

CONTENIDO



UNIVERSIDAD NACIONAL DE ENTRE RÍOS
FACULTAD DE INGENIERÍA
CENTRO DE MEDICINA
BIBLIOTECA

PARTE I FUNDAMENTOS DE PROGRAMACIÓN

1. Introducción a la ciencia de la computación y a la programación	3
1.1. ¿Qué es una computadora?	3
1.2. Organización física de una computadora (hardware)	4
1.2.1. Dispositivos de Entrada/Salida (E/S)	5
1.2.2. La memoria central (interna)	5
1.2.3. Unidad Central de Proceso (UCP)	7
1.2.4. El microprocesador	7
1.2.5. Memoria auxiliar (externa)	8
1.2.6. Las computadoras multimedia	9
1.2.7. Módem	9
1.2.8. La computadora personal ideal para programación	10
1.3. Concepto de algoritmo	10
1.3.1. Características de los algoritmos	11
1.4. El software (los programas)	12
1.5. Los lenguajes de programación	13
1.5.1. Instrucciones a la computadora	13
1.5.2. Lenguajes máquina	14
1.5.3. Lenguajes de bajo nivel	14
1.5.4. Lenguajes de alto nivel	15
1.5.5. Traductores de lenguaje	15
1.5.6. La compilación y sus fases	16
1.6. La resolución de problemas con computadora	17
1.6.1. Análisis del problema	18
1.6.2. Diseño del algoritmo	19
1.6.3. Herramientas de programación	19
1.6.4. Codificación de un programa	20
1.6.5. Compilación y ejecución de un programa	21
1.6.6. Prueba y depuración de un programa	21
1.6.7. Documentación y mantenimiento	22
1.7. Programación modular	22
1.8. Programación estructurada	23
1.8.1. Recursos abstractos	23
1.8.2. Diseño descendente (<i>top-down</i>)	24
1.8.3. Estructuras de control	24
1.8.4. Teorema de la programación estructurada: estructuras básicas	24
1.9. Representación gráfica de los algoritmos	25
*1.9.1. Diagramas de flujo	25
1.10. Diagramas de Nassi-Schneiderman (N-S)	30
1.11. Una breve historia de C++	31
RESUMEN	32
EJERCICIOS	32
PROBLEMAS	33
2. El lenguaje C++. Elementos básicos	37
2.1. Estructura general de un programa en C++	37
2.1.1. Directivas del preprocesador	39
2.1.2. Declaraciones globales	40
2.1.3. Función main()	40
2.1.4. Funciones definidas por el usuario	41
2.1.5. Comentarios	41
2.2. Creación de un programa	43
2.3. El proceso de ejecución de un programa en C++	44
2.4. Depuración de un programa en C++	45
2.4.1. Errores de sintaxis	46
2.4.2. Errores lógicos	46
2.4.3. Errores de regresión	47
2.4.4. Mensajes de error	47
2.4.5. Errores en tiempo de ejecución	47
2.5. Pruebas	48
2.6. Los elementos de un programa en C++	48
*2.6.1. Tokens (elementos léxicos de los programas) ..	48
*2.6.2. Identificadores	49
*2.6.3. Palabras reservadas	49

2.6.4.	Comentarios	49	3.7.3.	Operadores de direcciones	75
2.6.5.	Signos de puntuación y separadores	49	3.8.	Operador condicional	76
2.6.6.	Archivos de cabecera	50	3.9.	Operador coma	76
2.6.7.	El nuevo estándar C++ (ANSI C++)	50	3.10.	Operadores especiales (), [] y ::	77
2.7.	Tipos de datos en C++	50	3.10.1.	El operador ()	77
2.7.1.	Enteros (int)	51	3.10.2.	El operador []	77
2.7.2.	Tipos de coma flotante (float/double)	52	3.10.3.	El operador ::	77
2.7.3.	Caracteres (char)	52	3.11.	El operador sizeof	77
2.8.	El tipo de dato bool	53	3.12.	Conversiones de tipos	78
2.8.1.	Simulación del tipo bool	53	3.12.1.	Conversión implícita	78
2.8.2.	Escritura de valores bool	54	3.12.2.	Reglas	78
2.9.	Constantes	54	3.12.3.	Conversiones explícitas	78
2.9.1.	Constantes literales	54	3.13.	Prioridad y asociatividad	79
2.9.2.	Constantes definidas (simbólicas)	56	RESUMEN		79
2.9.3.	Constantes enumeradas	56	EJERCICIOS		80
2.9.4.	Constantes declaradas const y volatile	57	PROBLEMAS		81
2.10.	Variables	58	4. Estructuras de selección: sentencias if y switch		85
2.10.1.	Declaración	58	4.1.	Estructuras de control	85
2.10.2.	Inicialización de variables	59	4.2.	La sentencia if	86
2.10.3.	Declaración o definición	59	4.3.	Sentencia if de dos alternativas: if-else	87
2.11.	Duración de una variable	59	4.4.	Sentencias if-else anidadas	89
2.11.1.	Variables locales	60	4.4.1.	Sangría en las sentencias if anidadas	89
2.11.2.	Variables globales	60	4.4.2.	Comparación de sentencias if anidadas y secuencias de sentencias if	90
2.11.3.	Variables dinámicas y de objetos	60	4.5.	Sentencia de control switch	91
2.12.	Entradas y salidas	61	4.5.1.	Caso particular case	94
2.12.1.	Salida (cout)	61	4.5.2.	Uso de sentencias switch en menús	94
2.12.2.	Entrada (cin)	62	4.6.	Expresiones condicionales: el operador ?	94
RESUMEN		63	4.7.	Evaluación en cortocircuito de expresiones lógicas	95
EJERCICIOS		63	4.8.	Puesta a punto de programas	95
3. Operadores y expresiones		65	4.9.	Errores frecuentes de programación	96
3.1.	Operadores y expresiones	65	RESUMEN		97
3.2.	Operador de asignación	66	EJERCICIOS		98
3.3.	Operadores aritméticos	66	PROBLEMAS		99
3.3.1.	Asociatividad	67	5. Estructuras de control: bucles		101
3.3.2.	Uso de paréntesis	68	5.1.	La sentencia while	101
3.4.	Operadores de incrementación y decrementación	69	5.1.1.	Operadores de incremento y decremento (++, --)	103
3.5.	Operadores relacionales	70	5.1.2.	Terminaciones anormales de un ciclo	104
3.6.	Operadores lógicos	72	5.1.3.	Diseño eficiente de bucles	104
3.6.1.	Evaluación en cortocircuito	73	5.1.4.	Bucles while con cero iteraciones	104
3.6.2.	Asignaciones <i>booleanas</i> (lógicas)	74	5.1.5.	Bucles controlados por centinela	104
3.7.	Operadores de manipulación de bits	74			
3.7.1.	Operadores de asignación adicionales	75			
3.7.2.	Operadores de desplazamiento de bits (>>, <<)	75			

5.1.6.	Bucles controlados por indicadores (banderas) .	105	6.7.1.	Ámbito del programa	138
5.1.7.	La sentencia break en los bucles	106	6.7.2.	Ámbito del archivo fuente	139
5.1.8.	Bucles while (true)	106	6.7.3.	Ámbito de una función	139
5.2.	Repetición: el bucle for	107	6.7.4.	Ámbito de bloque	139
5.2.1.	Diferentes usos de bucle for	109	6.7.5.	Variables locales	139
5.3.	Precauciones en el uso de for	110	6.8.	Clases de almacenamiento	140
5.3.1.	Bucles infinitos	111	6.8.1.	Variables automáticas	140
5.3.2.	Los bucles for vacíos	111	6.8.2.	Variables externas	140
5.3.3.	Sentencias nulas en bucles for	112	6.8.3.	Variables registro	140
5.3.4.	Sentencias break y continue	112	6.8.4.	Variables estáticas	140
5.4.	Repetición: el bucle do-while	113	6.9.	Concepto y uso de funciones de biblioteca	141
5.4.1.	Diferencias entre while y do-while	114	6.10.	Funciones de carácter	141
5.5.	Comparación de bucles while, for y do-while	114	6.10.1.	Comprobación alfabética y de dígitos	142
5.6.	Diseño de bucles	115	6.10.2.	Funciones de prueba de caracteres especiales	142
5.6.1.	Bucles para diseño de sumas y productos	115	6.10.3.	Funciones de conversión de caracteres	143
5.6.2.	Fin de un bucle	115	6.11.	Funciones numéricas	143
5.6.3.	Otras técnicas de terminación de bucle	116	6.11.1.	Funciones matemáticas	143
5.6.4.	Bucles for vacíos	117	6.11.2.	Funciones trigonométricas	144
5.7.	Bucles anidados	117	6.11.3.	Funciones logarítmicas y exponenciales	144
RESUMEN		119	6.11.4.	Funciones aleatorias	144
EJERCICIOS		120	6.12.	Funciones de fecha y hora	145
PROYECTOS DE PROGRAMACIÓN		122	6.13.	Funciones de utilidad	146
PROBLEMAS		123	6.14.	Visibilidad de una función	147
6. Funciones		125	6.14.1.	Variables locales frente a variables globales	147
6.1.	Concepto de función	126	6.14.2.	Variables estáticas y automáticas	149
6.2.	Estructura de una función	127	6.15.	Compilación separada	150
6.2.1.	Nombre de una función	128	6.16.	Variables registro (<i>register</i>)	151
6.2.2.	Tipo de dato de retorno	128	6.17.	Sobrecarga de funciones (polimorfismo)	151
6.2.3.	Resultados de una función	128	6.17.1.	¿Cómo determina C++ la función sobrecargada correcta?	152
6.2.4.	Llamada a una función	129	6.18.	Recursividad	153
6.3.	Prototipos de las funciones	130	6.19.	Plantillas de funciones	154
6.3.1.	Prototipos con un número no especificado de parámetros	132	6.19.1.	Utilización de las plantillas de funciones	155
6.4.	Parámetros de una función	132	6.19.2.	Plantillas de función min y max	156
6.4.1.	Paso de parámetros por valor	132	RESUMEN		156
6.4.2.	Paso de parámetros por referencia	133	EJERCICIOS		157
6.4.3.	Diferencia entre los parámetros por valor y por referencia	134	PROBLEMAS		158
6.4.4.	Parámetros const de una función	135	7. Arrays (listas y tablas)		161
6.5.	Argumentos por omisión	135	7.1.	Arrays	161
6.6.	Funciones en línea (<i>Inline</i>)	137	7.1.1.	Declaración de un array	162
6.6.1.	Creación de funciones en línea	138	7.1.2.	Subíndices de un array	162
6.7.	Ámbito (alcance)	138	7.1.3.	Almacenamiento en memoria de los arrays	163
			7.1.4.	El tamaño de los arrays (sizeof)	163

7.1.5.	Verificación del rango del índice de un array . . .	163
7.2.	Inicialización (iniciación) de un array	164
7.3.	Arrays de caracteres y cadenas de texto	165
7.4.	Arrays multidimensionales	166
7.4.1.	Inicialización de arrays multidimensionales . . .	167
7.4.2.	Acceso a los elementos de arrays bidimensionales	168
7.4.3.	Lectura y escritura de elementos de arrays bidimensionales	169
7.4.4.	Acceso a elementos mediante bucles	169
7.4.5.	Arrays de más de dos dimensiones	169
7.4.6.	Una aplicación práctica	169
7.5.	Utilización de arrays como parámetros	170
7.5.1.	Precauciones	172
7.5.2.	Paso de cadenas como parámetros	173
7.6.	Ordenación de listas	173
7.6.1.	Algoritmo de la burbuja	174
7.7.	Búsqueda en listas	175
7.7.1.	Búsqueda secuencial	175
	RESUMEN	177
	EJERCICIOS	177
	PROBLEMAS	178
8.	Estructuras y uniones	183
8.1.	Estructuras	183
8.1.1.	Declaración de una estructura	184
8.1.2.	Definición de variables de estructuras	184
8.1.3.	Uso de estructuras en asignaciones	185
8.1.4.	Inicialización de una declaración de estructuras	185
8.1.5.	El tamaño de una estructura	185
8.2.	Acceso a estructuras	186
8.2.1.	Almacenamiento de información en estructuras	186
8.2.2.	Lectura de información de una estructura	187
8.2.3.	Recuperación de información de una estructura	187
8.3.	Estructuras anidadas	187
8.3.1.	Un ejemplo de estructuras anidadas	188
8.4.	Arrays de estructuras	189
8.4.1.	Arrays como miembros	190
8.5.	Utilización de estructuras como parámetros	191
8.6.	Uniones	191
8.7.	Enumeraciones	192
8.7.1.	Visibilidad en una estructura: ¿pública o privada?.	193
8.7.2.	Operador sizeof de una unión	194
8.7.3.	Operador typedef	195

RESUMEN	195
EJERCICIOS	196
9. Punteros (apuntadores)	197
9.1. Direcciones y referencias	197
9.1.1. Referencias	198
9.2. Concepto de puntero (apuntador)	199
9.2.1. Declaración de punteros	200
9.2.2. Inicialización (iniciación) de punteros	200
9.2.3. Indirección de punteros	200
9.2.4. Punteros y verificación de tipos	201
9.3. Punteros <i>null</i> y <i>void</i>	202
9.4. Puntero a puntero	202
9.5. Punteros y arrays	203
9.5.1. Nombres de arrays como punteros	203
9.5.2. Ventajas de los punteros	204
9.6. Arrays de punteros	204
9.6.1. Inicialización de un array de punteros a cadenas.. . . .	205
9.7. Punteros de cadenas	205
9.7.1. Punteros frente a arrays	205
9.8. Aritmética de punteros	206
9.8.1. Una aplicación de punteros	206
9.9. Punteros constantes frente a punteros a constantes	207
9.9.1. Punteros constantes	207
9.9.2. Punteros a constantes	208
9.9.3. Punteros constantes a constantes	208
9.10. Punteros como argumentos de funciones	209
9.10.1. Paso por referencia frente a paso por direcciones	209
9.11. Punteros a funciones	210
9.11.1. Inicialización de un puntero a una función.	211
9.11.2. Otra aplicación	212
9.11.3. Arrays de punteros de funciones	213
9.11.4. Una aplicación práctica	213
9.12. Punteros a estructuras	214
RESUMEN	215
EJERCICIOS	215
PROBLEMAS	216
10. Asignación dinámica de memoria	217
10.1. Gestión dinámica de la memoria	217
10.1.1. Almacén libre (<i>free store</i>).	218
10.1.2. Ventajas de la asignación dinámica de memoria en C++	218
10.2. El operador <i>new</i>	219

10.2.1.	Asignación de memoria de un tamaño desconocido	221
10.2.2.	Inicialización de memoria con un valor	221
10.2.3.	Uso de new para arrays multidimensionales	221
10.3.	El operador <i>delete</i>	222
10.4.	Ejemplos que utilizan <i>new</i> y <i>delete</i>	222
10.5.	Asignación de memoria para arrays	223
10.5.1.	Asignación de memoria interactivamente	223
10.5.2.	Asignación de memoria para un array de estructuras	223
10.6.	Arrays dinámicos	224
10.7.	Gestión del desbordamiento de memoria: <i>set_new_handler</i>	225
10.8.	Reglas de funcionamiento de <i>new</i> y <i>delete</i>	226
RESUMEN.	227
EJERCICIOS	227
PROBLEMAS	228

11. Cadenas 229

11.1.	Concepto de cadena	229
11.1.1.	Declaración de variables de cadena	230
11.1.2.	Inicialización de variables de cadena	230
11.2.	Lectura de cadenas	231
11.2.1.	Funciones miembro cin	232
11.2.2.	Función cin.get()	233
11.2.3.	Función cout.put()	233
11.2.4.	Funciones cin.putback() y cin.ignore()	234
11.2.5.	Función cin.peek()	234
11.3.	La biblioteca string.h	234
11.3.1.	La palabra reservada const	235
11.4.	Arrays y cadenas como parámetros de funciones	236
11.4.1.	Uso del operador de referencia para tipos array	236
11.4.2.	Uso de punteros para pasar una cadena	236
11.5.	Asignación de cadenas	237
11.5.1.	La función strncpy	237
11.6.	Longitud y concatenación de cadenas	238
11.6.1.	La función strlen()	238
11.6.2.	La función strcat y strncat	238
11.7.	Comparación de cadenas	239
11.7.1.	La función strcmp	239
11.7.2.	La función strcmp	240
11.7.3.	La función strncmp	240
11.7.4.	La función strnicmp	240
11.8.	Inversión de cadenas	240

11.9.	Conversión de cadenas	241
11.9.1.	Funciónstrupr	241
11.9.2.	Funciónstrlwr	241
11.10.	Conversión de cadenas a números	241
11.10.1.	La función atoi	241
11.10.2.	La función atof	242
11.10.3.	La función atol	242
11.11.	Búsqueda de caracteres y cadenas	242
11.11.1.	La función strchr()	242
11.11.2.	La función strchr()	243
11.11.3.	La función strstr()	243
11.11.4.	La función strstrn	243
11.11.5.	La función strpbrk	243
11.11.6.	La función strstr	244
11.11.7.	La función strtok	244
RESUMEN.	244
EJERCICIOS	245
PROBLEMAS	246

12. Ordenación y búsqueda 247

12.1.	Algoritmos de ordenación básicos	247
12.2.	Ordenación por intercambio	248
12.3.	Ordenación por selección	249
12.3.1.	Algoritmo de selección	250
12.3.2.	Análisis del algoritmo de ordenación por selección	251
12.4.	Ordenación por inserción	251
12.4.1.	Algoritmo de inserción	252
12.4.2.	Análisis del algoritmo de ordenación por inserción	252
12.5.	Ordenación por burbuja	252
12.5.1.	Algoritmo de la burbuja	253
12.5.2.	Análisis del algoritmo de la burbuja	255
12.6.	Búsqueda en listas: búsqueda secuencial y binaria	255
12.6.1.	Búsqueda secuencial	255
12.6.2.	Búsqueda binaria	257
12.7.	Análisis de los algoritmos de búsqueda	258
12.7.1.	Complejidad de la búsqueda secuencial	258
12.7.2.	Análisis de la búsqueda binaria	259
12.7.3.	Comparación de la búsqueda binaria y secuencial	259
RESUMEN.	260
EJERCICIOS	260
PROBLEMAS	261

PARTE II	
PROGRAMACIÓN AVANZADA EN C++	
13. Clases	265
13.1. Clases y objetos	265
13.1.1. ¿Qué son objetos?	265
13.1.2. ¿Qué son clases?	266
13.2. Definición de una clase	266
13.2.1. Objetos de clases	267
13.2.2. Acceso a miembros de la clase: encapsulamiento	268
13.2.3. Funciones miembro	269
13.2.4. Tipos de funciones miembro	270
13.2.5. Funciones en línea y fuera de línea	270
13.2.6. La palabra reservada inline	271
13.2.7. Nombres de parámetros de funciones miembro	272
13.2.8. Implementación de clases	272
13.2.9. Archivos de cabecera y de clases	272
13.3. Constructores	273
13.3.1. Constructor por defecto	274
13.3.2. Constructores alternativos	274
13.3.3. Constructores sobrecargados	275
13.3.4. Constructor de copia	275
13.3.5. Inicialización de miembros en constructores	275
13.4. Destructores	276
13.4.1. Clases compuestas	277
13.5. Errores de programación frecuentes	277
RESUMEN	280
LECTURAS RECOMENDADAS	281
EJERCICIOS	281
PROBLEMAS	284
14. Excepciones	285
14.1. Condiciones de error en programas	285
14.1.1. ¿Por qué considerar las condiciones de error?	285
14.2. El tratamiento de los códigos de error	286
14.3. Manejo de excepciones en C++	287
14.4. El mecanismo de manejo de excepciones	287
14.4.1. El modelo de manejo de excepciones	288
14.4.2. Diseño de excepciones	289
14.4.3. Bloques try	289
14.4.4. Lanzamiento de excepciones	290
14.4.5. Captura de una excepción: catch	291
14.5. Especificación de excepciones	293
14.6. Excepciones imprevistas	295
14.7. Aplicaciones prácticas de manejo de excepciones	295
14.7.1. Calcular las raíces de una ecuación de segundo grado	295
14.7.2. Control de excepciones en una estructura tipo pila	296
RESUMEN	297
EJERCICIOS	298
15. Sobrecarga de operadores	299
15.1. Sobrecarga	299
15.2. Operadores unitarios	300
15.3. Sobrecarga de operadores unitarios	301
15.3.1. Versiones prefija y postfija de los operadores ++ y --	303
15.3.2. Sobrecargar un operador unitario como función miembro	304
15.3.3. Sobrecarga de un operador unitario como una función amiga	304
15.3.4. Operadores de incremento y decremento	305
15.4. Operadores binarios	306
15.5. Sobrecarga de operadores binarios	306
15.5.1. Sobrecarga de un operador binario como función miembro	307
15.5.2. Sobrecarga de un operador binario como una función amiga	309
15.6. Operadores + y -	309
15.7. Sobrecarga de operadores de asignación	310
15.7.1. Sobrecargando el operador de asignación	311
15.7.2. Operadores como funciones miembro	312
15.7.3. Operador []	313
15.7.4. Sobrecargando el operador de llamada a funciones ()	314
15.8. Sobrecarga de operadores de inserción y extracción	314
15.8.1. Sobrecarga de flujo de salida	314
15.8.2. Sobrecarga de flujo de entrada	315
15.9. Clase cadena	316
15.9.1. Clase cadena (string)	316
15.9.2. Funciones amigas	317
15.10. Sobrecarga de new y delete: asignación dinámica	319
15.10.1. Sobrecarga de new	319
15.10.2. Sobrecarga del operador delete	320
15.11. Conversión de datos y operadores de conversión forzada de tipos	320

15.11.1.	Conversión entre tipos básicos	320
15.11.2.	Conversión entre objetos y tipos básicos	320
15.11.3.	Funciones de conversión	321
15.11.4.	Constructores de conversión	322
15.12.	Manipulación de sobrecarga de operadores	322
15.13.	Una aplicación de sobrecarga de operadores	323
RESUMEN.	325
LECTURAS RECOMENDADAS.	325
EJERCICIOS	326
16. Genericidad: plantillas (<i>templates</i>)	327	
16.1.	Genericidad	327
16.2.	Conceptos fundamentales de plantillas en C++	328
16.3.	Plantillas de funciones	328
16.3.1.	Fundamentos teóricos	328
16.3.2.	Definición de plantilla de función	329
16.3.3.	Un ejemplo de plantilla de funciones	330
16.3.4.	Un ejemplo de función plantilla	332
16.3.5.	Plantillas de función <i>ordenar y buscar</i>	332
16.3.6.	Una aplicación práctica	332
16.3.7.	Problemas en las funciones plantilla	333
16.4.	Plantillas de clases	333
16.4.1.	Definición de una plantilla de clase	334
16.4.2.	Instanciación de una plantilla de clases	335
16.4.3.	Utilización de una plantilla de clase	335
16.4.4.	Argumentos de plantillas	336
16.4.5.	Aplicaciones de plantillas de clases	336
16.5.	Una plantilla para manejo de pilas de datos	337
16.5.1.	Definición de las funciones miembro	338
16.5.2.	Utilización de una clase plantilla	338
16.5.3.	Instanciación de una clase plantilla con clases	340
16.5.4.	Uso de las plantillas de funciones con clases	340
16.6.	Plantillas frente a polimorfismo	341
RESUMEN.	341
EJERCICIOS	342
17. Flujos	343	
17.1.	Flujos (<i>streams</i>)	343
17.1.1.	Flujos de texto	344
17.1.2.	Flujos binarios	344
17.1.3.	Las clases de flujo E/S	345
17.1.4.	Archivos de cabecera	345
17.2.	La biblioteca de clases <i>iostream</i>	345
17.2.1.	La clase <i>streambuf</i>	346

17.2.2.	Jerarquía de clases ios	346
17.2.3.	Flujos estándar	346
17.2.4.	Entradas/salidas en archivos	346
17.2.5.	Entradas/salidas en un buffer de memoria	347
17.2.6.	Archivos de cabecera	347
17.2.7.	Entrada/salida de caracteres y flujos	347
17.3.	Clases <i>istream</i> y <i>ostream</i>	347
17.3.1.	Clase <i>istream</i>	347
17.3.2.	La clase <i>ostream</i>	349
17.4.	Salida a la pantalla y a la impresora	350
17.4.1.	Operadores de inserción en cascada	351
17.4.2.	Las funciones miembro <i>put()</i> y <i>write()</i>	351
17.4.3.	Impresión de la salida en una impresora	352
17.5.	Lectura del teclado	352
17.5.1.	Lectura de datos carácter	353
17.5.2.	Lectura de datos cadena	354
17.5.3.	Funciones miembro <i>get()</i> y <i>getline()</i>	354
17.5.4.	La función <i>getline</i>	357
17.5.5.	Problemas de utilización de <i>getline</i>	358
17.6.	Formateado de salida	359
17.7.	Manipuladores	359
17.7.1.	Bases de numeración	359
17.7.2.	Anchura de campos	361
17.7.3.	Rellenado de caracteres	361
17.7.4.	Precisión de números reales	362
17.8.	Indicadores de formato	362
17.8.1.	Uso de <i>setiosflags()</i> y <i>resetiosflags()</i>	362
17.8.2.	Las funciones miembro <i>setf()</i> y <i>unsetf()</i>	364
RESUMEN.	364
EJERCICIOS	365

PARTE III

ESTRUCTURA DE DATOS

18. Listas enlazadas	369	
18.1.	Fundamentos teóricos	369
18.1.1.	Clasificación de las listas enlazadas	370
18.2.	Operaciones en listas enlazadas	371
18.2.1.	Declaración de un nodo	371
18.2.2.	Puntero de cabecera y cola	371
18.2.3.	El puntero nulo	372
18.2.4.	El operador \rightarrow de selección de un miembro	372
18.2.5.	Construcción de una lista	373

18.2.6.	Insertar un elemento en una lista	374	21. Árboles	413
18.2.7.	Búsqueda de un elemento.	376	21.1. Árboles generales	413
18.3.	Lista doblemente enlazada	377	21.1.1. Terminología	414
18.3.1.	Declaración de una lista doblemente enlazada.	378	21.1.2. Representación de un árbol	416
RESUMEN.	378	21.2. Resumen de definiciones	417
EJERCICIOS		379	21.3. Árboles binarios	417
PROBLEMAS		379	21.3.1. Equilibrio	418
19. Pilas y colas		381	21.3.2. Árboles binarios completos	419
19.1. Concepto de pila		381	21.4. Estructura de un árbol binario.	420
19.1.1. Especificación de una pila		382	21.4.1. Diferentes tipos de representaciones en C++	421
19.2. La clase <i>pila</i> implementada con arrays.		382	21.4.2. Operaciones en árboles binarios.	422
19.2.1. Especificación de la clase pila		383	21.5. Árboles de expresión.	422
19.2.2. Implementación		384	21.5.1. Reglas para construcción de árboles de expresión	424
19.2.3. Operaciones de verificación del estado de la pila		385	21.6. Recorrido de un árbol	425
19.3. Colas		385	21.6.1. Recorrido <i>preorden</i>	426
19.4. La clase <i>cola</i> implementada con arrays.		386	21.6.2. Recorrido <i>enorden</i>	427
19.4.1. Definición de la especificación de una cola.		386	21.6.3. Recorrido <i>postorden</i>	428
19.4.2. Especificación de la clase cola.		387	21.6.4. Profundidad de un árbol binario.	430
19.4.3. Implementación de la clase cola		387	21.7. Árbol binario de búsqueda	430
19.4.4. Operaciones de la cola		388	21.7.1. Creación de un árbol binario	431
19.5. Implementación de una pila con una lista enlazada		389	21.7.2. Implementación de un nodo de un árbol binario de búsqueda	432
RESUMEN.		391	21.8. Operaciones en árboles binarios de búsqueda.	432
EJERCICIOS		391	21.8.1. Búsqueda	432
PROBLEMAS		392	21.8.2. Insertar un nodo	433
20. Recursividad		393	21.8.3. Insertar nuevos nodos.	433
20.1. La naturaleza de la recursividad		393	21.8.4. Eliminación	434
20.2. Funciones recursivas		395	21.8.5. Recorrido de un árbol.	435
20.2.1. Funciones mutuamente recursivas		397	21.8.6. Determinación de la altura de un árbol	435
20.2.2. Condición de terminación de la recursión		397	21.9. Aplicaciones de árboles en algoritmos de exploración	435
20.3. Recursión <i>versus</i> iteración		397	21.9.1. Visita a los nodos de un árbol.	435
20.3.1. Directrices en la toma de decisión: iteración/recursión		399	RESUMEN.	436
20.4. Recursión infinita		399	EJERCICIOS	437
20.5. Resolución de problemas con recursión		401	PROBLEMAS	438
20.5.1. Torres de Hanoi		401	REFERENCIAS BIBLIOGRÁFICAS	439
20.5.2. Búsqueda binaria recursiva		404	22. Archivos	441
20.6. Ordenación rápida (<i>quicksort</i>).		405	22.1. Archivos C++	441
20.6.1. Algoritmo <i>quicksort</i> en C++		407	22.2. Apertura de archivos	442
20.6.2. Análisis del algoritmo <i>quicksort</i>		408	22.2.1. Apertura de un archivo sólo para entrada.	443
RESUMEN.		409	22.3. E/S en archivos	443
EJERCICIOS		410	22.3.1. La función <i>open</i>	443
PROBLEMAS		411	22.3.2. La función <i>close</i>	445
			22.4. Lectura y escritura de archivos de texto	446

22.5.	E/S binaria.	447
22.5.1.	Funciones miembro get y put.	447
22.5.2.	Función put().	448
22.5.3.	Formato 2 de get().	448
22.5.4.	Funciones read y write.	449
22.6.	Acceso aleatorio.	450
RESUMEN.	452
EJERCICIOS	453

PARTE IV

PROGRAMACIÓN ORIENTADA A OBJETOS

23.	Conceptos fundamentales de programación orientada a objetos.	457
23.1.	¿Qué es la programación orientada a objetos?	457
23.1.1.	El objeto.	458
23.1.2.	Ejemplos de objetos.	458
23.1.3.	Métodos y mensajes.	459
23.1.4.	Clases.	460
23.2.	Un mundo de objetos.	460
23.2.1.	Definición de objetos.	461
23.2.2.	Identificación de objetos.	461
23.2.3.	Duración de los objetos.	462
23.2.4.	Objetos frente a clases. Representación gráfica (Notación de Ege).	462
23.2.5.	Datos internos.	463
23.2.6.	Ocultación de datos.	464
23.3.	Comunicaciones entre objetos: los mensajes.	464
23.3.1.	Activación de objetos.	465
23.3.2.	Mensajes.	465
23.3.3.	Paso de mensajes.	466
23.4.	Estructura interna de un objeto.	466
23.4.1.	Atributos.	466
23.4.2.	Métodos.	467
23.5.	Clases.	467
23.5.1.	Una comparación con tablas de datos.	467
23.6.	Herencia.	468
23.6.1.	Tipos de herencia.	470
23.6.2.	Herencia simple (<i>herencia jerárquica</i>).	470
23.6.3.	Herencia múltiple (<i>herencia en malla</i>).	471
23.6.4.	Clases abstractas.	471
23.6.5.	Anulación/sustitución.	472
23.7.	Sobrecarga.	472
23.8.	Ligadura dinámica.	473

23.8.1.	Funciones o métodos virtuales.	473
23.9.	Objetos compuestos.	474
23.9.1.	Un ejemplo de objetos compuestos.	475
23.9.2.	Niveles de profundidad.	475
23.10.	Reutilización con orientación a objetos.	476
23.10.1.	Objetos y reutilización.	476
23.11.	Polimorfismo.	477
RESUMEN.	477
EJERCICIOS	478

24.	Clases derivadas: herencia.	479
24.1.	Clases derivadas.	479
24.1.1.	Declaración de una clase derivada.	481
24.1.2.	Consideraciones de diseño.	482
24.2.	Tipos de herencia.	482
24.2.1.	Herencia pública.	483
24.2.2.	Herencia privada.	484
24.2.3.	Herencia protegida.	485
24.2.4.	Operador de resolución de ámbito.	485
24.2.5.	Constructores-inicializadores en herencia.	486
24.2.6.	Sintaxis del constructor.	487
24.2.7.	Sintaxis de la implementación de una función miembro.	487
24.3.	Destructores.	487
24.4.	Herencia múltiple.	488
24.4.1.	Características de la herencia múltiple.	489
24.4.2.	Dominación (prioridad).	490
24.4.3.	Inicialización de la clase base.	491
24.5.	Ligadura.	492
24.6.	Funciones virtuales.	493
24.6.1.	Ligadura dinámica mediante funciones virtuales.	493
24.7.	Polimorfismo.	495
24.7.1.	El polimorfismo sin ligadura dinámica.	495
24.7.2.	El polimorfismo con ligadura dinámica.	496
24.8.	Uso del polimorfismo.	496
24.9.	Ligadura dinámica frente a ligadura estática.	497
24.10.	Ventajas del polimorfismo.	497
RESUMEN.	498
EJERCICIOS	498

25.	Biblioteca de plantillas estándar STL.	501
25.1.	Biblioteca de STL: conceptos clave.	501
25.1.1.	Archivos de cabecera.	502

25.2.	Clases contenedoras	503
25.2.1.	Tipos de contenedores	504
25.2.2.	Contenedores secuenciales	504
25.2.3.	Contenedores asociativos	505
25.2.4.	Adaptadores de contenedores	505
25.3.	Iteradores	506
25.3.1.	Categorías de iteradores	507
25.3.2.	Comportamiento de los iteradores	508
25.3.3.	Iteradores puntero	508
25.3.4.	Iteradores definidos en cada contenedor	509
25.3.5.	Iteradores constantes	509
25.4.	Contenedores estándar	510
25.5.	Vector	510
25.5.1.	Declaración de vectores	510
25.5.2.	Una aplicación sencilla de vector <T>	510
25.5.3.	Uso de vectores	511
25.6.	Lista	511
25.6.1.	Declaración de una lista	512
25.6.2.	Uso de listas	512
25.7.	Deque (doble cola)	512
25.7.1.	Declaración de deques	513
25.7.2.	Uso de deques	513
25.7.3.	Una aplicación de una deque	513
25.7.4.	Funciones miembro de deque <T>	514
25.8.	Contenedores asociativos: <i>set</i> y <i>multiset</i>	514
25.8.1.	Declaración de conjuntos	514
25.8.2.	Uso de conjuntos	514
25.8.3.	Uso de mapas y multimapas	515
25.9.	Contenedores adaptadores	516

25.9.1.	<i>Stack</i> (pila)	516
25.9.2.	<i>Queue</i> (cola)	517
25.9.3.	Cola de prioridad	517
25.10.	Algoritmos	517
25.10.1.	Algoritmos de ordenación	518
25.10.2.	Algoritmos numéricos	519
25.10.3.	Algoritmos de secuencia no mutables	520
25.10.4.	Algoritmos de secuencia mutable	521
	REFERENCIAS BIBLIOGRÁFICAS	523

APÉNDICES

A.	C frente a C++	527
B.	Guía de sintaxis de ANSI/ISO Standar C++	563
C.	Operadores (<i>prioridad</i>)	595
D.	Palabras reservadas ISO/ANSI C++	597
E.	Códigos de caracteres ASCII	611
F.	Biblioteca de funciones estándar ANSI/ISO C++ y Borland C++ 5.0	615
G.	Biblioteca de clases ANSI/ISO C++	665
H.	Glosario	679
I.	Recursos (Libros/Revistas/URL de Internet C++)	691
J.	Bibliografía	697
	ÍNDICE	703