Conteúdo

Prefácio **xi**

0.1 Exercícios sobre classes . . . xiii

1 Introdução à Programação **1**

1.1 Computadores . . . 1

1.2 Programação . . . 2

1.3 Algoritmos: resolvendo problemas . . . 3

1.3.1 Regras do jogo . . . 4

1.3.2 Desenvolvimento e demonstração de correcção . . . 8

1.4 Programas . . . 16

1.5 Resumo: resolução de problemas . . . 18

2 Conceitos básicos de programação **19**

2.1 Introdução . . . 19

2.1.1 Consola e canais . . . 23

2.1.2 Definição de variáveis . . . 25

2.1.3 Controlo de fluxo . . . 26

2.2 Variáveis . . . 28

2.2.1 Memória e inicialização . . . 28

2.2.2 Identificadores . . . 30

2.2.3 Inicialização . . . 31

2.3 Tipos básicos . . . 31

2.3.1 Tipos aritméticos . . . 33

2.3.2 Booleanos ou lógicos . . . 39

2.3.3 Caracteres . . . 39

2.4 Valores literais 42
2.5 Constantes 43
2.6 Instâncias 46
2.7 Expressões e operadores 46
2.7.1 Operadores aritméticos 47
2.7.2 Operadores relacionais e de igualdade 49
2.7.3 Operadores lógicos 50
2.7.4 Operadores *bit*-a-*bit* 51
2.7.5 Operadores de atribuição 53
2.7.6 Operadores de incrementação e decrementação 54
2.7.7 Precedência e associatividade 55
2.7.8 Efeitos laterais e mau comportamento 55
2.7.9 Ordem de cálculo 57

3 Modularização: rotinas **59**
3.1 Introdução à modularização 59
3.2 Funções e procedimentos: rotinas 62
3.2.1 Abordagens descendente e ascendente 63
3.2.2 Definição de rotinas 70
3.2.3 Sintaxe das definições de funções 73
3.2.4 Contrato e documentação de uma rotina 74
3.2.5 Integração da função no programa 76
3.2.6 Sintaxe e semântica da invocação ou chamada 77
3.2.7 Parâmetros 78
3.2.8 Argumentos 79
3.2.9 Retorno e devolução 79
3.2.10 Significado de `void` 79
3.2.11 Passagem de argumentos por valor e por referência 81
3.2.12 Variáveis locais e globais 89
3.2.13 Blocos de instruções ou instruções compostas 89
3.2.14 Âmbito ou visibilidade de variáveis 90
3.2.15 Duração ou permanência de variáveis 93
3.2.16 Nomes de rotinas 94

3.2.17 Declaração vs. definição 95
3.2.18 Parâmetros constantes 99
3.2.19 Instruções de asserção 102
3.2.20 Melhorando módulos já produzidos 112
3.3 Rotinas recursivas 115
3.4 Mecanismo de invocação de rotinas 117
3.5 Sobrecarga de nomes 124
3.6 Parâmetros com argumentos por omissão 126

4 Controlo do fluxo dos programas 129
4.1 Instruções de selecção 129
4.1.1 As instruções `if` e `if else` 130
4.1.2 Instruções de selecção encadeadas 133
4.1.3 Problemas comuns 136
4.2 Asserções 137
4.2.1 Dedução de asserções 137
4.2.2 Predicados mais fortes e mais fracos 139
4.2.3 Dedução da pré-condição mais fraca de uma atribuição 139
4.2.4 Asserções em instruções de selecção 141
4.3 Desenvolvimento de instruções de selecção 146
4.3.1 Escolha das instruções alternativas 147
4.3.2 Determinação das pré-condições mais fracas 148
4.3.3 Determinação das guardas 149
4.3.4 Verificação das guardas 149
4.3.5 Escolha das condições 150
4.3.6 Alterando a solução 151
4.3.7 Metodologia 152
4.3.8 Discussão 153
4.3.9 Outro exemplo de desenvolvimento 156
4.4 Variantes das instruções de selecção 159
4.4.1 O operador `? :` 159
4.4.2 A instrução `switch` 160
4.5 Instruções de iteração 164

4.5.1 A instrução de iteração `while` . . . 165
4.5.2 Variantes do ciclo `while`: `for` e `do while` . . . 166
4.5.3 Exemplo simples . . . 175
4.5.4 `return`, `break`, `continue`, e `goto` em ciclos . . . 175
4.5.5 Problemas comuns . . . 182
4.6 Asserções com quantificadores . . . 183
4.6.1 Somas . . . 184
4.6.2 Produtos . . . 185
4.6.3 Conjunções e o quantificador universal . . . 186
4.6.4 Disjunções e o quantificador existencial . . . 187
4.6.5 Contagens . . . 188
4.6.6 O resto da divisão . . . 189
4.7 Desenvolvimento de ciclos . . . 189
4.7.1 Noção de invariante . . . 193
4.7.2 Correcção de ciclos . . . 199
4.7.3 Melhorando a função `potência()` . . . 202
4.7.4 Metodologia de Dijkstra . . . 204
4.7.5 Um exemplo . . . 217
4.7.6 Outro exemplo . . . 229

5 Matrizes, vectores e outros agregados 235
5.1 Matrizes clássicas do C++ . . . 236
5.1.1 Definição de matrizes . . . 237
5.1.2 Indexação de matrizes . . . 238
5.1.3 Inicialização de matrizes . . . 240
5.1.4 Matrizes multidimensionais . . . 241
5.1.5 Matrizes constantes . . . 242
5.1.6 Matrizes como parâmetros de rotinas . . . 242
5.1.7 Restrições na utilização de matrizes . . . 247
5.2 Vectores . . . 248
5.2.1 Definição de vectores . . . 249
5.2.2 Indexação de vectores . . . 249
5.2.3 Inicialização de vectores . . . 250

5.2.4 Operações . . . 251
5.2.5 Acesso aos itens de vectores . . . 251
5.2.6 Alteração da dimensão de um vector . . . 252
5.2.7 Inserção e remoção de itens . . . 253
5.2.8 Vectores multidimensionais? . . . 254
5.2.9 Vectores constantes . . . 255
5.2.10 Vectores como parâmetros de rotinas . . . 255
5.2.11 Passagem de argumentos por referência constante . . . 257
5.2.12 Outras operações com vectores . . . 261
5.3 Algoritmos com matrizes e vectores . . . 262
5.3.1 Soma dos elementos de uma matriz . . . 262
5.3.2 Soma dos itens de um vector . . . 266
5.3.3 Índice do maior elemento de uma matriz . . . 266
5.3.4 Índice do maior item de um vector . . . 272
5.3.5 Elementos de uma matriz num intervalo . . . 272
5.3.6 Itens de um vector num intervalo . . . 277
5.3.7 Segundo elemento de uma matriz com um dado valor . . . 278
5.3.8 Segundo item de um vector com um dado valor . . . 286
5.4 Cadeias de caracteres . . . 288
5.4.1 Cadeias de caracteres clássicas . . . 289
5.4.2 A classe `string` . . . 291

6 Tipos enumerados 297
6.1 Sobrecarga de operadores . . . 299

7 Tipos abstractos de dados e classes C++ 301
7.1 De novo a soma de fracções . . . 302
7.2 Tipos Abstractos de Dados e classes C++ . . . 307
7.2.1 Definição de TAD . . . 308
7.2.2 Acesso aos membros . . . 311
7.2.3 Alguma nomenclatura . . . 311
7.2.4 Operações suportadas pelas classes C++ . . . 312
7.3 Representação de racionais por fracções . . . 312

7.3.1 Operações aritméticas elementares . . . 313
7.3.2 Canonicidade do resultado . . . 313
7.3.3 Aplicação à soma de fracções . . . 314
7.3.4 Encapsulamento e categorias de acesso . . . 318
7.3.5 Rotinas membro: operações e métodos . . . 319
7.4 Classes C++ como módulos . . . 325
7.4.1 Construtores . . . 326
7.4.2 Construtores por cópia . . . 332
7.4.3 Condição invariante de classe . . . 332
7.4.4 Porquê o formato canónico das fracções? . . . 334
7.4.5 Explicitação da condição invariante de classe . . . 335
7.5 Sobrecarga de operadores . . . 343
7.6 Testes de unidade . . . 347
7.7 Devolução por referência . . . 351
7.7.1 Mais sobre referências . . . 351
7.7.2 Operadores ++ e -- prefixo . . . 358
7.7.3 Operadores ++ e -- sufixo . . . 361
7.8 Mais operadores para o TAD `Racional` . . . 364
7.8.1 Operadores de atribuição especiais . . . 364
7.8.2 Operadores aritméticos . . . 368
7.9 Construtores: conversões implícitas e valores literais . . . 370
7.9.1 Valores literais . . . 370
7.9.2 Conversões implícitas . . . 371
7.9.3 Sobrecarga de operadores: operações ou rotinas? . . . 372
7.10 Operadores igualdade, diferença e relacionais . . . 372
7.10.1 Inspectores e interrogações . . . 373
7.10.2 Operadores de igualdade e diferença . . . 375
7.10.3 Operadores relacionais . . . 376
7.11 Constância: verificando erros durante a compilação . . . 377
7.11.1 Passagem de argumentos . . . 378
7.11.2 Constantes implícitas: operações constantes . . . 383
7.11.3 Devolução por valor constante . . . 387

7.11.4 Devolução por referência constante 390

7.12 Reduzindo o número de invocações com `inline` 391

7.13 Optimização dos cálculos com racionais 395

7.13.1 Adição e subtracção 396

7.13.2 Multiplicação 399

7.13.3 Divisão 400

7.13.4 Simétrico e identidade 401

7.13.5 Operações de igualdade e relacionais 401

7.13.6 Operadores especiais 402

7.14 Operadores de inserção e extracção 403

7.14.1 Sobrecarga do operador << 404

7.14.2 Sobrecarga do operador >> 407

7.14.3 Lidando com erros 408

7.14.4 Coerência entre os operadores << e >> 411

7.14.5 Leitura e escrita de ficheiros 413

7.15 Amizades e promiscuidades 417

7.15.1 Rotinas amigas 417

7.15.2 Classes amigas 419

7.15.3 Promiscuidades 419

7.16 Código completo do TAD `Racional` 420

7.17 Outros assuntos acerca de classes C++ 434

7.17.1 Constantes membro 434

7.17.2 Membros de classe 435

7.17.3 Destrutores 437

7.17.4 De novo os membros de classe 439

7.17.5 Construtores por omissão 440

7.17.6 Matrizes de classe 443

7.17.7 Conversões para outros tipos 443

7.17.8 Uma aplicação mais útil das conversões 446

8 Programação baseada em objectos 449

8.1 Desenho de classes 451

9 Modularização de alto nível 453

9.1 Modularização física . . . 455
9.1.1 Constituição de um módulo . . . 456
9.2 Fases da construção do ficheiro executável . . . 456
9.2.1 Pré-processamento . . . 457
9.2.2 Compilação . . . 464
9.2.3 Fusão . . . 466
9.2.4 Arquivos . . . 469
9.3 Ligação dos nomes . . . 473
9.3.1 Vantagens de restringir a ligação dos nomes . . . 474
9.3.2 Espaços nominativos sem nome . . . 477
9.3.3 Ligação de rotinas, variáveis e constantes . . . 479
9.3.4 Ligação de classes C++ e tipos enumerados . . . 486
9.4 Conteúdo dos ficheiros de interface e implementação . . . 490
9.4.1 Relação entre interface e implementação . . . 491
9.4.2 Ferramentas de utilidade interna ao módulo . . . 491
9.4.3 Rotinas não-membro . . . 492
9.4.4 Variáveis globais . . . 492
9.4.5 Constantes globais . . . 492
9.4.6 Tipos enumerados não-membro . . . 493
9.4.7 Classes C++ não-membro . . . 494
9.4.8 Operações (rotinas membro) . . . 494
9.4.9 Atributos de instância . . . 495
9.4.10 Variáveis membro de classe . . . 495
9.4.11 Constantes membro de classe . . . 496
9.4.12 Classes C++ membro (embutidas) . . . 497
9.4.13 Enumerados membro . . . 497
9.4.14 Evitando erros devido a inclusões múltiplas . . . 498
9.4.15 Ficheiro auxiliar de implementação . . . 500
9.5 Construção automática do ficheiro executável . . . 500
9.6 Modularização em pacotes . . . 507
9.6.1 Colisão de nomes . . . 507

9.6.2 Espaços nominativos 509
9.6.3 Directivas de utilização 511
9.6.4 Declarações de utilização 512
9.6.5 Espaços nominativos e modularização física 513
9.6.6 Ficheiros de interface 515
9.6.7 Ficheiros de implementação e ficheiros auxiliares de implementação . . . 515
9.6.8 Pacotes e espaços nominativos 516
9.7 Exemplo final 517

10 Listas e iteradores 523
10.1 Listas 523
10.1.1 Operações com listas 524
10.2 Iteradores 527
10.2.1 Operações com iteradores 528
10.2.2 Operações se podem realizar com iteradores e listas 529
10.2.3 Itens fictícios 530
10.2.4 Operações que invalidam os iteradores 532
10.2.5 Conclusão 533
10.3 Interface 535
10.3.1 Interface de `ListaDeInt` 535
10.3.2 Interface de `ListaDeInt::Iterador` 540
10.3.3 Usando a interface das novas classes 543
10.3.4 Teste dos módulos 545
10.4 Implementação simplista 547
10.4.1 Implementação de `ListaDeInt` 547
10.4.2 Implementação de `ListaDeInt::Iterador` 550
10.4.3 Implementação dos métodos públicos de `ListaDeInt` 552
10.4.4 Implementação dos métodos públicos de `ListaDeInt::Iterador` . . . 556
10.5 Uma implementação mais eficiente 558
10.5.1 Cadeias ligadas 559

11 Ponteiros e variáveis dinâmicas 563

12 Herança e polimorfismo 565

13 Programação genérica **567**

14 Excepções e tratamento de erros **569**

A Notação e símbolos **587**

A.1 Notação e símbolos 587

A.2 Abreviaturas e acrónimos 590

B Um pouco de lógica **591**

C Curiosidades e fenómenos estranhos **593**

C.1 Inicialização 593

C.1.1 Inicialização de membros 594

C.2 Rotinas locais 594

C.3 Membros acessíveis ”só para leitura” 595

C.4 Variáveis virtuais 597

C.5 Persistência simplificada 603

D Nomes e seu formato: recomendações **615**

E Palavras-chave do C++ **617**

F Precedência e associatividade no C++ **621**

G Tabelas de codificação ISO-8859-1 (Latin-1) e ISO-8859-15 (Latin-9) **625**

H Listas e iteradores: listagens **633**

H.1 Versão simplista 633

H.1.1 Ficheiro de interface: `lista_int.H` 633

H.1.2 Ficheiro de implementação auxiliar: `lista_int_impl.H` 638

H.1.3 Ficheiro de implementação: `lista_int.C` 643

I Listas e iteradores seguros **651**