . . 357
13.2.2 Ficheros de lectura y escritura . . . 358
13.2.3 Apertura y cierre de archivos . . . 358
13.3 Acceso secuencial y aleatorio 361
13.3.1 Secuencial vs aleatorio . . . 362
13.3.2 E/S de bloques de caracteres . . . 363
13.3.3 Operaciones de posicionamiento . . . 365
13.4 Ficheros binarios y de texto 367
13.4.1 Conversiones de caracteres especiales . . . 368
13.4.2 Ficheros binarios que contienen texto . . . 371
13.4.3 Ficheros binarios, E/S por bloques y acceso aleatorio . . . 372
13.5 Archivos, registros y campos 378
13.5.1 Registros de longitud fija o variable . . . 378
13.5.2 Operaciones con registros . . . 379
13.5.3 E/S de objetos de una clase . . . 383
13.6 Flujos asociados a *string* 386
13.7 Problemas 389

14 C++11/14 . . . 391
14.1 Introducción 391
14.2 Tipos fundamentales y enumerados 392
14.2.1 Enumerados con ámbito . . . 393
14.2.2 Nuevos literales numéricos en *C++14* . . . 393
14.3 Nuevos tipos carácter, cadenas y *string* 394
14.3.1 Literales de tipo carácter . . . 394
14.3.2 Literales cadena de caracteres . . . 396
14.3.3 Tipos *string* . . . 398
14.4 Semántica de movimiento 401
14.4.1 lvalues y rvalues en C++98 . . . 402
14.4.2 Lvalues y rvalues en C++11/14 . . . 406
14.5 Clases en C++11/14 413
14.5.1 Funciones miembro generadas por defecto . . . 413
14.5.2 Constructores . . . 416
14.6 Algunos aspectos adicionales 419
14.6.1 Especificación de excepciones . . . 419
14.6.2 Expresiones constantes en compilación . . . 420
14.6.3 Literales de usuario . . . 421

A Solución a los ejercicios . . . 423
A.1 Representación binaria 423
A.2 Tipos básicos: Bits 425

A.3 Vectores y matrices 431
A.4 Estructuras y vectores-C 441
A.5 Punteros 444
A.6 Memoria dinámica 452
A.7 Funciones 455
A.8 Errores 459
A.9 Clases: conceptos básicos 463
A.10 Clases: diseño 468
A.11 Clases: avanzado 473
A.12 Sobrecarga de operadores 482
A.13 E/S. Acceso aleatorio 492
A.14 C++11/14 499

B Tablas 503
B.1 Tabla ASCII 503
B.2 Operadores C++ 504
B.3 Palabras reservadas de C89, C99, C11, C++ y C++11 506

Bibliografía 509

Índice alfabético 513

Prólogo

El objetivo de este libro es presentar conceptos de *Metodología de la programación* como un curso básico de programación. No debería considerarse un curso sobre un lenguaje concreto —en nuestro caso C++— sino una exposición principalmente motivada por conceptos que un estudiante de primer curso debería conocer.

Un futuro profesional debería conocer la forma en que se puede crear un sistema que resuelve un problema partiendo de las operaciones más básicas que nos ofrece el hardware: cómo podemos ascender de las operaciones más simples que implementa un ordenador hasta un diseño complejo que es capaz de resolver, de adaptarse y de mejorar de una forma simple y eficaz a pesar de su complejidad.

Este libro ofrece un curso que permite a un estudiante de primeros cursos empezar a comprender esta metodología, preparándolo para abordar cursos más avanzados y especializados que permitan completar una formación integral para enfrentarse a problemas de desarrollo software de mayor tamaño.

Aprender a programar: Contexto del curso

Aprender a programar no es una tarea simple. Aunque muchos cursos o tutoriales se oferten con mensajes del tipo *"aprenda a programar en X días"*, la formación de un programador debe pasar por distintos cursos que le ocuparán mucho más tiempo, desde cursos básicos hasta cursos más avanzados y especializados.

Este libro corresponde a un segundo curso de programación. Se supone que el estudiante ya ha asimilado unos contenidos básicos de programación donde se incluyan los conceptos fundamentales sobre programación estructurada, tipos de datos simples y compuestos, incluso una introducción a la compilación separada (véase la referencia Garrido[14]). Con esos conocimientos, el estudiante ya es capaz de resolver problemas simples y de entender cómo funciona un programa, cuál es el flujo de control de un programa y cómo se pueden abstraer algunos algoritmos en módulos.

Este segundo curso es de profundización. Ahora se estudiarán en profundidad los detalles más básicos de los tipos, es decir, mostraremos detalles más cercanos a la máquina, pero se alcanzarán

niveles de abstracción mucho más altos. El estudiante incluirá temas más complejos, como la reserva de recursos en memoria dinámica, pero se enfrentará a ellos proponiendo soluciones que permitan disponer de tipos y módulos que abstraen los detalles importantes para formular las soluciones fácilmente a pesar de la complejidad de los programas.

Como resultado final de este curso, el alumno debería entender mejor cómo la metodología de la programación nos permite enfrentarnos a problemas de mayor tamaño creando soluciones eficaces, simples de desarrollar y de modificar. Lógicamente, serán necesarios otros cursos para completar la formación como programador. Sin embargo, las bases obtenidas después de este curso lo sitúan en un nivel de conocimiento ideal para empezar a abordar contenidos más avanzados y especializados. Algunos de los aspectos que podemos destacar son:

- El estudiante ha trabajado con tipos de datos básicos y ha profundizado en el problema de la codificación, la representación de información en memoria o la memoria dinámica. Estos aspectos facilitan estudiar temas relacionados con programación a bajo nivel, más cercana al sistema operativo o incluso que interactúe con sistemas hardware para los que la API se formule en términos muy cercanos al hardware.
- El estudiante ha trabajado en profundidad el problema del encapsulamiento. Es uno de los pilares de la programación dirigida a objetos. Es el momento ideal para ampliar la capacidad de diseño incluyendo temas de herencia o polimorfismo. Además, se han incluido discusiones sobre la forma en que se usa la jerarquía de clases que ofrece el lenguaje para flujos, por lo que el estudiante debería haber comenzado a asimilar la facilidad o conveniencia de este tipo de programación.
- Se han propuesto distintas soluciones para los mismos problemas, incluyendo comentarios que se refieren a la eficiencia en memoria y en tiempo. Por ejemplo, se comentan distintos algoritmos de búsqueda y ordenación. Es un buen momento para entrar en detalle en el problema de diseñar algoritmos eficientes. Ya sabe lo importante que puede ser cambiar un algoritmo, ahora debería aprender a diseñar mejores algoritmos.
- La abstracción de datos muestra claramente cómo podemos ocultar la representación de un tipo de dato mediante una interfaz. Una de las decisiones que hay que tomar es la elección de la representación del tipo. Al ser un curso básico, se ha trabajado sólo con representaciones sencillas basadas fundamentalmente en vectores. Es el momento ideal para proponer nuevas formas de resolver los problemas, como son los árboles u otras estructuras no lineales.
- Se ha enfatizado continuamente la idea de la abstracción como una herramienta fundamental. Se habla de abstracción funcional, se enfatiza la abstracción de la representación incluso en los tipos simples, se crean clases que ofertan una interfaz para abstraer los aspectos relevantes del nuevo tipo. Es muy sencillo e incluso natural abordar nuevas formas de abstracción, como la programación genérica.

Podríamos incluir otros comentarios, pues es fácil enlazar las implicaciones de un curso básico de programación con cursos más avanzados. Sin embargo, basta con este conjunto de aspectos para poner de relieve los conceptos básicos que se presentan en este libro, así como la conveniencia como base ideal para garantizar los mejores resultados en los cursos más avanzados.

Con estas consideraciones, este libro podría situarse como una segunda parte de un curso básico de fundamentos de programación, o un curso intermedio que pone los cimientos para poder abordar con garantías temas más avanzados y especializados de programación.

El lenguaje de programación

Los contenidos del libro se desarrollarán sobre el lenguaje de programación C++. No se debe interpretar que está diseñado para enseñar este lenguaje, aunque la amplia variedad de situaciones y discusiones nos permite interpretar el libro como una exposición razonada de las herramientas que

nos ofrece este lenguaje. No se mostrarán todos los detalles, aunque sí las partes más importantes y habituales que le permitirán empezar a usarlo para resolver sus propios problemas.

Lógicamente, el lenguaje C++ será un componente fundamental del libro. Inevitablemente, gran parte del contenido se dedicará a entender los detalles sintácticos y las particularidades del lenguaje, pero no deja de ser una forma de asimilar y afianzar los conocimientos que se pretenden adquirir durante el curso. Al finalizar, el estudiante debería estar preparado tanto para desarrollar soluciones en este lenguaje como para adaptarse a uno nuevo.

La elección de este lenguaje como vehículo para alcanzar los objetivos de este libro no es casual. La capacidad que C++ tiene para expresar soluciones a muy bajo nivel hasta diseños a un nivel de abstracción muy alto —incluyendo diseños basados en la programación dirigida a objetos, programación genérica o metaprogramación— lo convierten en una herramienta ideal para poder describir cómo el desarrollo del software nos permite avanzar desde las soluciones más simples hasta complejos sistemas. El estudiante puede estudiar cómo resolver problemas que van desde los bits y la gestión de recursos muy cercana al sistema hasta discusiones a nivel de jerarquías de clases sin necesidad de cambiar de lenguaje.

Esta amplia gama de discusiones no se hace a un bajo costo. Efectivamente, el lenguaje puede parecer complejo, e incluso poco recomendable para un primer curso. Sin embargo, los objetivos del curso no son simples de conseguir y, aunque le parezcan ambiciosos, resultan ideales como una base donde apoyarse para cursos más avanzados de programación. Para compensar esta dificultad, este libro no hace una exposición guiada solamente por el lenguaje, sino siguiendo un esquema más pedagógico asumiendo que el estudiante es relativamente novato. Aprovechamos el lenguaje para situar una más amplia base de formación para el estudiante, aunque el curso resulte algo más complejo. Note que otras referencias bibliográficas hacen una exposición mucho más exhaustiva —véase Josuttis[17] o Stroustrup[37]— recorriendo todos los aspectos del lenguaje y que son ideales para programadores formados interesados en aprender un nuevo lenguaje en su totalidad.

Los estudios universitarios de Ingeniería Informática

La razón principal que ha motivado la creación de este libro ha sido la necesidad de ofrecer un documento práctico que facilite tanto la docencia del profesor como el aprendizaje del alumno en los cursos de programación de los estudios de *Ingeniería Informática*. La raíz de esta necesidad se sitúa en la dificultad de encontrar referencias bibliográficas que se adapten a los planes de estudios.

La evolución de los planes de estudios en los últimos 30 años ha desembocado en un modelo de enseñanza en el que el estudiante tiene una gran responsabilidad: por un lado, se han ampliado los contenidos de la carrera, mientras que por otro se han disminuido el tiempo de enseñanza en el aula. Una parte del esfuerzo debería realizarse de forma autodidacta. Los estudiantes deben asimilar una gran cantidad de contenidos en poco tiempo, por lo que resulta muy difícil disponer de un buen material de estudio si se limita únicamente a los apuntes de clase. Por ello, es necesario disponer de algún libro de apoyo.

Trabajar los contenidos de la asignatura a partir de varios libros implica un sobreesfuerzo que no es simple para un alumno de primer curso. Un libro que se adapte a los contenidos de la asignatura es ideal para obtener el mejor resultado, dejando las consultas bibliográficas en libros especializados para temas más puntuales, así como el modelo de trabajo autónomo para cursos más avanzados.

Como resultado, el libro constituye una buena referencia no sólo para el estudiante de estos estudios, sino para el lector que quiera introducirse en el mundo de la programación de forma autodidacta.