

# Contenido

Prefacio	v
<b>Capítulo P Introducción a C++</b>	<b>1</b>
<b>P.1 El ambiente de C++</b>	<b>2</b>
Archivos de inclusión	3
El preprocesador	3
El compilador de C++	3
El linker	3
Funciones, clases y objetos	3
La directiva <code>#include</code>	4
El enunciado <code>using</code> y <code>namespace</code>	5
Función principal	6
Ejecución de un programa en C++	6
Ejercicios para la sección P.1	7
<b>P.2 Directivas y macros del preprocesador</b>	<b>7</b>
Eliminar los comentarios	7
Macros	8
Compilación condicional	9
Más sobre la directiva <code>#include</code>	10
Ejercicios para la sección P.2	11
<b>P.3 Enunciados de control en C++</b>	<b>12</b>
Enunciados de secuencia y compuestos	12
Control de selección y repetición	12
Enunciados <code>if</code> anidados	14
El enunciado <code>switch</code>	15
Ejercicios para la sección P.3	16
<b>P.4 Tipos de datos primitivos y tipos de clase</b>	<b>16</b>
Tipos de datos primitivos	16
Variables de tipo primitivo	19
Operadores	19
Incremento de sufijo y prefijo	21
Compatibilidad y conversión de tipo	22
Ejercicios para la sección P.4	23
<b>P.5 Objetos, apuntadores y referencias</b>	<b>23</b>
Periodo de vida de los objetos	23
Apuntadores	24
El Apuntador Nulo	27
Objetos creados en forma dinámica	27
Referencias	29
Ejercicios para la sección P.5	29

<b>P.6</b>	<b>Funciones</b>	<b>29</b>
	Declaraciones de función	30
	Llamada por valor versus llamada por referencia	30
	Funciones de operador	31
	Funciones miembro de clase y parámetros implícitos	32
	Ejercicios para la sección P.6	32
<b>P.7</b>	<b>Arreglos y cadenas en C</b>	<b>33</b>
	Equivalencia del apuntador de un arreglo	34
	Argumentos del arreglo	36
	Literales de cadena y cadenas en C	36
	Arreglos multidimensionales	36
	Ejercicios para la sección P.7	37
<b>P.8</b>	<b>La clase string</b>	<b>38</b>
	Las cadenas en realidad son plantillas	43
	Ejercicios para la sección P.8	44
<b>P.9</b>	<b>Uso de entrada/salida con series</b>	<b>44</b>
	Entrada/salida en la consola	44
	Series de entrada	45
	Series de salida	49
	Dar formato a la salida con el uso de manipuladores de E/S	50
	Los formatos de punto flotante y la precisión	52
	Series de archivos	54
	openmode	54
	Series de cadenas	57
	Ejercicios para la sección P.9	58
	Repaso del capítulo	59
<b>Capítulo 1 Introducción al diseño del software</b>		<b>63</b>
<b>1.1</b>	<b>El ciclo de vida del software</b>	<b>64</b>
	Modelos del ciclo de vida del software	65
	Actividades relativas al ciclo de vida del software	67
	Especificación de los requerimientos	68
	Análisis	69
	Diseño	70
	Ejercicios para la sección 1.1	72
<b>1.2</b>	<b>Utilizando la abstracción para manejar la complejidad</b>	<b>73</b>
	Abstracción procedural	73
	Abstracción de datos	73
	Ocultar información	74
	Ejercicios para la sección 1.2	75
<b>1.3</b>	<b>Definición de las clases C++</b>	<b>75</b>
	Definición de clase	75
	Las partes <code>public</code> y <code>private</code>	77
	Implementación de la clase	79
	Utilizando la clase <code>clock</code>	81

La clase Person	81	
Constructores	85	
Funciones miembro modificador y accesorio	87	
Operadores	88	
Amigos	89	
Implementación de la clase Person	89	
El parámetro <code>this</code>	91	
Una aplicación que usa la clase Person	92	
Las clases como componentes de otras clases	94	
Campos de datos del arreglo	94	
Estilo de la documentación para clases y funciones	96	
Ejercicios para la sección 1.3	98	
<b>1.4 Tipos de datos abstractos, interfaces, y pre y poscondiciones</b>		<b>98</b>
Tipos de datos abstractos (ADT) e interfaces	99	
Una ADT para una clase de directorio telefónico	99	
Contratos y ADT	100	
Precondiciones y poscondiciones	100	
Ejercicios para la sección 1.4	101	
<b>1.5 Análisis de los requerimientos, casos de uso y diagramas de secuencia</b>		<b>102</b>
<i>Caso práctico:</i> Diseño de un programa de directorio telefónico	102	
Ejercicios para la sección 1.5	108	
<b>1.6 Diseño de un directorio telefónico basado en un arreglo</b>		<b>108</b>
<i>Caso práctico:</i> Diseño de un programa de directorio telefónico (continuación)	108	
Ejercicios para la sección 1.6	113	
<b>1.7 Implementación y prueba del directorio telefónico basado en un arreglo</b>		<b>114</b>
<i>Caso práctico:</i> Diseño de un programa de directorio telefónico (continuación)	114	
Ejercicios para la sección 1.7	121	
<b>1.8 Completando la aplicación del directorio telefónico</b>		<b>121</b>
<i>Caso práctico:</i> Diseño de un programa de directorio telefónico (continuación)	121	
Ejercicios para la sección 1.8	125	
Repaso del capítulo	125	

---

<b>Capítulo 2 Exactitud y eficiencia del programa</b>		<b>129</b>
---	--	------------

<b>2.1 Defectos y errores de programa ("bugs")</b>		<b>130</b>
Errores de sintaxis	131	
Errores en tiempo de ejecución	131	
Errores lógicos	136	
Ejercicios para la sección 2.1	137	
<b>2.2 Excepciones</b>		<b>138</b>
Formas de indicar un error	138	
El enunciado <code>throw</code>	138	
Excepciones no capturadas	139	
Atrapar y manejar excepciones con bloques <code>try</code> y <code>catch</code>	140	
Excepciones estándar	141	
Jerarquía de clase de excepción	145	

	Todas las excepciones de catching	147	
	Ejercicios para la sección 2.2	147	
2.3	<b>Prueba de programas</b>		148
	Structured Walkthroughs	148	
	Niveles y tipos de prueba	149	
	Preparaciones para la prueba	151	
	Prueba de recomendaciones para sistemas de programa	152	
	Desarrollo de los datos de prueba	152	
	Prueba de las condiciones límite	153	
	¿Quién hace la prueba?	156	
	Stubs	156	
	Manejadores (drivers)	157	
	Prueba de una clase	157	
	Uso de un marco de prueba	158	
	Prueba de regresión	159	
	Prueba de integración	159	
	Ejercicios para la sección 2.3	160	
2.4	<b>Depuración de un programa</b>		160
	Uso de un depurador (debugger)	162	
	Ejercicios para la sección 2.4	164	
2.5	<b>Razonamiento acerca de programas: afirmaciones e invariantes de ciclo</b>		166
	Afirmaciones	166	
	Invariantes de ciclo	167	
	El macro <code>assert</code> de C++	168	
	Ejercicios para la sección 2.5	169	
2.6	<b>Eficiencia de algoritmos</b>		170
	Notación Big-O	172	
	Comparación de desempeño	176	
	Algoritmos con tasas de crecimiento exponenciales y factoriales	178	
	Ejercicios para la sección 2.6	178	
	Repaso del capítulo	179	
<b>Capítulo 3 Herencia y jerarquías de clase</b>			<b>185</b>
3.1	<b>Introducción a la herencia y las jerarquías de clase</b>		186
	Relaciones <i>Is-a</i> contra <i>Has-a</i>	187	
	Una clase base y una clase derivada	188	
	Inicializar campos de datos en una clase derivada	190	
	El constructor sin parámetro	191	
	Visibilidad protegida para campos de datos de clase base	192	
	Ejercicios para la sección 3.1	192	
3.2	<b>Anulación de la función miembro, sobrecarga de la función miembro y polimorfismo</b>		193
	Anulación de la función miembro	193	
	Sobrecarga de la función miembro	194	
	Funciones virtuales y polimorfismo	196	

	Ejercicios para la sección 3.2	201	
3.3	<b>Clases abstractas, asignación y conversión en una jerarquía</b>		202
	Referencia a objetos reales	203	
	Resumen de características de clases reales y clases abstractas	204	
	Asignaciones en una jerarquía de clase	204	
	Conversión en una clase jerárquica	205	
	<i>Caso práctico:</i> Presentación de direcciones para diferentes países	206	
	Ejercicios para la sección 3.3	209	
3.4	<b>Herencia múltiple</b>		210
	Refactorización de las clases <code>Employee</code> y <code>Student</code>	212	
	Ejercicios para la sección 3.4	212	
3.5	<b>Nombres de espacios y visibilidad</b>		213
	Nombres de espacios	213	
	Declaración de un nombre de espacio	213	
	Nombre de espacio global	214	
	La declaración <code>using</code> y la directiva <code>using</code>	215	
	Uso de la visibilidad para apoyar la encapsulación	217	
	Ejercicios para la sección 3.5	219	
3.6	<b>Una jerarquía de clase <code>Shape</code></b>		220
	<i>Caso práctico:</i> Procesamiento de formas geométricas	220	
	Ejercicios para la sección 3.6	224	
	Repaso del capítulo	225	
<b>Capítulo 4 Contenedores secuenciales</b>			<b>231</b>
4.1	<b>Clases de plantilla y el vector</b>		232
	Vector	233	
	Especificación de la clase <code>vector</code>	235	
	Función <code>at</code> y el operador de subíndice	236	
	Ejercicios para la sección 4.1	237	
4.2	<b>Aplicaciones de vector</b>		238
	Nuevo tratamiento de la aplicación del directorio telefónico	239	
	Ejercicios para la sección 4.2	239	
4.3	<b>Implementación de una clase <code>vector</code></b>		240
	Constructor por omisión	240	
	Función <code>swap</code>	241	
	Operador de subíndice	242	
	Función <code>push_back</code>	243	
	Función <code>insert</code>	244	
	Función <code>erase</code>	244	
	Función <code>reserve</code>	245	
	Desempeño del <code>KW: :vector</code>	246	
	Ejercicios para la sección 4.3	246	
4.4	<b>Constructor de copia, operador de asignación y destructor</b>		247
	Copia de objetos y el constructor de copia	247	

	Copia superficial <i>versus</i> copia profunda	247
	Operador de asignación	250
	Destructor	251
	Ejercicios para la sección 4.4	252
4.5	<b>Listas simplemente ligadas y doblemente ligadas</b>	<b>252</b>
	Nodo de lista	255
	Conexión de nodos	256
	Inserción de un nodo en una lista	257
	Eliminación de un nodo	257
	Travesía de una lista ligada	258
	Listas doblemente ligadas	258
	Creación de un objeto de lista doblemente ligada	262
	Listas circulares	262
	Ejercicios para la sección 4.5	262
4.6	<b>La clase <code>list</code> y el iterador</b>	<b>264</b>
	La clase <code>list</code>	264
	El iterador	264
	Requisitos comunes para iteradores	267
	Ejercicios para la sección 4.6	271
4.7	<b>Implementación de una clase de lista doblemente ligada</b>	<b>271</b>
	Implementación de las funciones <code>KW::list</code>	272
	Implementación del <code>iterator</code>	278
	<code>Const_iterator</code>	283
	Ejercicios para la sección 4.7	284
4.8	<b>Aplicación de la clase <code>list</code></b>	<b>285</b>
	<i>Caso práctico:</i> Mantener una lista ordenada	285
	Ejercicios para la sección 4.8	291
4.9	<b>Contenedores de la biblioteca estándar</b>	<b>292</b>
	Características comunes de los contenedores	293
	Secuencias	294
	Contenedores asociativos	295
	De nuevo el tema de la implementación del vector	295
	Ejercicios para la sección 4.9	296
4.10	<b>Algoritmos de la biblioteca estándar y objetos de función</b>	<b>297</b>
	Función <code>find</code>	297
	Biblioteca de algoritmos	299
	Función <code>swap</code>	302
	Especialización de la función <code>swap</code>	302
	Objetos de función	303
	Ejercicios para la sección 4.10	306
	Repaso del capítulo	307

---

**Capítulo 5 Pilas (Stacks)**

311

5.1	<b>La pila ADT</b>	
	Especificación de la pila ADT	312

312

	Ejercicios para la sección 5.1	315	
<b>5.2</b>	<b>Aplicaciones de pilas</b>		<b>315</b>
	<i>Caso práctico:</i> Localización de palíndromos	315	
	<i>Caso práctico:</i> Probar expresiones para paréntesis equilibrados	320	
	Ejercicios para la sección 5.2	325	
<b>5.3</b>	<b>Ejecución de una pila</b>		<b>325</b>
	Clases adaptadoras y patrón de delegación	326	
	Otro tratamiento del archivo de definición <code>stack.h</code>	327	
	Implementación de una pila como una estructura de datos ligada	329	
	Comparación de implementaciones de pila	331	
	Ejercicios para la sección 5.3	331	
<b>5.4</b>	<b>Más aplicaciones de la pila</b>		<b>332</b>
	<i>Caso práctico:</i> Evaluación de expresiones sufijo	333	
	<i>Caso práctico:</i> Convertir de infijo a sufijo	339	
	<i>Caso práctico:</i> Parte 2: Conversión de expresiones con paréntesis	347	
	Ejercicios para la sección 5.4	350	
	Repaso del capítulo	351	

---

**Capítulo 6 Colas y deque** **357**

<b>6.1</b>	<b>El tipo de datos abstractos cola</b>		<b>358</b>
	Una cola de clientes	358	
	Una cola de impresión	359	
	Inconveniencia de una "pila de impresión"	359	
	Especificación de una cola ADT	359	
	Ejercicios para la sección 6.1	362	
<b>6.2</b>	<b>Mantenimiento de una cola de clientes</b>		
	<i>Caso práctico:</i> Mantenimiento de una cola	362	
	Ejercicios para la sección 6.2	365	
<b>6.3</b>	<b>Implementación de la cola ADT</b>		<b>365</b>
	Uso de <code>std::list</code> como un recipiente para una cola	365	
	Uso de una lista simplemente ligada para implementar la cola ADT	367	
	El uso de un arreglo circular para almacenaje en una cola	370	
	Revisión del diseño	370	
	Comparación de las tres implementaciones	375	
	Ejercicios para la sección 6.3	376	
<b>6.4</b>	<b>La deque</b>		<b>376</b>
	Especificación de la deque	376	
	Implementación de la deque por medio de una arreglo circular	376	
	Implementación de la biblioteca estándar de la deque	378	
	Ejercicios para la sección 6.4	380	
<b>6.5</b>	<b>Simulación de filas de espera por medio de colas</b>		<b>381</b>
	<i>Caso práctico:</i> Simular una estrategia para atender a pasajeros de aerolíneas	381	
	Ejercicios para la sección 6.5	398	
	Repaso del capítulo	398	

<b>Capítulo 7</b>	<b>Recursión</b>	<b>403</b>
7.1	<b>Pensamiento recursivo</b>	<b>404</b>
	Pasos para diseñar un algoritmo recursivo	406
	Demostración de que una función recursiva es correcta	408
	Rastreo de una función recursiva	409
	La pila y los marcos de activación	409
	Ejercicios para la sección 7.1	411
7.2	<b>Definiciones recursivas de fórmulas matemáticas</b>	<b>412</b>
	Recursión <i>versus</i> iteración	415
	Recursión de extremo o recursión de última línea	416
	Eficiencia de recursión	416
	Ejercicios para la sección 7.2	419
7.3	<b>Búsqueda recursiva</b>	<b>420</b>
	Diseño de un algoritmo de búsqueda lineal recursiva	420
	Implementación de la búsqueda lineal	420
	Diseño de un algoritmo de búsqueda binaria	421
	Eficiencia de la búsqueda binaria	423
	Implementación de la búsqueda binaria	424
	Prueba de búsqueda binaria	426
	Ejercicios para la sección 7.3	426
7.4	<b>Resolución de problemas con recursión</b>	<b>426</b>
	<i>Caso práctico:</i> Torres de Hanoi	427
	<i>Caso práctico:</i> Conteo de celdas en un blob	431
	Ejercicios para la sección 7.4	434
7.5	<b>Backtracking</b>	<b>435</b>
	<i>Caso práctico:</i> Hallar una trayectoria por un laberinto	436
	Ejercicios para la sección 7.5	439
	Repaso del capítulo	440
<b>Capítulo 8</b>	<b>Árboles</b>	<b>445</b>
8.1	<b>Terminología de árboles y aplicaciones</b>	<b>447</b>
	Terminología de árboles	447
	Árboles binarios	448
	Algunos tipos de árboles binarios	449
	Llenos y completos	451
	Árboles generales	452
	Ejercicios para la sección 8.1	453
8.2	<b>Travesías de árboles</b>	<b>454</b>
	Visualización de las travesías de un árbol	455
	Travesías de árboles de búsqueda binaria y de árboles de expresión	455
	Ejercicios para la sección 8.2	456
8.3	<b>Implementación de una clase Binary_Tree</b>	<b>457</b>
	La clase BTNode	457

	La clase <code>Binary_Tree</code>	458	
	Constructor de copia, asignación y destructor	465	
	Ejercicios para la sección 8.3	465	
8.4	<b>Árboles de búsqueda binaria</b>		466
	Revisión de un árbol de búsqueda binaria	466	
	Rendimiento	468	
	La clase <code>Binary_Search_Tree</code>	468	
	Inserción en un árbol de búsqueda binaria	472	
	Eliminación de un árbol de búsqueda binaria	474	
	Prueba de un árbol de búsqueda binaria	479	
	<i>Caso práctico:</i> Escribir un índice para un trabajo final de investigación	479	
	Ejercicios para la sección 8.4	484	
8.5	<b>Montones y colas de prioridad</b>		484
	Inserción de un elemento en el montón	485	
	Eliminación de un elemento de un montón	486	
	Implementación del montón	486	
	Colas de prioridad	489	
	La clase <code>priority_queue</code>	490	
	Uso de un montón como base para la cola de prioridad	490	
	Diseño de una clase <code>KW::priority_queue</code>	491	
	Uso de la clase de función <code>Compare</code>	494	
	Ejercicios para la sección 8.5	495	
8.6	<b>Los árboles de Huffman</b>		496
	<i>Caso práctico:</i> Construcción de un árbol de Huffman a la medida	498	
	Ejercicios para la sección 8.6	504	
	Repaso del capítulo	505	

---

**Capítulo 9 Conjuntos y mapeos** **511**

9.1	<b>Requerimientos del contenedor asociativo</b>		512
	La abstracción del conjunto	512	
	Las funciones de <code>set</code>	513	
	El <code>multiset</code> (multiconjunto)	518	
	Biblioteca estándar clase <code>pair</code>	519	
	Ejercicios para la sección 9.1	520	
9.2	<b>Mapeos y multimapeos</b>		521
	Las funciones de <code>map</code>	522	
	Definición de la función <code>Compare</code>	528	
	El multimapeo	528	
	Ejercicios para la sección 9.2	530	
9.3	<b>Tablas hash</b>		530
	Códigos hash y cálculos del índice	531	
	Funciones para la generación de los códigos hash	532	
	Direccionamiento abierto	533	
	Lectura secuencial de la tabla hash	535	

	Borrar un objeto utilizando el direccionamiento abierto	536
	Reducción de la colisión ampliando el tamaño de la tabla	536
	Reducción de la colisión mediante sondeo cuadrático	537
	Problemas con el sondeo cuadrático	538
	Encadenamiento (chaining)	538
	Rendimiento de las tablas hash	539
	Ejercicio para la sección 9.3	541
9.4	<b>Implementación de la tabla hash</b>	542
	El ADT (TAD tipo de dato abstracto) KW: :hash_map	542
	El Entry_Type	542
	La clase hash_map tal y como se implementa utilizando Hash_Table_Open.h	544
	La clase hash_map tal y como es implementada por Hash_Table_Chain.h	551
	Probando las implementaciones de la tabla hash	553
	Ejercicios para la sección 9.4	554
9.5	<b>Consideraciones de implementación para el hash_map</b>	555
	Definición de la clase de función hash	555
	El hash_map::iterator y el hash_map::const_iterator	556
	Ejercicios para la sección 9.5	558
9.6	<b>Aplicaciones adicionales de mapeo</b>	558
	<i>Caso práctico:</i> La implementación del directorio telefónico utilizando un mapeo	558
	<i>Caso práctico:</i> Completar el problema del Código Huffman	561
	Ejercicios para la sección 9.6	564
	Repaso del capítulo	564

<b>Capítulo 10</b>	<b>Ordenación</b>	<b>569</b>
10.1	<b>El uso de las funciones de ordenación C++</b>	570
	Ejercicios para la sección 10.1	572
10.2	<b>Ordenación por selección</b>	572
	Análisis de la ordenación por selección	573
	Código de ordenación por selección con iteradores	574
	Código para la ordenación de un arreglo	576
	Ejercicios de la sección 10.2	577
10.3	<b>Ordenación de burbuja</b>	577
	Análisis de la ordenación de burbuja	579
	Código para la ordenación de burbuja	579
	Ejercicios para la sección 10.3	580
10.4	<b>Ordenación por inserción</b>	581
	Análisis de la ordenación por inserción	582
	Código para la ordenación por inserción	583
	Utilización de iterator_traits para determinar el tipo de información de un elemento	584
	Ejercicios para la sección 10.4	586

10.5	<b>Comparación de clasificaciones cuadráticas</b>	586
	Las comparaciones contra los intercambios	588
	Ejercicios para la sección 10.5	588
10.6	<b>Shell sort: una mejor ordenación por inserción</b>	588
	Análisis del Shell sort	590
	El código para Shell sort	590
	Ejercicios para la sección 10.6	592
10.7	<b>Merge Sort</b>	592
	Análisis del merge	593
	El código del merge	593
	Algoritmo para merge sort	595
	Rastreo del algoritmo merge sort	595
	Análisis del merge sort	596
	Código para el merge sort	597
	Ejercicios para la sección 10.7	598
10.8	<b>Heapsort</b>	599
	Algoritmo heapsort	599
	Algoritmo para la construcción de un heap	601
	Análisis del algoritmo heapsort	601
	Código para heapsort	601
	Ejercicios para la sección 10.8	604
10.9	<b>Quicksort</b>	604
	Algoritmo para el quicksort	605
	Análisis de quicksort	605
	Código para quicksort	606
	Algoritmo para la partición	607
	Código para <code>partition</code>	608
	Algoritmo revisado de la partición	610
	Código para función revisada <code>partition</code>	612
	Ejercicios para la sección 10.9	614
10.10	<b>Prueba de los algoritmos de ordenación</b>	614
	Ejercicios para la sección 10.10	616
10.11	<b>El problema de la bandera nacional alemana (tema opcional)</b>	616
	<i>Caso práctico:</i> El problema de la bandera nacional alemana	617
	Ejercicios para la sección 10.11	620
	Repaso del capítulo	620

---

**Capítulo 11 Árboles de búsqueda autobalanceados**

623

11.1	<b>Equilibrio y rotación del árbol</b>	624
	Por qué es importante el equilibrio	624
	La rotación	625
	Algoritmo para la rotación	625
	Implementación de la rotación	627
	Ejercicios para la sección 11.1	628

11.2	<b>Árboles AVL</b>	628
	Equilibrio de un árbol izquierdo-izquierdo	629
	Equilibrio de un árbol izquierdo-derecho	630
	Cuatro tipos de árboles seriamente desbalanceados	632
	Implementación de un árbol AVL	634
	Inserción en un árbol AVL	636
	Eliminación de un elemento de un árbol AVL	642
	El rendimiento de los árboles AVL	642
	Ejercicios para la sección 11.2	643
11.3	<b>Árboles rojo-negro</b>	643
	La inserción en un árbol rojo-negro	644
	Eliminación de un elemento de un árbol rojo-negro	655
	Rendimiento de un árbol rojo-negro	655
	Ejercicios para la sección 11.3	656
11.4	<b>Árboles 2-3</b>	656
	Búsqueda de un árbol 2-3	657
	Inserción de un elemento en el árbol 2-3	658
	Análisis de los árboles 2-3 y comparación con los árboles binarios balanceados	661
	Eliminación de un elemento de un árbol 2-3	661
	Ejercicios para la sección 11.4	663
11.5	<b>Árboles 2-3-4 y árboles B</b>	663
	Árboles 2-3-4	663
	Implementación de la clase <code>Two_Three_Four_Tree</code>	666
	Relación entre los árboles 2-3-4 y los árboles rojo-negro	671
	Los árboles B	672
	Ejercicios para la sección 11.5	680
	Repaso del capítulo	681

---

**Capítulo 12 Grafos**

691

12.1	<b>Terminología de los grafos</b>	692
	Representación visual de los grafos	692
	Grafos dirigidos y no dirigidos	693
	Trayectorias y ciclos	694
	Relación entre grafos y árboles	696
	Aplicaciones de los grafos	696
	Ejercicios para la sección 12.1	697
12.2	<b>Un grafo ADT y la clase <code>Edge</code></b>	697
	Ejercicios para la sección 12.2	701
12.3	<b>Implementación del grafo ADT</b>	701
	Lista de adyacencia	701
	Matriz de adyacencia	702
	Generalidades de la jerarquía	703
	La clase <code>Graph</code>	704
	La clase <code>List_Graph</code>	707

La clase <code>Matrix_Graph</code>	713	
Comparación de implementaciones	713	
Ejercicios para la sección 12.3	715	
12.4 Travesías de grafos		715
Búsqueda de primero en anchura ( <code>breadth-first</code> )	715	
Búsqueda de primero en profundidad ( <code>depth-first</code> )	720	
Ejercicios para la sección 12.4	726	
12.5 Aplicaciones de travesías de grafos		727
<i>Caso práctico</i> : La trayectoria más corta a través de un laberinto	727	
<i>Caso práctico</i> : Ordenamiento topológico de un grafo	731	
Ejercicios para la sección 12.5	733	
12.6 Algoritmos que utilizan grafos ponderados		734
Búsqueda de la trayectoria más corta de un vértice a todos los demás vértices	734	
Árboles de mínima expansión	738	
Ejercicios para la sección 12.6	742	
Repaso del capítulo	743	

---

**Apéndice A Temas avanzados de C++** **755**

A.1 El conjunto de caracteres fuente, trigrafos, dígrafos y palabras clave alternas		755
A.2 El allocator		756
A.3 Características		757
Estructura básica de la clase características	759	
La clase <code>char_traits</code>	759	
A.4 Clases de base virtual		759
Reacomodo de las clases <code>Employee</code> y <code>Student</code>	759	
Clases de base virtual	763	
A.5 Apuntadores inteligentes		764
El <code>auto_ptr</code>	765	
La <code>shared_ptr</code>	765	

---

**Apéndice B Revisión del UML** **769**

B.1 El diagrama de clase		770
Representación de las clases y de las interfases	770	
Generalización	773	
Clases internas o anidadas	774	
Asociación	774	
Composición	775	
Clases genéricas (o de plantilla)	776	
B.2 Diagramas de secuencia		776
Eje del tiempo	776	
Objetos	776	
Líneas de vida	778	

Barras de activación 778  
Mensajes 778  
Uso de notas 778

**Apéndice C El marco de prueba de la CppUnit 779**

---

**Glosario 783**

---

**Índice 795**

---