

Premessa *XXIII*

Capitolo 1 **Introduzione e storia del linguaggio C++** *1*

- 1.1 Organizzazione e struttura del testo *1*
- 1.2 Librerie di funzioni e di classi *2*
- 1.3 Il C++ come evoluzione del C *3*
- 1.4 Storia del C++ *4*
- 1.5 Il C++ e gli altri linguaggi *5*
- 1.6 I meccanismi sintattici *6*
- 1.7 Filosofia di progetto *8*
- 1.8 Il C++ nella didattica *9*

Parte I **Il C++ e la programmazione procedurale** *10*

Capitolo 2 **Un primo contatto con il C++** *11*

- 2.1 Premessa *11*
- 2.2 Dichiarazioni di variabili, di tipo e di costanti *11*
 - 2.2.1 Dichiarazioni di variabili, 11 – 2.2.2 Dichiarazioni di tipo, 12 – 2.2.3 Dichiarazioni di costanti, 12
- 2.3 Tipi semplici *12*
 - 2.3.1 Definizione di un tipo enumerativo, 12 – 2.3.2 Definizione del tipo booleano, 13
- 2.4 Tipo array *13*
 - 2.4.1 Definizione di un tipo array monodimensionale, 13 – 2.4.2 Definizione di un tipo array bidimensionale, 14 – 2.4.3 Definizione di un tipo stringa di caratteri, 14 – 2.4.4 Definizione di un tipo *struct*, 15
- 2.5 Istruzioni semplici *15*

- 2.5.1 Istruzione di assegnazione, 15 – 2.5.2 Chiamata di funzione e definizione di funzione, 16 – 2.5.3 Istruzione *goto*, 18
- 2.6 Istruzioni strutturate 18
 - 2.6.1 Istruzione composta, 18 – 2.6.2 Istruzioni condizionali (*if... then... else, case*), 19 – 2.6.3 Istruzioni iterative (*while, do-while e for*), 20
- 2.7 Definizione di puntatori 21
 - 2.7.1 Puntatori a variabili semplici, 21 – 2.7.2 Puntatori e variabili dinamiche, 22
- 2.8 Istruzioni di ingresso/uscita 22
- 2.9 Le classi del linguaggio C++ 23
 - 2.9.1 Definizione della specifica di una classe, 23 – 2.9.2 L'utente della classe, 24
- 2.10 Classi derivate 24
 - 2.10.1 Meccanismo base di derivazione, 24 – 2.10.2 Polimorfismo, 25 – 2.10.3 Funzioni e classi modello, 26
- 2.11 Programmi completi: struttura generale ed esempi 27

Capitolo 3 Variabili e tipi semplici 30

- 3.1 Le variabili 30
 - 3.1.1 Definizioni e dichiarazioni, 30 – 3.1.2 Allocazione in memoria della variabile, 30 – 3.1.3 Definizione della variabile, 30 – 3.1.4 Valore iniziale, 31 – 3.1.5 Modalità di allocazione e visibilità intermodulo, 31 – 3.1.6 Ulteriori specifiche di tipo, 31
- 3.2 Le costanti 31
 - 3.2.1 Costanti letterali, 31 – 3.2.2 Costanti con nome, 32 – 3.2.3 Costante simbolica, 33
- 3.3 Classificazione dei tipi del C++ 33
 - 3.3.1 I tipi semplici, 33 – 3.3.2 Il tipo puntatore, 34 – 3.3.3 I tipi strutturati, 34 – 3.3.4 I tipi sottoinsieme, 34
- 3.4 Tipo intero 34
 - 3.4.1 Descrizione, 34 – 3.4.2 Costanti del tipo intero, 35 – 3.4.3 Definizione della variabile di tipo intero, 35 – 3.4.4 Funzioni applicabili al tipo intero, 35 – 3.4.5 Rappresentazione dei valori interi, 36 – 3.4.6 Tipo intero senza segno, 37
- 3.5 Tipo reale 37
 - 3.5.1 Descrizione, 37 – 3.5.2 Uso del tipo reale, 37 – 3.5.3 Costanti del tipo reale, 37 – 3.5.4 Dichiarazione della variabile di tipo reale, 38 – 3.5.5 Funzioni applicabili al tipo reale, 38 – 3.5.6 Altre funzioni definibili sui reali, 38 – 3.5.7 Rappresentazione dei valori reali, 39
- 3.6 Tipo carattere 39
 - 3.6.1 Descrizione, 39 – 3.6.2 Costanti del tipo carattere, 39 – 3.6.3 Dichiarazione della variabile di tipo carattere, 40 – 3.6.4 Rappresentazione dei caratteri, 40 – 3.6.5 Funzioni e predicati per la manipolazione dei caratteri, 40
- 3.7 Tipo enumerativo 40
 - 3.7.1 Descrizione e dichiarazione, 40 – 3.7.2 Tipo enumerativo anonimo, 42 – 3.7.3 Uso del tipo enumerativo, 42 – 3.7.4 Operatori applicabili al tipo enumerativo, 42
- 3.8 Tipo booleano 42
 - 3.8.1 Descrizione, 42 – 3.8.2 Uso del tipo booleano, 43 – 3.8.3 Definizione del tipo booleano, 43 – 3.8.4 Funzioni applicabili al tipo booleano, 44
- 3.9 Tipo stringa di bit 44
- 3.10 Relazioni e predicati 44
 - 3.10.1 Descrizione, 44 – 3.10.2 Relazioni e operazioni logiche, 44
- 3.11 Funzioni per la trasformazione di tipo 45
- 3.12 Rappresentazione dei dati 45

- 3.12.1 Rappresentazione degli interi, 45 – 3.12.2 Rappresentazione dei reali, 46 – 3.12.3 Rappresentazione dei valori booleani, 48
- 3.13 Costanti letterali 48
 - 3.13.1 Costanti intere ottali ed esadecimali, 48 – 3.13.2 Costanti reali, 48 – 3.13.3 Costante carattere, 49 – 3.13.4 Costante enumerativa, 49 – 3.13.5 Stringa letterale, 49
- 3.14 Identificatori 50
- 3.15 Riepilogo 51

Capitolo 4 I tipi strutturati 52

- 4.1 Premessa 52
- 4.2 Il tipo array 52
 - 4.2.1 Descrizione, 52 – 4.2.2 Uso degli array, 53 – 4.2.3 Definizione di array, 54 – 4.2.4 Costruttore di array, 54 – 4.2.5 Array monodimensionale allocato staticamente, 54 – 4.2.6 Array bidimensionale allocato staticamente, 55
- 4.3 Il tipo record o struttura 56
 - 4.3.1 Descrizione, 56 – 4.3.2 Uso del tipo record, 56 – 4.3.3 Dichiarazione di record, 56 – 4.3.4 Funzioni di accesso al record, 57 – 4.3.5 Operazioni sul tipo record, 57 – 4.3.6 Esempi per il tipo record, 58
- 4.4 Il tipo unione 60
 - 4.4.1 Descrizione, 60 – 4.4.2 Dichiarazione del tipo unione e definizione di variabile, 60 – 4.4.3 Unione anonima, 61 – 4.4.4 Unione come record con variante, 61
- 4.5 Dimensioni delle strutture dati in C++ 62
- 4.6 Riepilogo 62

Capitolo 5 Le istruzioni 64

- 5.1 Classificazione delle istruzioni 64
- 5.2 Modalità di scrittura 64
- 5.3 Istruzioni di ingresso e di uscita 65
 - 5.3.1 Esempi di istruzioni di ingresso/uscita, 66
- 5.4 Istruzione di salto (*goto*) 67
- 5.5 Il costruito sequenza 68
- 5.6 Istruzioni condizionali 68
 - 5.6.1 Istruzione *if-then*, 68 – 5.6.2 Istruzione *if-then-else*, 69 – 5.6.3 Selezione multipla mediante il costruito *else-if*, 70 – 5.6.4 Istruzione di selezione multipla con costruito *switch-case*, 70 – 5.6.5 Esempi di istruzioni condizionali, 71
- 5.7 Costrutti di iterazione 74
 - 5.7.1 Generalità, 74 – 5.7.2 Ciclo a condizione iniziale, 74 – 5.7.3 Ciclo a condizione finale, 75 – 5.7.4 Ciclo a conteggio, 75 – 5.7.5 Forme incomplete del *for*, 76 – 5.7.6 Ciclo *for* con blocco vuoto, 76 – 5.7.7 Uso di espressioni multiple, 77 – 5.7.8 Definizione e visibilità della variabile di controllo, 77 – 5.7.9 Esempi di strutture di iterazione, 77
- 5.8 Altri costrutti per il controllo di sequenza 79
 - 5.8.1 Terminazione anticipata di una istruzione strutturata (*break*), 79 – 5.8.2 Istruzione di continuazione di un ciclo, 79 – 5.8.3 Cicli indefiniti, 80
- 5.9 Riepilogo 80

Capitolo 6 Le espressioni 83

- 6.1 Sintassi delle espressioni 83
 - 6.1.1 Aspetti generali, 83 – 6.1.2 Notazione utilizzata per gli operatori, 84 – 6.1.3 Precedenza fra gli operatori, 84 – 6.1.4 Associatività degli operatori, 85 – 6.1.5 Valutazione delle espressioni, 86 – 6.1.6 Ambiguità nella valutazione delle espressioni, 86 – 6.1.7 Valutazione parziale, 86 – 6.1.8 Operatore di assegnazione, 86
- 6.2 Espressioni particolari del C++ 87
 - 6.2.1 Operatori di assegnazione composta, 87 – 6.2.2 Operatori di incremento e decremento in forma prefissa e postfissa, 87
- 6.3 Espressioni miste 88
 - 6.3.1 Esempi di istruzioni di assegnazione, 88

Capitolo 7 Le funzioni 90

- 7.1 Premessa 90
- 7.2 Sottoprogrammi, procedure e funzioni 90
 - 7.2.1 Sottoprogrammi, 90 – 7.2.2 Le funzioni del C++, 91 – 7.2.3 Definizione e attivazione di una funzione, 91
- 7.3 Parametri della funzione 92
 - 7.3.1 Parametri di ingresso e uscita, 92 – 7.3.2 Modalità di corrispondenza, 92 – 7.3.3 Notazioni per la modalità della corrispondenza, 93 – 7.3.4 Puntatori parametri di scambio, 93 – 7.3.5 Casi particolari, 94 – 7.3.6 Realizzazione della chiamata di funzione, 94
- 7.4 Programma principale 94
- 7.5 Esempi di funzioni 95
- 7.6 Funzioni con argomenti di default 98
- 7.7 Sovraccarico dei nomi di funzione 99
- 7.8 Funzioni in linea 101
- 7.9 Oggetti come parametri di scambio 102
 - 7.9.1 Generalità, 102 – 7.9.2 Oggetto scambiato per riferimento, 102 – 7.9.3 Oggetto scambiato per valore, 103
- 7.10 Oggetto quale valore restituito 103
 - 7.10.1 La funzione restituisce il valore di un oggetto, 104 – 7.10.2 Realizzazione della restituzione del valore, 104 – 7.10.3 Uso del costruttore di copia, 105 – 7.10.4 La funzione restituisce il riferimento a un oggetto, 106 – 7.10.5 La funzione restituisce il puntatore a un oggetto, 108 – 7.10.6 Restituzione di un riferimento a un'area allocata dalla funzione, 108 – 7.10.7 Riferimenti e puntatori, 109
- 7.11 Riepilogo 110

Capitolo 8 Puntatori e variabili dinamiche 112

- 8.1 Il tipo puntatore 112
- 8.2 Puntatori a dati 114
 - 8.2.1 Definizione del tipo e della variabile, 114 – 8.2.2 Puntatori a variabili semplici, 114 – 8.2.3 Operazioni sui puntatori, 115 – 8.2.4 Array di puntatori, 116 – 8.2.5 Puntatori costanti, 116
- 8.3 Puntatori a dati strutturati 117
 - 8.3.1 Puntatore ad array, 118 – 8.3.2 Puntatori all'interno di un array, 118 – 8.3.3 Puntatori

- relativi all'interno di un array, 119 – 8.3.4 Puntatori assoluti all'interno di un array, 119 – 8.3.5 Puntatore a record, 121 – 8.3.6 Puntatori per il collegamento tra più record, 123
- 8.4 Variabili e strutture dinamiche 124
 - 8.4.1 Generalità, 124 – 8.4.2 Allocazione dinamica di una variabile, 125 – 8.4.3 Allocazione dinamica di un array, 125 – 8.4.4 Allocazione dinamica di un record, 125 – 8.4.5 Collegamento tra più record generati dinamicamente, 127
- 8.5 Riepilogo 128

Capitolo 9 Puntatori e funzioni 130

- 9.1 Premessa 130
- 9.2 Puntatori parametri di scambio 130
- 9.3 Parametro di tipo puntatore 130
- 9.4 Valore restituito di tipo puntatore 135
- 9.5 Argomento e valore restituito di tipo puntatore 135
- 9.6 Puntatori a void 137
- 9.7 Puntatori a funzioni 138
 - 9.7.1 Aspetti generali, 138 – 9.7.2 Meccanismi per i puntatori a funzione, 138
- 9.8 Puntatore a funzione come parametro di scambio 140
- 9.9 Array di puntatori a funzione 143
- 9.10 Riepilogo 144

Capitolo 10 Vettori e matrici 145

- 10.1 Premessa 145
- 10.2 Allocazione degli array 145
 - 10.2.1 Allocazione statica, 145 – 10.2.2 Allocazione dinamica, 146 – 10.2.3 Confronto tra allocazione statica e dinamica, 146
- 10.3 Ordinamento degli elementi dell'array 146
- 10.4 Array quali parametri di scambio 147
 - 10.4.1 Il problema in generale, 147 – 10.4.2 Array monodimensionali, 148 – 10.4.3 Array multidimensionali, 148 – 10.4.4 Array a dimensioni fisse, 149 – 10.4.5 Array a dimensioni adattabili, 149
- 10.5 Controllo degli accessi agli elementi dell'array 150
- 10.6 Esempi sull'uso degli array 151

Capitolo 11 Le stringhe 154

- 11.1 Il tipo stringa del C++ 154
 - 11.1.1 Descrizione, 154 – 11.1.2 Definizione e inizializzazione di una stringa, 155 – 11.1.3 Costanti stringa, 155
- 11.2 Operazioni sulle stringhe 156
 - 11.2.1 Accesso allo *i*-esimo carattere di una stringa, 156 – 11.2.2 Lettura di una stringa, 156 – 11.2.3 Scrittura di una stringa, 156 – 11.2.4 Assegnazione tra stringhe, 157 – 11.2.5 Concatenazione fra stringhe, 157 – 11.2.6 Determinazione della lunghezza della stringa, 158 – 11.2.7 Relazione tra stringhe, 158 – 11.2.8 Ricerca di un carattere

- in una stringa, 158 – 11.2.9 Accesso a una sottostringa, 159 – 11.2.10 Ricerca di una sottostringa in una stringa, 159 – 11.2.11 Trasformazione di una stringa in intero e viceversa, 159 – 11.2.12 Allocazione dinamica di una stringa, 159
- 11.3 Realizzazione della libreria *string* 160
- 11.4 La classe *string* per la manipolazione di stringhe 162
 - 11.4.1 Generalità, 162 – 11.4.2 Definizione e inizializzazione di variabili di tipo *string*, 162 – 11.4.3 Operatori per operazioni su stringhe, 163 – 11.4.4 Altre operazioni realizzate con funzioni membro, 163

Capitolo 12 Struttura dei programmi 164

- 12.1 Ciclo di vita dei programmi 164
- 12.2 Dichiarazioni e definizioni 165
 - 12.2.1 Dichiarazione e definizione di una funzione, 165 – 12.2.2 Dichiarazione e definizione di una variabile, 166
- 12.3 Le unità strutturali di un programma 167
 - 12.3.1 Blocco, 167 – 12.3.2 Funzione, 168 – 12.3.3 File, 169 – 12.3.4 Programma completo, 169
- 12.4 Attributi delle variabili 169
- 12.5 Visibilità 170
- 12.6 Tempo di legame 171
- 12.7 Allocazione della memoria 171
- 12.8 Ciclo di vita delle variabili 172
 - 12.8.1 Variabili statiche, 172 – 12.8.2 Variabili allocate a stack, 172 – 12.8.3 Variabili dinamiche esplicite, 173
- 12.9 Visibilità intermodulo (*linkage*) 173
- 12.10 Variabili globali usate nelle funzioni 174
- 12.11 Inizializzazione delle variabili 175
- 12.12 Struttura del programma 176
 - 12.12.1 Variabile definita al livello di un modulo e importata in un altro, 177 – 12.12.2 Variabile definita al livello di un modulo e non esportabile, 177 – 12.12.3 Variabile definita all'interno della funzione *main()*, 177 – 12.12.4 Variabile definita in una funzione con qualificatore *static*, 178 – 12.12.5 Variabile allocata e deallocata esplicitamente, 178
- 12.13 Precompilazione del testo origine 178
 - 12.13.1 Inserimento di un file, 179 – 12.13.2 I file di intestazione, 179 – 12.13.3 Sostituzione di stringhe e macro, 179 – 12.13.4 Compilazione condizionale, 180 – 12.13.5 Duplicazione di dichiarazioni e compilazione condizionale, 181

Capitolo 13 Costruzione e documentazione dei programmi 182

- 13.1 Meccanismi per la modularizzazione 182
 - 13.1.1 Aspetti concettuali relativi alla modularizzazione, 182 – 13.1.2 Meccanismi offerti dal C++ per la modularizzazione, 183 – 13.1.3 Modalità di documentazione del programma, 184 – 13.1.4 Modalità di documentazione di un singolo modulo, 184
- 13.2 Programma costituito da un unico modulo 185
 - 13.2.1 Struttura, 185 – 13.2.2 Documentazione, 186
- 13.3 Programma costituito da due moduli 187
 - 13.3.1 Caratteristiche generali, 187 – 13.3.2 Struttura del programma, 187 –

- 13.3.3 Compilazione e collegamento, 188 – 13.3.4 Riutilizzabilità del codice del modulo servente, 189 – 13.3.5 Documentazione, 190
- 13.4 Uso di moduli con nomi 192
- 13.5 Programma costituito da più moduli 194
 - 13.5.1 Caratteristiche generali, 194 – 13.5.2 Struttura del programma, 195 – 13.5.3 Documentazione, 197
- 13.6 Uso dei package 197
 - 13.6.1 Package con moduli con nome, 197 – 13.6.2 Package con moduli anonimi, 199
- 13.7 Riepilogo degli stereotipi utilizzati 199

Capitolo 14 Le stringhe di bit 201

- 14.1 Il tipo stringa di bit del C++ 201
 - 14.1.1 Rappresentazione delle stringhe di bit, 201
- 14.2 Operazioni sulle stringhe di bit 202
 - 14.2.1 Operazioni e operandi, 202 – 14.2.2 Costanti di tipo stringa di bit, 202 – 14.2.3 Semantica degli operatori logici e di scorrimento, 202 – 14.2.4 Esempi di operazioni, 203
- 14.3 Stringhe di bit memorizzate in una *struct* 205
- 14.4 La classe *bitset* 206

Parte II Il C++ e la programmazione a oggetti 208

Capitolo 15 Generalità sulle classi 209

- 15.1 Le classi 209
 - 15.1.1 Caratteristiche generali, 209 – 15.1.2 Il sovraccarico dei nomi delle funzioni e degli operatori, 210 – 15.1.3 La genericità, 210 – 15.1.4 L'ereditarietà, 210 – 15.1.5 Il polimorfismo, 211 – 15.1.6 Programmazione a oggetti, 211 – 15.1.7 Le biblioteche e le classi, 211
- 15.2 Produzione e uso della classe 213
 - 15.2.1 Produttore e utente della classe, 213 – 15.2.2 Compilazione e collegamento dei moduli, 213 – 15.2.3 Riutilizzabilità del codice della classe, 214 – 15.2.4 Limiti alla riutilizzabilità in forma chiusa, 214
- 15.3 Specifica e implementazione della classe 215
 - 15.3.1 Struttura della specifica e dell'implementazione, 215 – 15.3.2 Specifica della classe, 216 – 15.3.3 Implementazione della classe, 216 – 15.3.4 L'utente della classe, 216 – 15.3.5 Funzioni membro e oggetto proprio, 217
- 15.4 La specifica come interfaccia 217
 - 15.4.1 Funzionalità offerte dalla specifica, 217 – 15.4.2 Potenzialità e limiti della specifica, 218 – 15.4.3 Variabili membro e attributi della classe, 218
- 15.5 Struttura degli oggetti, costruttore e distruttore 218

Capitolo 16 Le classi. Notazioni di base 220

- 16.1 Dichiarazione della specifica di una classe 220
- 16.2 Tipologia delle funzioni membro 222

- 16.3 Costruttori 222
 - 16.3.1 Costruttore con zero argomenti, 223 – 16.3.2 Costruttore di copia, 223
- 16.4 Ciclo di vita degli oggetti e funzione distruttore 224
- 16.5 Esempi di funzioni costruttore e distruttore 225
 - 16.5.1 Costruttori per la classe *Complex*, 225 – 16.5.2 Costruttori e distruttore per la classe *String*, 227
- 16.6 Funzioni di accesso e di posizionamento 229
- 16.7 Funzioni ordinarie operanti sugli oggetti 229
- 16.8 Accesso ai membri di una classe 230
 - 16.8.1 Accesso da parte del programma utente, 230 – 16.8.2 Accesso da parte di una funzione membro (puntatore *this*), 230
- 16.9 Funzioni in linea 231
- 16.10 Funzioni amiche 232
 - 16.10.1 Funzione amica di una classe, 232 – 16.10.2 Funzione operante su oggetti appartenenti a classi distinte, 233 – 16.10.3 Funzione amica di più classi, 233 – 16.10.4 Relazione di amicizia tra classi, 234

Capitolo 17 Realizzazione di un tipo di dato astratto 235

- 17.1 Premessa 235
- 17.2 Progetto della classe *Pila* 235
- 17.3 La specifica della classe *Pila* 236
- 17.4 Realizzazione della classe *Pila* 238
- 17.5 Uso della classe *Pila* 240
- 17.6 Altre versioni della libreria e della classe *pila* 240
- 17.7 Confronto tra libreria di funzioni e classi 241
 - 17.7.1 Confronto tra le specifiche, 241 – 17.7.2 Confronto tra le realizzazioni, 242 – 17.7.3 Confronto tra i programmi utente, 243 – 17.7.4 Versione della libreria con una *struct*, 243 – 17.7.5 Versione della libreria con dati dinamici, 245

Capitolo 18 Dettagli sintattici e implementativi sulle classi 247

- 18.1 Operazioni predefinite e definite dal progettista 247
- 18.2 Oggetti o componenti costanti 249
 - 18.2.1 Oggetti costanti, 249 – 18.2.2 Oggetti con componenti costanti, 249
- 18.3 Oggetti costanti e componenti modificabili 250
 - 18.3.1 Qualificatore *mutable*, 250 – 18.3.2 Operatore di conversione *const_cast*, 250
- 18.4 Variabili, funzioni membro e costanti statiche 251
 - 18.4.1 Variabili membro statiche, 251 – 18.4.2 Funzioni membro statiche, 251 – 18.4.3 Costanti statiche, 253
- 18.5 Relazioni tra classi, strutture e unioni 254
 - 18.5.1 Classi e strutture, 254 – 18.5.2 Classi e unioni, 255
- 18.6 Contenimento di classi (aggregazione) 255
- 18.7 Domini di visibilità delle classi 257
 - 18.7.1 Dichiarazioni ausiliarie per le classi, 257 – 18.7.2 Classi annidate, 257
- 18.8 Funzioni membro ricorsive 259
- 18.9 Array di oggetti 259
- 18.10 Oggetti con estensione allocati in area heap 260

- 18.11 Puntatori a membri *.** e *->** 261
 - 18.11.1 Puntatori a variabili membro, 261 – 18.11.2 Puntatori a funzioni membro, 262

Capitolo 19 Ridefinizione degli operatori 264

- 19.1 Premessa 264
- 19.2 Funzioni operatore 264
- 19.3 Regole per la ridefinizione degli operatori 267
- 19.4 Ridefinizione di operatori specifici 270
 - 19.4.1 Operatore di assegnazione, 270 – 19.4.2 Operatori di incremento e decremento, 272 – 19.4.3 Operatori di indicizzazione, 272 – 19.4.4 Operatore *&*, 273 – 19.4.5 Operatore virgola, 273 – 19.4.6 Operatore di chiamata di funzione, 274 – 19.4.7 Operatore *!*, 274 – 19.4.8 Operatori di shift *<<* e *>>*, 276 – 19.4.9 Operatore *->*, 276 – 19.4.10 Operatore *[]* con dimensioni multiple, 277 – 19.4.11 Operatori *new* e *delete*, 278 – 19.4.12 Operatore di conversione, 279

Capitolo 20 Ereditarietà: aspetti generali 280

- 20.1 Motivazioni per l'ereditarietà 280
 - 20.1.1 L'ereditarietà nella progettazione del software, 280 – 20.1.2 L'ereditarietà quale strumento per il riuso del software, 281 – 20.1.3 L'ereditarietà nei linguaggi di programmazione, 281
- 20.2 Le classi derivate nel C++: aspetti generali 282
 - 20.2.1 Meccanismo base di derivazione, 282 – 20.2.2 Meccanismo di accesso, 282 – 20.2.3 Modalità di derivazione, 282 – 20.2.4 Polimorfismo, 282 – 20.2.5 Derivazione multipla, 283
- 20.3 I meccanismi sintattici per la derivazione 283
 - 20.3.1 Struttura della classe derivata, 283
- 20.4 Meccanismi e diritti di accesso 286
 - 20.4.1 Aspetti generali, 286 – 20.4.2 Accesso delle funzioni membro e funzioni utente, 286 – 20.4.3 Uso della gerarchia di derivazione, 289 – 20.4.4 Overriding e overloading, 290 – 20.4.5 Modalità di derivazione privata, 291
- 20.5 Ampliamento dei meccanismi di protezione e di derivazione 292
- 20.6 Meccanismi generali di protezione e di derivazione 294
 - 20.6.1 Trasmissione dei diritti di accesso, 294 – 20.6.2 Meccanismi selettivi di derivazione, 296 – 20.6.3 Funzioni o classi amiche nella derivazione, 297
- 20.7 Un esempio di gerarchia di derivazione 297
 - 20.7.1 Esempio base, 297 – 20.7.2 Una variante all'esempio base, 301

Capitolo 21 Gerarchie di classi 303

- 21.1 Struttura della gerarchia di classi 303
 - 21.1.1 Derivazione pubblica e privata, la classe *Lista*, 304 – 21.1.2 Riuso con riduzione delle funzionalità, 308
- 21.2 Costruttori e distruttori nella gerarchia di derivazione 309
 - 21.2.1 Costruttori, 309 – 21.2.2 Distruttore, 311 – 21.2.3 Costruttore di copia nella classe derivata, 311 – 21.2.4 Operatore di assegnazione, 312

- 21.3 Compatibilità tra classe base e classi derivate 313
- 21.4 Alterazione dei meccanismi di accesso 316

Capitolo 22 Polimorfismo 318

- 22.1 Premessa 318
- 22.2 Motivazioni per il polimorfismo 318
- 22.3 Meccanismi sintattici per il polimorfismo 319
- 22.4 Un esempio di gerarchia di derivazione con funzioni virtuali 322
- 22.5 Funzioni virtuali pure e classi astratte 323
- 22.6 Costruttori e distruttori in classi polimorfe 323
 - 22.6.1 Chiamata di costruttori e distruttori, 323 – 22.6.2 Polimorfismo all'interno di costruttori e distruttori, 324
- 22.7 Un esempio di progetto: la classe *Figura* 325
 - 22.7.1 Aspetti generali del progetto, 325 – 22.7.2 Notazioni disponibili all'utente della classe *Figura*, 325 – 22.7.3 Dichiarazione e realizzazione della gerarchia di classi *Figura*, 328
- 22.8 Compilazione e collegamento delle classi derivate 332
- 22.9 Struttura della gerarchia di classi 333
 - 22.9.1 Funzioni associate alle classi, 333 – 22.9.2 Accesso alle funzioni membro della gerarchia, 334 – 22.9.3 Un esempio di gerarchia di derivazione, 334 – 22.9.4 Tabelle dei metodi, 335 – 22.9.5 Attivazione delle funzioni membro, 337

Capitolo 23 Derivazione multipla 339

- 23.1 Premessa 339
- 23.2 Derivazione multipla da classi indipendenti 339
- 23.3 Derivazione multipla da classi collegate 343
- 23.4 Derivazione virtuale 346
 - 23.4.1 Aspetti sintattici, 346 – 23.4.2 Realizzazione della derivazione virtuale, 347
- 23.5 Costruttori e distruttori nella derivazione multipla 348
- 23.6 Un esempio di gerarchia di derivazione 351

Capitolo 24 Funzioni e classi generiche 355

- 24.1 Motivazioni per la genericità 355
- 24.2 Funzioni modello 356
 - 24.2.1 Definizione delle funzioni modello, 356 – 24.2.2 Istanziamento di una funzione modello, 356 – 24.2.3 Realizzazione del meccanismo delle funzioni modello, 357 – 24.2.4 Funzioni modello con più parametri tipo, 358 – 24.2.5 Differenza tra funzioni sovraccaricate e funzioni modello, 359 – 24.2.6 Un approfondimento sulla struttura delle funzioni modello, 360
- 24.3 Classi modello 360
 - 24.3.1 Specifica della classe modello, 360 – 24.3.2 Implementazione della classe modello, 361 – 24.3.3 Specializzazione della classe modello, 361 – 24.3.4 Realizzazione del meccanismo delle classi modello, 362 – 24.3.5 Modalità di preparazione della classe modello, 362 – 24.3.6 Classi modello con parametri-tipo e parametri-variabile, 365 – 24.3.7 Funzioni modello

- e classi modello, 367 – 24.3.8 Funzioni membro con parametri-tipo, 368 – 24.3.9 Classi modello con parametri di default, 369
- 24.4 Relazioni di amicizia tra modelli 370
- 24.5 Esempi di classi modello 371

Capitolo 25 Meccanismi di incapsulamento: namespace 373

- 25.1 Classi di identificatori 373
- 25.2 Controllo dello spazio dei nomi: namespace 374
 - 25.2.1 Inquinamento dello spazio dei nomi, 374 – 25.2.2 La dichiarazione namespace, 375 – 25.2.3 Accesso allo spazio dei nomi, 376 – 25.2.4 Esempi di impiego del namespace, 377
- 25.3 Il namespace come interfaccia 378
 - 25.3.1 Composizione e selezione dello spazio dei nomi, 378 – 25.3.2 Dichiarazione di un'interfaccia per una libreria, 378 – 25.3.3 Composizione dello spazio dei nomi, 379 – 25.3.4 Adeguamento della libreria standard al namespace, 381
- 25.4 Spazio dei nomi anonimo, namespace sinonimi 382
 - 25.4.1 Spazio dei nomi anonimo, 382 – 25.4.2 Namespace sinonimi, 383

Capitolo 26 Conversioni di tipo 384

- 26.1 Conversioni di tipo nel C++ 384
- 26.2 Operatori per la conversione di tipo 385
 - 26.2.1 Aspetti generali, 385 – 26.2.2 Operatore di conversione *static_cast*, 386 – 26.2.3 Operatore *reinterpret_cast*, 387 – 26.2.4 Operatore *dynamic_cast*, 387 – 26.2.5 Operatore *typeid* per l'accesso al tipo di un oggetto, 388 – 26.2.6 Operatore *const_cast*, 389
- 26.3 Espressioni aritmetiche miste e assegnazione 389
 - 26.3.1 Aspetti generali, 389 – 26.3.2 Promozione delle variabili, 390 – 26.3.3 Operazione di assegnazione, 391
- 26.4 Conversioni di tipo per oggetti di tipo classe 391
 - 26.4.1 Aspetti generali, 391 – 26.4.2 Conversione mediante costruttori, 393 – 26.4.3 Operatori di conversione di tipo, 394 – 26.4.4 Riepilogo dei meccanismi di conversione, 395
- 26.5 Risoluzione delle chiamate di funzioni sovrapposte 397
- 26.6 Risoluzione delle chiamate di operatori sovrapposti 398
- 26.7 Tipo dei letterali 399

Parte III La libreria standard del C++ 400

Capitolo 27 La libreria *iostream* per le operazioni di I/O 401

- 27.1 Premessa 401
- 27.2 La libreria standard del C++ 401
- 27.3 La libreria di classi per l'I/O 401
 - 27.3.1 Aspetti generali, 401 – 27.3.2 La gerarchia di classi di I/O, 402 – 27.3.3 Stream di testo e binari, 403
- 27.4 Operazioni di ingresso e uscita primarie 403

- 27.4.1 I/O primario in generale, 403 – 27.4.2 Le classi *istream* e *ostream*, 404 – 27.4.3 Gli stream standard *cin* e *cout*, 404
- 27.5 Operazione di lettura da file standard di ingresso 405
- 27.6 Operazione di scrittura su file standard di uscita 407
- 27.7 I/O per tipi definiti dall'utente 408
- 27.8 Operazioni di ingresso e uscita con formato 409
 - 27.8.1 Funzioni per il controllo del formato, 409 – 27.8.2 Indicatori di formato, 410
- 27.9 Manipolatori senza argomento 411
 - 27.9.1 Uso dei manipolatori senza argomento, 411 – 27.9.2 Realizzazione dei manipolatori senza argomento, 412
- 27.10 Manipolatori con un argomento 413
 - 27.10.1 Uso dei manipolatori con un argomento, 413 – 27.10.2 Realizzazione dei manipolatori con un argomento, 413
- 27.11 Gestione dello stato dello stream 415

Capitolo 28 Operazioni di I/O verso le memorie di massa 417

- 28.1 Collegamenti con il sistema operativo. Apertura e chiusura 417
 - 28.1.1 Generalità, 417 – 28.1.2 Apertura e chiusura dei file, 418 – 28.1.3 Specifica delle operazioni da compiere, 418
- 28.2 Input e output 421
 - 28.2.1 Operazioni per file ad accesso diretto, 422 – 28.2.3 Controllo di errore, 423
- 28.3 Generazione e ispezione di un file sequenziale 424
 - 28.3.1 Processo di generazione di un file sequenziale, 424 – 28.3.2 Processo di ispezione di un file sequenziale, 425 – 28.3.3 Copia di file, 427
- 28.4 Generazione e uso di file ad accesso diretto 429
 - 28.4.1 Generalità, 429 – 28.4.2 Processo di generazione casuale di un file, 429 – 28.4.3 Processo di ispezione casuale di un file, 431 – 28.4.4 Processo di aggiornamento casuale di un file, 432
- 28.5 Le stringhe per le operazioni di ingresso/uscita 433
- 28.6 Interfacciamento verso il sistema operativo 435
 - 28.6.1 Operazioni eseguite dalla classe *streambuf* 435
- 28.7 Collegamento tra stream 436
- 28.8 La libreria standard *stdio* del linguaggio C 436
 - 28.8.1 Le librerie *stdio* e *iostream*, 436 – 28.8.2 La libreria *stdio*: generalità, 437 – 28.8.3 Operazioni su file, 437

Capitolo 29 STL: la libreria standard con classi modello 443

- 29.1 Aspetti generali 443
- 29.2 Contenitori 444
 - 29.2.1 Contenitori di sequenza di prima classe, 444 – 29.2.2 Contenitori con adattatori, 445 – 29.2.3 Quasi contenitori, 445 – 29.2.4 Aspetti implementativi, 447 – 29.2.5 Operazioni applicabili ai contenitori, 448
- 29.3 Iteratori 448
- 29.4 Algoritmi generici operanti sui contenitori 452
- 29.5 Funzioni oggetto 452
 - 29.5.1 Scopo e struttura delle funzioni oggetto, 452 – 29.5.2 Esempi relativi alle funzioni oggetto, 453

Capitolo 30 Le classi contenitore 456

- 30.1 La classe *vector* 456
 - 30.1.1 Principali funzioni della classe *vector*, 457 – 30.1.2 Esempi di operazioni sulla classe *vector*, 459
- 30.2 La classe *deque* 459
 - 30.2.1 Principali funzioni della classe *deque*, 460 – 30.2.2 Esempi di operazioni sulla classe *deque*, 461
- 30.3 La classe *list* 461
 - 30.3.1 Principali funzioni della classe *list*, 462 – 30.3.2 Esempi di operazioni sulla classe *list*, 464
- 30.4 La classe *set* 464
 - 30.4.1 Principali funzioni della classe *set*, 465 – 30.4.2 Esempi di operazioni sulla classe *set*, 466
- 30.5 La classe *multiset* 467
 - 30.5.1 Principali funzioni della classe *multiset*, 467 – 30.5.2 Esempi di operazioni sulla classe *multiset*, 469
- 30.6 La classe *map* 469
 - 30.6.1 Principali funzioni della classe *map*, 470 – 30.6.2 Esempi di operazioni sulla classe *map*, 471
- 30.7 La classe *multimap* 472
 - 30.7.1 Principali funzioni della classe *multimap*, 472 – 30.7.2 Esempi di operazioni sulla classe *multimap*, 474
- 30.8 La classe *stack* adattatore delle classi contenitore 474
 - 30.8.1 Principali funzioni della classe *stack*, 475 – 30.8.2 Esempi di operazioni sulla classe *stack*, 475
- 30.9 La classe *queue* adattatore delle classi contenitore 476
 - 30.9.1 Principali funzioni della classe *queue*, 476 – 30.9.2 Esempi di operazioni sulla classe *queue*, 477
- 30.10 La classe *priority_queue* adattatore delle classi contenitore 477
 - 30.10.1 Principali funzioni della classe *priority_queue*, 478 – 30.10.2 Esempi di operazioni sulla classe *priority_queue*, 478

Capitolo 31 Gli algoritmi generici 479

- 31.1 Classificazione degli algoritmi generici 479
- 31.2 Algoritmi che non modificano la sequenza degli elementi 479
- 31.3 Algoritmi che modificano la sequenza degli elementi 481
- 31.4 Algoritmi di ordinamento 485
- 31.5 Algoritmi numerici generalizzati 491

Parte IV La progettazione a oggetti 494

Capitolo 32 Il linguaggio di modellazione a oggetti UML 495

- 32.1 Premessa 495
- 32.2 Sviluppo orientato agli oggetti 495
 - 32.2.1 Modelli orientati agli oggetti, 495 – 32.2.3 Analisi, progetto e programmazione a oggetti, 496

- 32.3 OOA: realizzazione di un modello statico 497
 - 32.3.1 Identificazione degli oggetti, 497 – 32.3.2 Classificazione, 498
- 32.4 Il linguaggio UML 498
 - 32.4.1 Cenni storici, 498 – 32.4.2 Scopi di UML, 498
- 32.5 Diagramma dei casi d'uso 499
 - 32.5.1 Diagramma dei casi d'uso: semantica, 499 – 32.5.2 Diagramma dei casi d'uso: sintassi, 500 – 32.5.3 Casi d'uso e cicli di vita, 500 – 32.5.4 Scenari, 501
- 32.6 Diagramma delle classi 502
 - 32.6.1 Diagramma delle classi: semantica, 502 – 32.6.2 Diagramma delle classi: sintassi, 502 – 32.6.3 Attributi e metodi, 504 – 32.6.4 Visibilità di attributi e metodi, 504 – 32.6.5 Il package, 505
- 32.7 Relazioni tra Classi&Oggetti 506
 - 32.7.1 Le relazioni in generale, 506 – 32.7.2 Generalizzazione-specializzazione, 506 – 32.7.3 Aggregazione, 507 – 32.7.4 Associazione, 508 – 32.7.5 Asserzioni relative ai metodi delle classi, 512

Capitolo 33 UML: aspetti dinamici del modello 513

- 33.1 Premessa 513
- 33.2 Diagrammi di interazione 513
 - 33.2.1 Dinamiche inter-oggetto: i diagrammi di interazione, 513 – 33.2.2 I diagrammi di sequenza, 514 – 33.2.3 I diagrammi di collaborazione, 515
- 33.3 Diagrammi di attività 516
 - 33.3.1 Diagramma di attività: semantica, 516 – 33.3.2 Diagramma di attività: sintassi, 517
- 33.4 Diagrammi di stato 519
 - 33.4.1 Diagramma di stato: semantica, 519 – 33.4.2 Diagramma di stato: sintassi, 519
- 33.5 Diagrammi di implementazione 520
 - 33.5.1 Diagramma dei componenti, 521 – 33.5.2 Diagramma di allocazione, 522

Capitolo 34 Da UML a C++ 524

- 34.1 Premessa 524
- 34.2 Modelli a oggetti statici e dinamici 524
 - 34.2.1 Modello statico: relazioni tra classi, 524 – 34.2.2 Modello dinamico: diagrammi di interazione, 525
- 34.3 Gerarchia di generalizzazione-specializzazione 525
 - 34.3.1 Organizzazione della gerarchia, 525 – 34.3.2 Il ruolo del polimorfismo nella progettazione, 526
- 34.4 Contenimento tra classi (aggregazione) 527
 - 34.4.1 Realizzazione del contenimento lasco, 527 – 34.4.2 Realizzazione del contenimento stretto, 528 – 34.4.3 Un esempio di aggregazione tra classi, 529
- 34.5 Associazione tra classi 531
 - 34.5.1 Associazione uno a uno, 531 – 34.5.2 Associazione uno a molti, 533 – 34.5.3 Un esempio di associazione tra classi, 533
- 34.6 Interazioni dinamiche tra oggetti 535

Capitolo 35 Sviluppo di un progetto completo 537

- 35.1 Premessa 537
- 35.2 Analisi orientata agli oggetti (OOA) 537
 - 35.2.1 Requisiti informali di utente, 537 – 35.2.2 Analisi dei requisiti di utente: diagramma dei casi d'uso, 537
- 35.3 Gli oggetti di dominio 541
- 35.4 Problematiche di interfacciamento 544
- 35.5 Progetto orientato agli oggetti (OOD) 546
- 35.6 OOP, specifica di basso livello e implementazione 550
 - 35.6.1 Specifica Dipendente, 550 – 35.6.2 Specifica Agente, 551 – 35.6.3 Specifica Dirigente, 552 – 35.6.4 Specifica Ordinario, 552 – 35.6.5 Specifica Missione, 553 – 35.6.6 Specifica Contratto, 553 – 35.6.7 Implementazione di Dipendente, 553 – 35.6.8 Implementazione di Dirigente, 554 – 35.6.9 Implementazione di Agente, 555 – 35.6.10 Implementazione di Missione, 555 – 35.6.11 Implementazione di Contratto, 556 – 35.6.12 Esempio Gestione Dipendenti, 556

Parte V Libreria di programmi in C++ 559

Capitolo 36 Semplici programmi completi 560

- 36.1 Calcolo della somma 560
- 36.2 Calcolo del tasso di inflazione 561
- 36.3 Minimo fra due interi 562
- 36.4 Massimo fra tre interi 562
- 36.5 Calcolo di x^n 563
- 36.6 Calcolo della serie armonica 564
- 36.7 Stampa tabella triangolare 565
- 36.8 Valore medio di una lista (numero di elementi dato) 566
- 36.9 Valore medio di una lista (uso di un valore tappo) 566
- 36.10 Massimo e minimo in una sequenza 567

Capitolo 37 Algoritmi numerici 569

- 37.1 Calcolo di $n!$ 569
- 37.2 Calcolo della radice quadrata di x 570
- 37.3 Calcolo dei quadrati dei numeri interi 571
- 37.4 Calcolo degli zeri di $f(x)$: metodo di bisezione 571
- 37.5 Calcolo degli zeri di $f(x)$: metodo dello scandaglio 573
- 37.6 Calcolo degli zeri di $f(x)$: metodo di Newton-Raphson 575
- 37.7 Calcolo dell'integrale definito con metodo di Eulero e precisione assegnata 577
- 37.8 Calcolo dell'integrale definito con metodo di Simpson e precisione assegnata 579
- 37.9 Risoluzione di un sistema di equazioni lineari (metodo di Gauss-Jordan) 581
- 37.10 Risoluzione di un sistema di equazioni lineari (metodo di Gauss-Seidel) 584
- 37.11 Risoluzione di un sistema equazioni differenziali (metodi di Eulero e Runge-Kutta) 588
- 37.12 Libreria associata al programma SEDRKE 590

Capitolo 38 Schemi di programmi e programmi completi 593

- 38.1 Schema menu a un livello 593
- 38.2 Schema menu a due livelli 596
- 38.3 Libreria per lettura e stampa di una matrice 599
- 38.4 Simulazione di una lista di attesa 601
- 38.5 Analisi rete elettrica in corrente continua 605
- 38.6 Calcolo del valore di un polinomio 608

Capitolo 39 Algoritmi di ricerca e ordinamento 610

- 39.1 Libreria ricerca elemento in un array 610
- 39.2 Libreria per l'ordinamento di una lista 611
- 39.3 Fusione di due vettori 616
- 39.4 Fusione di due file 617
- 39.5 Distribuzione di un file 618
- 39.6 Ordinamento per distribuzione e fusione 619

Capitolo 40 Strutture dati 621

- 40.1 Libreria per il trattamento della pila 621
 - 40.1.1 *PILASL1*: libreria per la gestione di una pila realizzata con un array, 622 –
 - 40.1.2 *PILASL2*: libreria per la gestione di una pila con array e *struct*, 624 – 40.1.3 *PILADL*: gestione pila con lista dinamica, 626 – 40.1.4 Programma *USALIBRERIAPILA*, 628
- 40.2 Libreria per il trattamento della coda 630
 - 40.2.1 *CODASL*: libreria gestione coda con array, 630 – 40.2.2 *CODADL*: libreria per la gestione di una coda realizzata con una struttura dati dinamica, 633 – 40.2.3 Programma *USALIBRERIA CODA*, 635
- 40.3 Libreria per il trattamento della lista 636
 - 40.3.1 *LISTASL*: libreria per la gestione di una lista concatenata realizzata con un array allocato staticamente, 637 – 40.3.2 *LISTADL*: libreria per la gestione di una lista concatenata realizzata con una struttura dati dinamica, 641 – 40.3.3 Programma *USALIBRERIALISTA*, 644
- 40.4 Libreria per il trattamento dell'albero 646
 - 40.4.1 *AlberoL*: libreria per la gestione di un albero, 647 – 40.4.2 Programma *USAALBERO*, 651
- 40.5 Libreria per il trattamento di una tabella 653
 - 40.5.1 *TABSL*: tabella realizzata con un array, 653 – 40.5.2 *TABDL*: libreria per manipolare tabelle con liste di tipo dinamico, 656 – 40.5.3 Programma uso libreria tabella, 659

Parte VI Libreria di classi in C++ 662**Capitolo 41 Strutture dati realizzate con classi 663**

- 41.1 Classe *Pila* per il trattamento della pila 663
 - 41.1.1 Classe *Pila* con array statico, 663 – 41.1.2 Classe *Pila* con lista dinamica, 665 –
 - 41.1.3 Programma per l'uso della classe *Pila*, 666 – 41.1.4 Classe modello *Pila* con array, 667 –

- 41.1.5 Programma per l'uso della classe modello *Pila*, 668 – 41.1.6 Classe modello *Pila* con struttura dinamica, 669
- 41.2 Classe *Coda* per il trattamento della coda 670
 - 41.2.1 Classe *Coda* con struttura dati di tipo statico, 671 – 41.2.2 Classe *Coda* con struttura dati dinamica, 673 – 41.2.3 Programma per l'uso della classe *Coda*, 675
- 41.3 Classe *Lista* ordinata 676
 - 41.3.1 Classe *Lista* ordinata con array di tipo statico, 677 – 41.3.2 Classe *Lista* dinamica ordinata, 680 – 41.3.3 Programma per l'uso della classe *Lista*, 683
- 41.4 Classe *Albero* binario ordinato 684
 - 41.4.1 Classe *Albero* con struttura dinamica, 684 – 41.4.2 Programma per l'uso della classe *Albero*, 688
- 41.5 Classe *Tabella* per il trattamento della tabella 689
 - 41.5.1 Classe *Tabella* con implementazione di tipo statico, 689 – 41.5.2 Classe *Tabella* con implementazione di tipo dinamico, 692 – 41.5.3 Programma per l'uso della classe *Tabella*, 695

Capitolo 42 Semplici classi concrete 697

- 42.1 Classe *Contatore* modulo *N* 697
- 42.2 Classe *Punto* 698
- 42.3 Classe *Data* per il trattamento delle date 699
- 42.4 Classe *Complex* per il trattamento di numeri complessi 703
 - 42.4.1 Dichiarazione della classe *Complex* (versione 1 - ridotta), 704 – 42.4.2 Classe *Complex* (versione completa), 706 – 42.4.3 Classe *Complex* (con template), 710
- 42.5 Classe *Razionale* per il trattamento di numeri razionali 712
- 42.6 Classe *Stringa* per il trattamento di stringhe di caratteri 715
- 42.7 Classe *BigInt* per il trattamento di interi lunghi 716

Capitolo 43 Classi con vettori 721

- 43.1 Classe *Vettore* per array monodimensionali 721
 - 43.1.1 Classe modello *Vettore*, 724
- 43.2 Classe *UsaOggetti* 725
- 43.3 Classe *Matrice* per array bidimensionali 729
 - 43.3.1 Dichiarazione della classe *Matrice*, 730

Capitolo 44 Gestione di un archivio ad accesso casuale con indice 738

- 44.1 Premessa 738
- 44.2 Analisi orientata agli oggetti (OOA) 738
- 44.3 Gli oggetti di dominio 739
- 44.4 Progetto orientato agli oggetti (OOD) 739
- 44.5 OOP specifica di basso livello e implementazione 739
- 44.6 Testo del programma Gestione Archivio con Tabella di Accesso 743

Capitolo 45 Progetto di uno schedatore 761

- 45.1 Premessa 761
- 45.2 Analisi orientata agli oggetti (OOA) 761
 - 45.2.1 Requisiti informali di utente 761 – 45.2.2 Analisi dei requisiti di utente: diagramma dei casi d'uso 761
- 45.3 Gli oggetti di dominio 763
- 45.4 Progetto orientato agli oggetti (OOD) 764
- 45.5 OOP specifica di basso livello e implementazione 765
- 45.6 Testo del programma schedatore 767

Appendice A Sintesi delle notazioni del C++ 775

- A.1 Classe *Vettore* per array monodimensionali 775
 - A.1.1 Dichiarazioni, definizioni e prototipi di funzioni, 775 – A.1.2 Unità di traduzione, 776
- A.2 Elementi lessicografici 777
- A.3 Struttura del programma 778
- A.4 Tipi di dato e relativi operatori 779
- A.5 Definizioni di variabili con relativo tipo 781
- A.6 Operatori ed espressioni 781
- A.7 Istruzioni 782
- A.8 Classi 784
- A.9 Classi derivate 786
- A.10 Tabelle (parole chiave, delimitatori, operatori) 788

Bibliografia 795