



- ▶ Procure um identificador pelo seu nome e não pela categoria à qual ele pertence. Por exemplo, não tente encontrar *função printf()*; em vez disso, procure diretamente *printf()*.
- ▶ Número de página em negrito significa que a respectiva informação procurada está em nota de rodapé.

Símbolos

- | | (barra vertical dupla) disjunção, operador de 49
- && (e comercial duplo) conjunção, operador de 49–50
- & (e comercial) endereço, operador de 69
- \$ (cifrão) makefile, usado com 102
- "" (aspas)
 - cabeçalho, delimitadoras de 54
 - string, uso em iniciação de 129
- * (asterisco)
 - acesso, operador de 139–140
 - indireção, operador de 70–71
 - ponteiro, definidor de 69
- */ (asterisco barra) comentário, delimitador de 53
- *= (asterisco igual) multiplicação com atribuição, operador de 51
- /* (barra asterisco) comentário, delimitador de 53
- // (barra barra) comentário, delimitador de 53
- / (barra) divisão, operador de 49
- /= (barra igual) divisão com atribuição, operador de 51
- \ (barra invertida)
 - caractere de escape, usado com 47
 - diretiva, usado em 92
- # (cerquilha)
 - diretiva, início de 57
 - operador de pré-processamento 92
- { } (chaves)
 - bloco, delimitador de 58
 - variável estruturada, uso em iniciação de 121, 135, 138–139
- [] (colchete)
 - array, usado em acesso a 120
 - array, usado em definição de 120
 - indexação, operador de 139–140
- <> (colchete angular) cabeçalho, delimitador de 54
- : (dois pontos)
 - case**, usado com 64
 - default**, usado com 64
 - rótulo, usado como declarador de 66
- != (exclamação igualdade) diferença, operador de 49
- ! (exclamação) negação, operador de 49
- (final de prova) q.e.d. 563
- = (igualdade) atribuição, operador de 50
- == (igualdade dupla) igualdade, operador de 49
- ? : (interrogação dois pontos) condicional, operador 66–67
- >= (maior igual) maior ou igual, operador 49
- > (maior) maior do que, operador 49
- ++ (mais duplo) incremento, operador de 52–53
- += (mais igual) soma com atribuição, operador de 51
- + (mais) soma, operador de 49
- <= (menor igual) menor ou igual, operador 49
- < (menor) menor do que, operador 49
- = (menos igual) soma com atribuição, operador de 51
- () (parênteses)
 - acesso, operador de 139

() (parênteses) (*continuação*)
 função, usados com 82, 84, 87
 vazios 83, 87

%= (percentagem igual) resto de divisão com atribuição, operador de 51

• (ponto) acesso a campo de estrutura, operador de 135–139

; (ponto e vírgula)
 declaração, terminador de 58
for, usado com 60, 61
 instrução, terminador de 58
 instrução vazia, como 58

% (porcentagem)
 especificador de formato, usado em 55–56
 resto de divisão, operador de 49–50

– (subtração)
 usado em expressão 479
 usado em identificador 46, 92, 95–96, 573–574

-- (traço duplo) decremento, operador de 52–53

-> (traço seguido por maior)
 acesso, operador de 139–140
 campo de estrutura, usado em acesso a 135–136

... (três pontos) parâmetro, usados como 56

, (vírgula)
 separador de parâmetros, usada como 82, 84, 87
 separador de variáveis, usada como 50
 vírgula, operador 67

A

aborto de programa 49
 array, causado por acesso a 121
 divisão por zero, causado por 49
exit(), por meio de 138–139
 ponteiro nulo, causado por 71
 resto de divisão por zero, causado por 49
 stack overflow, devido a 169, 193
 string constante, causado por alteração de 130

abstração 219

acesso a campo
 de estrutura 135
 de união 138

acesso a elemento
 de array 120
 de lista 280, 282

AcrescentaC(), função 332

AcrescentaDigitoFim(), função 445

AcrescentaDigitoInicio(), função 451

AcrescentaItem(), função 306

AcrescentaListaIdxD(), função 373, 380

AcrescentaListaIdx(), função 284, 292, 293, 295

acréscimo
 em fila 326
 em lista 281
 em lista indexada 284–285

ACRESCIMO, constante simbólica 374, 377, 378, 426

acréscimo e inserção, diferença entre 281

adição 49–50

adição de ponteiro com inteiro 122–123

alargamento, conversão de 52

algoritmo 220–223. *Procure um algoritmo específico pelo nome dele*
 análise do problema 222
 como construir 221–222
 correto 220
 desempenho de 236
 dividir e conquistar, método 190
 de divisão e conquista 222
 eficiência de 236
 entrada de 220
 de força bruta 223
 funcionalmente equivalente (a outro) 220
 incorreto 220
 paradigma de 222–225. *V. também* paradigma algorítmico
 refinamento de 222
 refinamentos sucessivos, método de 190, 221
 de retrocesso 222. *V. também* retrocesso
 saída de 220
 teste de 222
 voraz 223

alocação de memória
 definição de tipo e 133
 dinâmica. *V. alocação dinâmica de memória*
 estática 50
 subdimensionamento em 362
 superdimensionamento em 362

typedef e, uso de 133

alocação dinâmica de memória
 funções de 363–366
 justificativa para 362
 teste de 369

AlteraElementoListaIdxD(), função 380

AlteraItem(), função 306

AlturaArvoreBin(), função 512

alusão de função 87

alusão de variável global 97

ambiente integrado de desenvolvimento (IDE) 98–99

análise assintótica 237–238. *V. também* análise de algoritmo

análise combinatória 561–562

análise de algoritmo
 constantes aditivas e multiplicativas em 249
 definição de 237
 espacial 236, 253–254. *V. também* custo espacial
 função comum em 246–249
 regra da soma de 250
 regra de transitividade de 250
 regra do produto de 250
 temporal 236. *V. também* custo temporal; *V. também* análise temporal

análise temporal. *V. também* custo temporal
 de algoritmo recursivo 254–255
 de chamada de função 252–253
 de desvio condicional 251
 de desvio incondicional 252

- análise temporal (*continuação*)
 - de estrutura de controle 251–252
 - de instrução **return** 252
 - de instrução simples 251
 - de laço de repetição 252
 - regras práticas de 250–253
 - de sequência de instruções 251
- ancestral, nó 501
- AnexaEmExpressao()**, função 489
- aninho
 - de bloco 91
 - de comentário 53
 - de estrutura 136–137
 - de estrutura de controle 62
 - de laço de repetição 62, 66
 - em registro variante 138–139
 - de união 138–139
- ANSI C, padrão 46
- antecessor 280
- AnteriorListaDEC()**, função 423
- AnteriorListaDE()**, função 416
- ApresentaGrade()**, função 207
- ApresentaLista()**, função 305
- ApresentaMenu()**, função 302, 347
- argc**, parâmetro 132
- argumento. *V.* parâmetro
- argumento de linha de comando 132–133
- argv**, parâmetro 133
- Aridade()**, função 491
- aritmética de ponteiros 122–123
- arquivo
 - de cabeçalho 54, 97–98, 103–104
 - inclusão de 54
 - de programa 97. *V. também* arquivo de implementação
- arquivo binário
 - escrita em 312–313, 409
 - Tudor.bin** 308
- arquivo de implementação 97–98, 103
 - leitura de dados robusta 103–106
 - número complexo 226, 228
- arquivo de texto
 - escrita em 202, 312–313
 - leitura em 130, 311–312, 381–386
 - Tudor.txt** 307
- arranjo 561
- array
 - acesso com índice de 120
 - acesso com ponteiro de 123–124
 - acesso direto 280
 - de caracteres 129–130
 - const**, qualificado com 127
 - definição de 120
 - dimensão de 127
 - dinâmico. *V.* array dinâmico
 - de duração automática 121–122
 - de duração fixa 121
 - elemento de 127
 - endereço de 123
 - estático 280, 369
 - indexação de 120–122
 - índice de 120
 - multidimensional 127–129
 - como parâmetro de função 125–129
 - ponteiro, acesso com 123–124
 - ponteiro e 123–124
 - de ponteiros 132–133
 - como retorno de função 126–127
 - static**, qualificado com 121
 - de strings 132–133
 - unidimensional 127
 - zumbi e 126–127
- array dinâmico
 - análise 380
 - custo espacial de 380
 - custo temporal de 380
 - deficiência 398
 - fila implementada com 378–380
 - lista implementada com 369–376
 - pilha implementada com 376–378
 - realloc()** e 370
 - redimensionamento de 365–366, 371–373, 375
- árvore
 - 0/2 503
 - altura de 501
 - binária. *V.* árvore binária
 - conceitos fundamentais 500–502
 - de decisão 524
 - definição de 500
 - estritamente binária 503
 - de expressão 509
 - floresta e 501
 - folha de 501
 - grau de 501
 - grau de nó de 501
 - nível de nó de 501
 - nó de 501
 - nó descendente de 501
 - nó terminal de 501
 - profundidade de 501
 - de recursão 199, 255–260
 - representação em forma de lista de 501–502
 - representação gráfica de 501
 - terminologia de 500
- árvore binária 502–506
 - 0/2 503
 - baseadas em caminhamento 514–518
 - caminhamento 506–510. *V. também* caminhamento em árvore binária
 - clonagem de 514
 - completa 504–506. *V. também* árvore binária completa
 - conversão de árvore ordinária e floresta em 521–523
 - costurada 518–521

árvore binária (*continuação*)
 degenerada 503
 equivalência de 513
 estritamente binária 503
 expressão aritmética representada em 509
 igualdade de 513
 implementação de 510–514. *V. também* implementação de
 árvore binária
 inclinada 503
 número de folhas de 503, 504, 569, 570
 número de nós de 503, 514, 570
 número de nós de grau dois de 570
 número de nós de grau n de 503
 número máximo de nós em 503, 569
 número máximo de nós em nível de 503
 patológica 503
 perfeita 503
 profundidade de 503, 504, 570
 repleta 503–504
 semelhança de 513
 subárvore direita de 502
 subárvore esquerda de 502
 árvore binária completa
 array, implementada com 505–506, 571
 filho de nó de 505
 pai de nó de 505
 profundidade de 504, 571
 aspas (símbolo)
 array com string, uso em iniciação de 129–132
 cabeçalho, como delimitadoras de 54
ASSEGURA, macro 294
 assembler 121
 associatividade 553
 asterisco (símbolo)
 acesso, operador de 139–140
 comentário, usado em 53
 indireção, operador de 70–71
 ponteiro, definidor de 69
atan2(), função 148, 150
 atribuição
 aritmética, operador de 51
 conversão de 52
 dependência de implementação causada por 51
 efeito colateral em 50–51
 entre estruturas 135
 expressão de 50
 instrução de 50
 operador de 50–51
AtualizaArquivoBin(), função 312
AtualizaArquivoBinLSE(), função 409
auto, qualificador de tipo 90
 autorreferência de estrutura 134–135, 400
AvalExpressaoInfixa(), função 492
AvalExpressaoSufixa(), função 490
 avaliação de expressão com curto-circuito 50
 avaliação de operandos, ordem de 49, 51, 67
AVALIAEXPRESSAOSUFIXA, algoritmo 480

B

backtracking. *V. retrocesso*
 barra inversa (símbolo) 47–48
 barra (símbolo)
 comentário, usada em delimitador de 53
 divisão, operador de 49
 barra vertical (símbolo) 49
 base de indução 562
 base numérica 47, 56
 biblioteca 53–54
 cabeçalho de 54
 componente de 53–55
 módulo de 54
 padrão de C 53–54
 bit de sinal 67
 bloco
 escopo de 91–92
 de instruções 58
 de memória 363
 bolha, método de ordenação da 143–144, 484–485
break, instrução 65–66
bsearch(), função 252, 476–478
BUBBLESORT, algoritmo 143. *V. também* bolha, método de
 ordenação da
BubbleSort(), função 143, 261
BubbleSortGen(), função 484
 buffer 88–89
 busca 281, 286
 binária 262–263, 289
 chave de 286
 sequencial 287. *V. também* busca sequencial
 em string 131
BuscaBinaria2(), função 292
BUSCABINARIA, algoritmo 290
BuscaBinaria(), função 262, 290
BuscaBinariaNome(), função 310
BuscaBinariaRec(), função 291, 293
BuscaItem(), função 305
BuscaListaDECCab(), função 425
BuscaListaIdx(), função 287, 293
BuscaListaSEC(), função 420
BuscaListaSE(), função 406
BuscaListaSEOrd(), função 413
 busca sequencial 287
 custo temporal de 288
 em lista encadeada com cabeça 425
 em lista indexada 286–288
 em lista simplesmente encadeada 406, 413
BUSCASEQUENCIAL, algoritmo 287
BuscaSequencialMatr(), função 310

C

%c, especificador de formato 55, 56

- C (linguagem) 46
 - ANSI da, padrão 46
 - biblioteca padrão da 53–55
 - extensão **void main()** da 133
 - história da 46
 - padrão da 46
 - programa simples em 68
- C11, padrão 46
- C89, padrão 46
- C99, padrão 46
- cabeçalho de biblioteca 54
 - <ctype.h> 132
 - leitura.h 103
 - <math.h> 54
 - <stdlib.h> 131
 - <string.h> 130
- cabeçalho de função 82–83
- cadeia recursiva 166
- calloc()**, função 363
- caminhamento em árvore binária 506
 - exemplo de 507–510
 - infixo 507
 - em largura 510
 - por nível 510
 - prefixo 506
 - sufixo 507
- CaminhamentoInfixo2()**, função 518
- CAMINHAMENTOINFIXO**, algoritmo 507
- CaminhamentoInfixoCost()**, função 520
- CaminhamentoInfixo()**, função 512
- CAMINHAMENTOPREFIXO**, algoritmo 507
- CaminhamentoPrefixoCost()**, função 521
- CaminhamentoPrefixo()**, função 512
- CaminhamentoSufixo()**, função 512
- CAMINHAMENTOSUFIXO**, algoritmo 507
- CaminhamentoSufixoCost()**, função 521
- campo 308
 - acesso a 135–136
 - alternativo 138–139
 - de estrutura 138–139
 - fixo 138–139
 - indicador 139
 - iniciação de 135
 - de registro variante 138–139
 - variante 138–139
- caractere 47
 - classificação de 132
 - constante 47–48
 - escape, de 47
 - letra maiúscula/minúscula de, conversão em 132
 - nulo 129
 - representação gráfica 47
 - sequência de escape, representado como 47
 - terminal 129
 - transformação de 132
- caracteres, array de 129–130
- CartesianoParaPolar()**, função 150
- casamento de parênteses, colchetes e chaves 339
- Casam()**, função 341
- case**, parte de instrução **switch** 63–64
- caso de entrada
 - mediano 245
 - médio 245
 - melhor 245
 - pior 245
- casting 52
- cerquilha (símbolo) 57
- chamada de função 49, 84–87
 - alusão e 87
 - em atribuição 84
 - epílogo de 170
 - em expressão 84
 - como instrução isolada 84
 - main()** 68, 132–133
 - passagem de parâmetro em 126–130
 - prólogo de 170
 - registro de ativação e 167–170
 - retorno de valor em 84
- char*** (ponteiro para **char**), tipo 123, 130
- char**, tipo 46
- chave
 - de busca 286
 - de ordenação 288
 - primária 287
 - secundária 287
- chaves (símbolo)
 - array, uso em iniciação de 121, 127–128
 - bloco, uso em delimitação de 58
 - estrutura, uso em iniciação de 135
 - união, uso em iniciação de 138–139
- cliente, programa- 218, 219–220, 324–325
- codificação de Huffman 525–527
 - algoritmo de 527
 - código livre de prefixo e 526
 - decodificação de 538
 - estrutura de dados usada em 528
 - frequência de uso de caracteres em 526
 - implementação de 528–536
- CODIFICAÇÃODEHUFFMAN**, algoritmo 527
- CodificaHuff()**, função 532
- código livre de prefixo 526
- colação de caracteres 131–162
- colchete angular (símbolo) 54
- colchetes (símbolo)
 - array, uso em acesso a 120
 - array, uso em definição de 120
 - indexação, uso como operador de 139–140
- combinação 562
- comentário
 - delimitador de 53
 - de makefile 102
 - de programa 53
- comparação de números reais 107
- comparação, função de 477–478

- ComparaDoubles()**, função 107
- Compara()**, função 525
- ComparaInteiros()**, função 446
- ComparaModulos()**, função 447
- compatibilidade
 - em conversão de tipo 70
 - entre ponteiros 70
 - entre tipos primitivos 51
- compilação condicional 94
- compilação de programa 68–69, 99
- compilador GCC 69, 99–100
- complexidade de algoritmo 236. *V. também* análise de algoritmo
- complexidade espacial 253–254. *V. também* custo espacial
- complexidade temporal. *V. análise temporal; V. também* custo temporal
- comprimento
 - de lista 280
 - de lista encadeada com cabeça 424–425
 - de lista simplesmente encadeada 401
 - de string 130, 200
- ComprimentoListaDECCab()**, função 424
- ComprimentoListaIdxD()**, função 380
- ComprimentoListaIdx()**, função 282, 292
- ComprimentoListaSE()**, função 401, 536
- ComprimentoStrRec()**, função 200
- concatenação de strings 130
 - múltipla 389–396
- ConcatenaNVEzes()**, função 390
- condicional, operador 66–67
- conjectura 255
- conjectura de Goldbach 118
- constante 47–48
 - caractere 47–48
 - de enumeração 57–58
 - inteira 47–48
 - ponteiro 124–125
 - real 47–48
 - simbólica 57. *V. também* constante simbólica
 - string 130–132
 - variável 124–125
- constante simbólica. *Procure uma constante simbólica específica pelo nome dela*
- #define** e 57
- definição de 57
- notação para escrita de 574
- const**, qualificador de tipo 124
 - aplicado a conteúdo apontado 124–125
 - aplicado a parâmetro 125, 126, 127, 137
 - aplicado a ponteiro 124–125
 - aplicado a variável 124–125
- ConstroiNoArvoreBin()**, função 511
- ConstroiNoArvoreCost()**, função 519
- construtor de tipo 140
- const void ***, tipo 476
- contagem
 - laço de 61–62
 - de referência 225
 - variável de 61
- conteúdo de memória 70
 - de buffer 88–89
 - constante 124–125, 127, 130–132
 - indeterminado 90, 121
 - de parâmetro 85–87
 - ponteiro, referenciado por 70–71
- continue**, instrução 65–66
- conversão de árvore ordinária em árvore binária 521–523
- conversão de string em número 131–132
- conversão de tipo
 - de alargamento 52
 - aritmética usual 52
 - de atribuição 52–80
 - automática 51–52
 - casting 52–55
 - explícita 52–55
 - hierarquia de 52
 - em passagem de parâmetro 85
 - regra de 52
 - em retorno de função 84
- conversão entre letra maiúscula e minúscula 132
- CONVERTEALGORITMOSRECURSIVOS**, meta-algoritmo 333
- CONVERTEEXPRESSÃOINFIXAEMSUFIXA**, algoritmo 483
- coordenada angular 148
- coordenada radial 148
- CopiaArvoreBin()**, função 514
- cópia de parâmetro 85–87
- cópia de string 130
- CopiaListaG()**, função 470
- corolário
 - 6.1 $[f(n) \text{ é } \theta(g(n)) \text{ se e somente se } f(n) \text{ é } O(g(n)) \text{ e } g(n) \text{ é } O(f(n))]$ 243, 566
 - 6.2 $[f(n) \text{ é } \theta(g(n)) \text{ se e somente se } f(n) \text{ é } \Omega(g(n)) \text{ e } g(n) \text{ é } \Omega(f(n))]$ 243, 566
 - 12.1 (número de folhas de árvore binária perfeita) 504, 570
- corpo de função 83
- corpo de laço 59
- corrupção de memória 121
- cos()**, função 148
- criação
 - de fila 326, 327
 - de lista 280, 370
 - de lista indexada 374
 - de pilha 324
- CriaComplexo()**, função 224, 226
- CriaFilaIdxD()**, função 379
- CriaFilaIdx()**, função 327
- CriaFilaSE()**, função 429
- CriaInteiro()**, função 443
- CriaListaDECCab()**, função 424
- CriaListaDeSoldados()**, função 441
- CriaListaIdxD2()**, função 374
- CriaListaIdxD()**, função 371, 380
- CriaListaSEOrd()**, função 529

CriaPilha(), função 518
CriaPilhaG(), função 473
CriaPilhaIdxD(), função 377
CriaPilhaIdx(), função 324
CriaPilhaSE(), função 427
 crivo de Eratóstenes 387–389
<ctype.h>, cabeçalho 54, 132
 curto-circuito de operador 49, 50
 custo
 constante 246
 cúbico 247
 espacial. *V.* custo espacial
 exponencial 247
 fatorial 247
 linear 247
 linear logarítmico 247
 logarítmico 247
 polinomial 247
 de processamento 236
 quadrático 247
 temporal. *V.* custo temporal
 $\theta(1)$ 246
 $\theta(c^n)$ 247
 $\theta(\log n)$ 247, 248
 $\theta(n!)$ 247
 $\theta(n)$ 247
 $\theta(n^2)$ 247
 $\theta(n^3)$ 247
 $\theta(n^c)$ 247
 $\theta(n \log n)$ 247
 custo espacial 236
 de algoritmo recursivo 253
 de array dinâmico 380
 de fila circular 332
 de lista indexada 292–293
 $\theta(1)$ 316
 $\theta(n)$ 266–267, 315
 custo temporal 236
 de operação com array dinâmico 380
 de operação com busca sequencial 288
 de operação com fila circular 332
 de operação com fila linear 329
 de operação com lista indexada 292–293
 de operação com pilha 325
 $\theta(1)$ 261
 $\theta(2^n)$ 265–266
 $\theta(\log n)$ 262–263, 268
 $\theta(n)$ 261
 $\theta(n^2)$ 261–262, 315
 $\Omega(1.5^n)$ 266–267

D

%d, especificador de formato 55, 56
 data, leitura e validação de 144–145

declaração de função 87
 decremento, operador de 52–53
default, parte de instrução **switch** 65
#define, diretiva 57, 92
 definição
 de array 120–122
 de constante simbólica 57
 de estrutura 134
 de função 82–87
 de ponteiro 69–70
 de tipo 133
 de união 138–139
 de variável 50, 219. *V. também* definição de variável
 definição de variável
 const, com uso de 124
 objetivo de 50
 static, com uso de 91–92
 definidor de tipo 69, 140
 de array 120–122
 de enumeração 57
 de estrutura 134
 de função 82
 de ponteiro 69–70
 de união 138–139
DELTA, constante simbólica 107
 descendente, nó 501
Desempilha(), função 518
DesempilhaG(), função 475
DesempilhaIdx(), função 325
 desempilhamento 322. *V. também* pilha
DesempilhaSE(), função 427
DesenfileiraIdx(), função 328
 desenfileiramento 326. *V. também* fila
DesenfileiraSE(), função 429
 desfazer (undo) 343–352
Desfaz(), função 349
DestroiArvoreBin(), função 514
DestroiComplexo(), função 224
DestroiFilaIdxD(), função 379
DestroiFilaSE(), função 429
DestroiListaDECCab(), função 425
DestroiListaDEC(), função 423, 445
DestroiListaIdxD(), função 371
DestroiListaSEC(), função 420
DestroiListaSE(), função 407, 430
DestroiNumero(), função 445
DestroiPilhaG(), função 474
DestroiPilhaIdxD(), função 377
 destruição
 de árvore binária 514
 de fila 379
 de lista 371
 de lista encadeada 407
 de pilha genérica 474
 de pilha indexada 377

desvio condicional 62–65
if-else 62–63
switch-case 63–65
desvio incondicional 65–66
break 65, 66
continue 65, 66
goto 65–66
desvio múltiplo 63–65
diagrama de recursão 164
dicionário 322
dígito, classificação de caractere como 132
diretiva de pré-processamento 92–96
DISPONIVEL, constante de enumeração 179
dispositivo periférico, buffer associado a 88–89
divisão por zero 49
dois pontos (símbolo)
case, uso com 64
default, uso com 64
operador condicional, uso em 66–67
rótulo, uso como declarador de 66
(double), operador de conversão 52
double, tipo 46
do-while, instrução 60, 65, 66
duração de variável 90–91

E

%e, especificador de formato 55
%E, especificador de formato 55
e comercial (símbolo)
conjunção, uso em operador de 49–50
endereço, uso como operador de 69
economia de memória 138–139
editor de ligações 98
efeito colateral 48, 50, 52–53
EhAbertura(), função 341
EhAnoBissexto(), função 145
EhFechamento(), função 341
EhFeliz(), função 432
EhOperador(), função 489
elemento de array
acesso de 120–121, 123–124
iniciação de 121–122, 127
ElementoFrenteIdx(), função 328
ElementoFrenteSE(), função 429
ElementoTopoG(), função 476
ElementoTopoIdx(), função 324
ElementoTopoSE(), função 427
#elif, diretiva 94
#else, diretiva 94
else, parte de instrução **if** 62–63
EmArray2(), função 192
EmArray(), função 191, 315
EmArrayRec(), função 171, 191
EmpilhaDouble(), função 473
Empilha(), função 518
EmpilhaG(), função 474

EmpilhaIdxD(), função 378
EmpilhaIdx(), função 325
EmpilhaInt(), função 472
empilhamento 322. *V. também* pilha
EmpilhaSE(), função 427
encapsulamento de dado 219
encerramento de execução de programa
anormal. *V. aborto de programa*
por meio de **exit()** 282
normal 68
encerramento de laço de repetição
break, por meio de 65–66
for 61
infinito 62–66
while 59
endentação 83
endereço
de array 123
de função 465–466
de string constante 130
de variável 69
#endif, diretiva 94
EnfileiraIdxD(), função 379
EnfileiraIdx(), função 328
enfileiramento 326. *V. também* fila
EnfileiraSE(), função 429
[ENTER], tecla
leitura de dados com, encerramento de 56, 88
'\n', representada como 88
quebra de linha, como representação de 88
remoção de buffer de caractere correspondente a 88–89
entrada, meio de 87–88
enumeração 57–58
enum, palavra-chave 57
epílogo de função 170
equação de recorrência 254. *V. também* relação de recorrência
Eratostenes(), função 387
Eratóstenes, peneira de 386
erro de programação
de execução. *V. aborto de programa*
de lógica 126–127
overflow, devido a 52
zumbi, causado por 126–127
EscolheFuncao2(), função 467
EscolheFuncao(), função 466
escopo 91
EscreveEspacos(), função 341
escrita de dados na tela 54–55
especificador de formato 56
%c (família printf) 55
%c (família scanf) 56
%d (família printf) 55
%d (família scanf) 56
%E (família printf) 55
%f (família printf) 55
%g (família printf) 55

- especificador de formato (*continuação*)
 - %G (família printf) 55
 - %i (família printf) 55
 - %i (família scanf) 56
 - %lf (família scanf) 56
 - %s (família printf) 55
 - %s (família scanf) 56
- EstaCheiaListaIdx()**, função 284
- EstaDisponivel()**, função 206
- EstaVaziaFilaSE()**, função 429
- EstaVaziaListaIdxD()**, função 380
- EstaVaziaListaIdx()**, função 286, 292
- EstaVaziaListaSE()**, função 401
- EstaVaziaPilhaSE()**, função 427
- estilo de programação 573–576
- estrutura de controle 59–66
 - break** 65–66
 - continue** 65–66
 - desvio condicional 59, 62–65
 - desvio incondicional 59
 - de desvios múltiplos 63
 - do-while** 60–62
 - fluxo de execução e 59
 - for** 60–62
 - goto** 65–66
 - de repetição 59–66
 - de seleção 63
 - switch-case** 63–66
 - while** 59–62
- estrutura de dados 220
 - disciplina 218
 - genérica 472–473, 479
 - hierárquica 500. *V. também* árvore
 - linear 500, 506. *V. também* fila; *V. também* lista; *V. também* pilha
 - não linear 506. *V. também* árvore
- estrutura (variável) 134
 - aninhada 136–137
 - atribuição de 135
 - com autorreferência 134, 400
 - campo de, acesso a 135–136
 - definição de 134
 - iniciação de 135
 - como parâmetro de função 137
 - como retorno de função 137–139
- EsvaziaPilha()**, função 351
- exceção 293
 - captura de 293
 - condição de 293
 - lançamento de 293
 - tratamento de 294–296. *V. também* tratamento de exceção
- exclamação (símbolo)
 - diferença, uso em operador de 49
 - negação, uso como operador de 49
- exemplo de programação
 - algoritmo com custo temporal $\theta(1)$ 261
 - algoritmo com custo temporal $\theta(\log n)$ 262–263
 - algoritmo com custo temporal $\theta(n)$ 261
 - algoritmo com custo temporal $\theta(n^2)$ 261–262
 - algoritmo com custo temporal $\theta(n \log n)$ 263–264
 - avaliação de expressão sufixa 490–492
 - calculando comprimento de string recursivamente 200
 - casamento de parênteses, colchetes e chaves 339–341
 - codificação de Huffman 525–538
 - comparando números reais 107
 - concatenação múltipla de strings 389–391
 - conversão de expressão infix em sufixa 485–490
 - coordenadas retangulares e polares 148–151
 - desfazendo e refazendo 343–352
 - Eratóstenes, crivo ou peneira de 386–389
 - exibindo-se em frente e verso 202–204
 - exponenciação por quadratura 201, 268
 - Fibonacci, sequência de 198–200, 266–267
 - fila de banco, simulação de 352–355
 - invertendo entradas 202, 338–339
 - invertendo lista simplesmente encadeada 430–432
 - Josephus, problema de 440–442
 - leitura de dados resiliente 103–106, 141–143
 - linha ilimitada, leitura de 380–396
 - lista de compras 301–307
 - lista ordenada armazenada em arquivo 307–315
 - número complexo 225–229
 - número feliz 432–434
 - número inteiro ilimitado 443–453
 - oito moedas, problema das 524–525
 - operações com vetores reais 146–148
 - ordenação de arrays pelo método da bolha 143–144
 - ordenação generalizada de lista indexada 484–485
 - palíndromos 342–343
 - problema das oito moedas 524–525
 - raiz quadrada usando o método de Newton e Raphson 109–110
 - remoção de duplicatas de lista 315–316
 - remoção recursiva de vogal 200–201
 - representação de polinômio usando lista encadeada 434–440
 - série de Taylor para cálculo de seno 107–109
 - Sudoku 204–207
 - torres de Hanói 195–198, 264–266
 - validando data 144–145
- ExibeArquivoInvNaTeLaRec()**, função 203, 253
- ExibeArquivoNaTeLaRec()**, função 203
- ExibeArrayBi()**, função 129
- ExibeArrayDeChars()**, função 534
- ExibeArray()**, função 261
- ExibeArrayInts()**, função 126
- ExibeCodigosHuff()**, função 534
- ExibeComplexo()**, função 226, 228
- ExibeDoFinal()**, função 417
- ExibeDoInicio()**, função 408
- ExibeFila()**, função 354
- ExibeLista()**, função 314
- ExibeListaG()**, função 472

ExibeNumero(), função 453
ExibePolinomio(), função 439
ExibePonto(), função 149
ExibeTermo(), função 439
 exibição de arquivo em frente e verso 202–204
exit(), função 282
 expansão de macro 92
 expoente 47
 exponenciação por quadratura 201, 268
ExponenciacaoQuad(), função 201, 223, 268
 expressão 49

- aritmética 49. *V. também* expressão aritmética
- árvore de 509
- de comparação 49
- constante 64
- lógica 49
- relacional 49

 expressão aritmética 478

- árvore binária, representação em 509
- avaliação de forma sufixa de 480, 490–492
- conversão de forma infixada para forma prefixada de 479–480
- conversão de forma infixada para forma sufixada de 479, 480–483, 485–490
- infixa, em forma 478
- prefixa, em forma 478
- sufixa, em forma 478

ExpressaoSufixa(), função 486
 extensão da linguagem C **void main()** 133
 extensão de arquivo

- .c 97
- .h 54, 97

extern, palavra-chave

- função global, uso com 87
- variável global, uso com 97

F

%f, especificador de formato 55
fabs(), função 107
 fator de escala 123
 fatorial 192–193
Fatorial2(), função 193
Fatorial(), função 171, 192, 194, 254
feof(), função 385
fgetc(), função 383
fgets(), função 130, 143, 311

- LeLinhaIlimitada()** vs. 382–383

Fib2(), função 199
Fib(), função 194, 199, 266
 Fibonacci (número ou sequência de) 198–200, 266–267
 FIFO, estrutura 326. *V. também* fila
 fila 325

- acrécimo em 326
- aplicações de 326
- array dinâmico, implementada com 378–380
- de banco 352–355
- circular 329. *V. também* fila circular

criação de 326
 elemento da frente de 326
 encadeada 428–430
 frente de 325
 fundo de 325
 linear 326. *V. também* fila linear
 lista encadeada, implementada com 428–430
 remoção em 326
 tipo abstrato de dado (TAD), como 378–380, 428–430
 vazia 326
FilaCheiaC(), função 332
 fila circular

- array estático, implementada com 329–332
- custo espacial de 332
- custos temporal de 332

 fila de banco, simulação de 352–355
FilaIdxCheia(), função 328
FilaIdxVazia(), função 327
 fila linear

- array estático, implementada com 326–329
- custo temporal de 329
- lista encadeada, implementada com 428–430

FilhoDireitoCost(), função 519
FilhoDireito(), função 511
FilhoEsquerdoCost(), função 519
FilhoEsquerdo(), função 511
 filho (nó) 501
 final de string 129
 floresta 501
 fluxo de execução de programa 573
for, instrução 60–62

- aninhada 62
- array, usada em acesso sequencial de 121–122
- break**, usada com 65, 66
- continue**, usada com 65–66
- iniciação de variável em 62
- instrução vazia em 61
- omissão de expressão em 61
- operador vírgula em 67

 formato de argumento passado para programa 133
 formato de inclusão de cabeçalho 54
 fórmula de Stirling 237
 fragmentação de heap 368
free(), função 224, 363, 364, 377, 383
FrequenciasDeLetras(), função 530
 função. *Procure uma função específica pelo nome dela*

- alusão de 87
- cabeçalho de 82–83
- chamada de 84–87
- de comparação 477–478
- corpo de 83
- com corpo vazio 83
- declaração de 87
- definição de 82–87
- denominação de 83, 574–576
- escopo de 91

com escopo de arquivo 91–92
 função (*continuação*)
 extern, uso com 97
 forma de 82
 global 91
 local 97
 nome de 83, 574–576
 parâmetro de 83
 protótipo de 87
 recursiva 164–165. *V. também* função recursiva
 retorno de 83–84, 126–127
 retorno de, tipo de 82–83
 static, uso com 91, 97
 zumbi e retorno de 126–127
 função polinomial 556
 função recursiva. *V. também* cadeia recursiva
 caso base de 164
 caso não recursivo de 164
 caso recursivo de 164
 condição de parada de 164
 condição terminal de 164
 diagrama de recursão de 164
 fase de acréscimo de 169–170
 fase de decréscimo de 169
fwrite(), função 312, 409

G

%g, especificador de formato 55
%G, especificador de formato 55
 GCC, compilador 69, 99–100
getchar(), função 55–56, 87–88
 Goldbach, conjectura de 118
goto, instrução 65–66
 laço de repetição aninhado, uso em 66
 rótulo usado com 66
 grau de polinômio 556
Grau(), função 437
GRAU_POR_RADIANO, constante simbólica 150
 grupo de precedência 48

H

Hanói, torres de 195–198, 264–266
 heap 167, 368
 esgotamento de 368–369
 fragmentação de 368
 hipótese indutiva 562
 história da linguagem C 46
 história de Josephus 440–442
Horner(), função 558
 Horner, método de 557–559
 hospedeiro, sistema 68
 Huffman, codificação de 525–538

I

%i, especificador de formato 55

IDE 98–99
 identificador
 caractere permitido em 46
 maiúscula e minúscula em, diferenciação de 46
 notação para escrita de 83, 133, 574, 573–576
 regra de formação de 46
 restrição de denominação de 46
 tamanho de 46
#if, diretiva 94
#ifdef, diretiva 95
if-else, instrução 62–63
#ifndef, diretiva 95
 igualdade (símbolo)
 atribuição, usada como operador de 50
 diferença, usada em operador de 49
 igualdade, usada em operador de 49
 maior ou igual, usada em operador 49
 menor ou igual, usada em operador 49
 implementação (conceito) 219
 implementação de árvore binária
 altura 512
 caminhamento 512
 clonagem 514
 construção de nó 511
 definição de tipo 510
 destruição 514
 inserção de nó 511
 número de nós, cálculo de 514
 profundidade 512
#include, diretiva 54, 95
 inclusão de arquivo 95
 de cabeçalho 54
 uso de compilação condicional em 95
#include e 95
 múltipla 95
 incompatibilidade de tipos 70
 incremento, operador de 52–53
 indexação 120
 de array 120, 124, 128
 operador de 140
 índice 280
IndiceElemento(), função 517
 indireção, operador de 70–71
INDISPONIVEL, constante de enumeração 179
 indução matemática 255
 forte 563–564
 fraca 562–563
 iniciação 50
 de array com string 129–132
 duração de variável e 90–91
 de estrutura 135
 de estrutura aninhada 136
 explícita 90–92
 implícita 90–92
 de lista 280
 de ponteiro 69
 de ponteiro com string 130–132

iniciação (*continuação*)
 de união 138–139
 de variável de duração automática 90
 de variável de duração fixa 90

IniciaComplexo(), função 228

iniciador 121
 de array 121
 de estrutura 135
 de ponteiro 69
 de string 129
 de união 138

IniciaListaIdx(), função 282, 292

IniciaListaSE(), função 401

inserção
 em lista 280
 em lista duplamente encadeada 414–415
 em lista duplamente encadeada circular 420–422
 em lista encadeada com cabeça 425
 em lista indexada 283–284, 371–372
 em lista ordenada 288–289, 295–296, 375–376, 410–412
 em lista simplesmente encadeada 401–403, 410–413
 em lista simplesmente encadeada circular 417–419

inserção e acréscimo, diferença entre 281

InserEmOrdemIdxD2(), função 375, 380

InserEmOrdemIdx(), função 288, 292, 293, 295, 309

InserEmOrdemLDECCab(), função 425

InserEmOrdemLDEC(), função 422

InserEmOrdemLDE(), função 414

InserEmOrdemLSE(), função 410, 530

InserItem(), função 305

InserListaIdxD(), função 371, 380

InserListaIdx(), função 283, 292, 293, 294

InserListaSE(), função 402

InserTermo(), função 436

instrução 58
 rotulada 66
 rótulo de 66
 de seleção 63–65
 tipo de 58
 vazia 58, 62, 67

instruções, bloco ou sequência de 58

interação dirigida por menu 301

interrogação (símbolo) operador condicional, uso em 66–67

INT_MAX, constante simbólica 145

int, tipo 46

inversão de lista simplesmente encadeada 430–432

inversão de sinal 49

InverteArray(), função 172

InverteArrayRec(), função 171

Inverte(), função 202

InverteListaSE(), função 431

invertendo entrada de dados via teclado 202, 338–339

irmão (nó) 501

isalnum(), função 132

isalpha(), função 132

isblank(), função 132

isdigit(), função 132

islower(), função 132

ispunct(), função 132

isspace(), função 132

isupper(), função 132

item. *V.* elemento

iteração 59. *V. também* laço de repetição
 conversão de recursão em 333–338
 vs. recursão 193

J

Josephus, história de 440–442

K

Knuth, D. 107, 622

L

laço de contagem 61–62

laço de repetição 59–66
 aninhado 62
do-while 60
for 60–62
 infinito 62–66
while 59–62

LeArquivo(), função 311

LeCaractere(), função 104

LeDataValida(), função 144

LeInteiro(), função 104

LeIntEntre(), função 141, 145

leitura
 em arquivo de texto 130
 de caractere 141
 de data 144–145
 de linha ilimitada 380–386
 de número inteiro entre dois valores 141
 de opção 141
 de string 141
 via teclado 55–80. *V. também* leitura de dados via teclado

leitura de dados via teclado
 de caractere 55–56
 com **getchar()** 55–56
 laço de repetição em, uso de 89–92
 de número real 56–58
 com **scanf()** 56–58

LeLinhaIlimitada(), função 381–386
fgets() vs. 382–383

lema
 B.1 (expoente e logaritmo) 568

LeNatural(), função 104

LeNaturalPositivo(), função 105

LeOpcao(), função 141

LePolinomio(), função 435

LePonto(), função 149

LeReal(), função 106

LeString(), função 141

- letra maiúscula/minúscula
 - conversão entre 132
 - distinção entre 46
- %lf**, especificador de formato 56
- liberação de memória
 - dinâmica 364–365
 - estática 90, 126
- LIFO, estrutura 322. *V. também* pilha
- <limits.h>**, cabeçalho 145
- LimpaBuffer()**, função 88, 104
- linguagem algorítmica 221
- linguagem Pascal 83, 85
- linha de comando, argumento de 132–133
- linha ilimitada 380–386
- linker 69. *V. também* editor de ligações
- lista 280–281
 - aplicação de 296–301
 - armazenada em arquivo 307–315
 - array, implementada usando. *V. lista indexada*
 - de compras 301–307
 - elemento de 280
 - item de 280
 - operação complementar de 281
 - operação essencial de 280–281
 - ordenada 288
 - ponteiro, implementada usando. *V. lista encadeada*
 - vazia 280
- lista duplamente encadeada 414–417
 - acesso sequencial invertido em 416, 423
 - circular 420–423
 - inserção em 414–415, 420–422
 - número inteiro representado usando 443–453
 - remoção em 415–416, 422–423
- lista encadeada 399. *V. também* lista simplesmente encadeada
 - com cabeça. *V. lista encadeada com cabeça*
 - dupla. *V. lista duplamente encadeada*
 - fila implementada usando 428–430
 - operações 399
 - ordenada 409–413
 - pilha implementada usando 426–427
 - tipos de 413
- lista encadeada com cabeça 424–425
 - acesso sequencial em 425
 - acesso sequencial invertido em 425
 - busca em 425
 - comprimento de 424–425
 - iniciação de 424
 - inserção em 425
 - remoção em 425
- lista generalizada 468–472
 - átomo de 468
 - cabeça de 468
 - cauda de 468
 - clonagem de 470
 - conceito de 468
 - exibição de 472
- igualdade de 470–471
- implementação de 469–472
 - nó de 469
 - profundidade de 471
 - sublista de 468
 - vazia 468
- lista indexada
 - abstração de 280
 - acréscimo em 284–286
 - array vs. 281
 - busca sequencial em 286–288
 - custo espacial de 293
 - custo temporal de 292–293
 - dinâmica 370–376
 - estática 281–292
 - implementação usando array dinâmico 369–376
 - implementação usando array estático 282–286
 - inserção em 283–284
 - sem ordenação 281–288, 370–373
 - ordenada 288–292, 373–376
 - remoção de duplicatas em 315–316
 - remoção em 285–286
 - como tipo abstrato de dado (TAD) 369–376
- lista ordenada 288
 - encadeada 409–413
 - indexada 288–292, 373–376
- lista simplesmente encadeada
 - acesso sequencial em 408–409
 - busca em 406, 413
 - circular 417–420
 - comprimento de 401
 - destruição de 407–408
 - iniciação de 401
 - inserção em 417–419
 - inserção em ordem em 410–413
 - inserção no início de 401–403
 - inversão de 430–432
 - linear 399–413
 - ordenada 409–413
 - permutação de Josephus implementada com 440–442
 - pilha implementada usando 426–427
 - polinômio representado usando 434–440
 - remoção em 403–406, 419–420
 - sem ordenação 399–409
 - vazia 401
- livre de prefixo (código) 526
- logaritmo 559
- long int**, tipo 131
- long long int**, tipo 443

M

- macro 92. *Procure uma macro específica pelo nome dela*
 - desvantagem de 93
 - uso de **do-while** com 93
 - expansão de 92
 - com instrução 93

macro 92 (*continuação*)
 de makefile 102
 com parâmetro 92
 sem parâmetro 92
 parênteses usados com 92
 vantagem de 93
 vazia 94

main(), função 68, 132–133
 argumento de linha de comando e 132–133
 hospedeiro, relação com 132
 com parâmetro 132–133
 parâmetro **argc** de 132
 parâmetro **argv** de 132
 portátil 133
 programa com argumento e 132–133
 programa hospedado, uso em 68, 132
 protótipo de 132
 registro de ativação de 167
 retorno de 68, 133
return 0 em, uso de 68
void como tipo de retorno de 133

MAIOR_ANO, constante simbólica 145

maior do que, operador 49

maior ou igual, operador 49

maior (símbolo)
 acesso, usado em operador de 139–140
 estrutura, usado em acesso a 135–136
 maior do que, usado como operador 49
 maior ou igual, usado em operador 49

mais (símbolo)
 incremento, usado em operador de 52–53
 soma, usado como operador de 49

make 100. *V. também* makefile

makefile 100–103
 alvo 100
 comando 100
 dependência 100
 macro 102
 variável 102

malloc(), função 204, 224, 363, 377

<math.h>, cabeçalho 54

matriz 298, 559–560
 coluna de 559
 diagonal principal de 559
 esparsa 298–301
 identidade 559
 linha de 559
 multiplicação de 560
 quadrada 298, 559
 representação de 298–301
 soma de 560
 transposta 560
 triangular 317

MAX_ELEMENTOS, constante simbólica 324, 362

MAX_NOME, constante simbólica 441

meio de entrada 87–94

memcpy(), função 475, 485

memória
 acesso com ponteiro a 69–71, 122–124
 acesso inválido de 71
 alocação de 50, 90
 compartilhamento de 138
 conteúdo constante de 124–125, 130–132
 definição de tipo e alocação de 133
 duração de variável em 90–91, 121–122
 endereço em 69
 estrutura em, representação de 137
 liberação de 90
 parâmetro em, representação de 83, 86–87
 união em, representação de 138
 zumbi e 126–127

menor do que, operador 49

menor ou igual, operador 49

menor (símbolo) 49

menos unário 49

menu
 de contexto 301
 interação dirigida por 301
 sensível ao contexto 301

meta-algoritmo 333

método de Horner 557–559

MinhaCalloc(), função 364

módulo 97

módulo de biblioteca 54
 ctype 132
 stdlib 131

moedas, problema das oito 524–525

MostraErro(), função 341

multiplicação 49

N

Newton e Raphson, método de 109

NitensFilaIdx(), função 355

nó 399
 ancestral 501
 de árvore 501
 cabeça 424
 conteúdo efetivo de 399
 descendente 501
 filho 501
 folha 501
 grau de 501
 interno (de árvore) 501
 irmão 501
 de lista encadeada 399
 de lista generalizada 469
 não terminal (de árvore) 501
 nível de 501
 terminal 501
 visita a 506

nome
 de array 123
 de campo de estrutura 134

nome (*continuação*)
 de constante de enumeração 57
 de constante simbólica 574
 de função 83, 574
 de identificador 573–574
 de parâmetro em alusão 87
 de programa-fonte recebido por programa 132
 de tipo definido pelo programador 133, 574

Norma(), função 147

notação
 camelo 573
 de identificador 573
 de número real 47, 55
 ó 238–244
 ômega 240–244
 polonesa 479
 teta 241–244

notações ó, ômega e teta
 comparação entre 244
 relação entre 242

NULL, constante simbólica 71, 363

número complexo
 implementação de 226
 interface de 225
 como tipo abstrato de dado (TAD) 223–225, 227–229
 como tipo de dado transparente 225–227

NumeroDeDiasNoMes(), função 145

NumeroDeNosArvoreBin(), função 514

número feliz 432–434

número harmônico 213

número inteiro
 escrita de 54–55
 feliz 432–434
 ilimitado 443–453
 leitura de 104–105
 perfeito 116
 sem sinal 67

número perfeito 116

número real
 comparação de 107
 escrita de 54–55
 expoente de 47
 leitura de 56–58
 mantissa de 47
 notação científica de 47
 notação convencional de 47

número sem sinal 67

NUM_MOEDAS, constante simbólica 524

O

.o, extensão de arquivo 54

ObtemElementoListaIdxD(), função 380

ObtemElementoListaIdx(), função 282, 292, 293, 294

ObtemItem(), função 305

ocorrência de constante simbólica em programa-fonte 57

ocorrência de string em string 131

octal, base 47

ocultação de informação 219

oito moedas, problema das 524–525

operador 48
 de acesso 139–140
 aritmético 49
 associatividade de 553
 de atribuição aritmética 51
 binário 48
 de chamada de função 140
 condicional 66–67
 de conjunção 49–50
 de conversão explícita 52–55
 curto-circuito de 49, 50
 de decremento 52–53
 de disjunção 49–50
 efeito colateral de 48, 50–51, 52–53
 grupo de precedência de 48
 de incremento 52–53
 de indexação 140
 indireção 70–71
 lógico 49–51
 de negação 49
 ordem de avaliação e 49
 ponto 135–136
 precedência de 48, 553
 prefixo 52–53
 relacional 49
 seta 135–136
sizeof 67, 90–91
 sufixo 52–53
 ternário 66
 unário 48
 vírgula 67

operando 48
 aridade e 48
 aritmético, de operador 49
 atribuição, do operador de 50–51
 condicional, do operador 67
 conversão implícita e 52
 curto-circuito e 49, 50
 decremento, do operador de 52
 efeito colateral e 48
 incremento, do operador de 52
 ordem de avaliação de 49
 relacional, de operador 49
 retorno de função como 82
sizeof, do operador 67–68

ordem de avaliação de operando 49

ordem de caminhamento em árvore binária 506

ordenação por borbulhamento 143–144, 484–485

organizador prévio xli

Orientacao(), função 147

P

pai (nó) 501

- palavra-chave da linguagem C 46
- auto** 90
 - break** 65–66
 - case** 63–65
 - char** 46
 - const** 124–125
 - continue** 65–66
 - default** 65
 - do** 60–62
 - double** 46
 - else** 62–63
 - enum** 57
 - extern** 87, 97
 - for** 60–62
 - goto** 65–66
 - if** 62–63
 - int** 46
 - long** 131
 - return** 83–87
 - sizeof** 67
 - static** 90
 - struct** 134
 - switch** 63–65
 - typedef** 133
 - union** 138
 - void** 82
 - while** 59–62
- palavra reservada 46
- palíndromo 342–343
- palíndromo numérico 116
- paradigma 222
- paradigma algorítmico
- de divisão e conquista 222
 - de força bruta 223
 - de retrocesso 222. *V. também* retrocesso voraz 223
- parâmetro 83, 84
- conversão de 85
 - declaração de 83
 - de entrada 84–85
 - de entrada e saída 84–85
 - formal em pilha de execução 167
 - de linha de comando 132–133
 - modo de 84
 - passagem por referência de 85
 - passagem por valor de 85
 - de saída 84–85
- parênteses (símbolo)
- acesso, usados como operador de 140
 - função, usados em 82, 84, 87
 - sizeof**, uso com 67
 - vazios 83, 87
- paridades distintas 116
- partição de memória 166–167
- código de programa em 166
 - heap 167
 - pilha de execução. *V.* pilha de execução
 - string constante em 166
 - variável de duração fixa em 166
- Pascal, linguagem 46, 83, 85
- passagem de parâmetro 85
- array multidimensional 129
 - array unidimensional 126
 - endereço de variável 86–87
 - estrutura 137
 - ponteiro 86–87
 - por referência 85, 86–87
 - por valor 85
- PC (prioridade de chegada) 483, 489
- PDP (prioridade dentro da pilha) 483, 489
- peneira de Eratóstenes 386
- pensando recursivamente 190–192
- permutação 562
- permutação de Josephus 440–442
- pilha 322
- aplicações de 323
 - array dinâmico, implementada com 376–378
 - array estático, implementada com 324–325
 - criação de 323, 324
 - custo temporal de operação em 325
 - desempilhamento 323
 - elemento do topo de 323
 - empilhamento 323
 - de execução 167–170. *V. também* pilha de execução genérica 473–476
 - lista encadeada, implementada com 426–427
 - operações com 323
 - tipo abstrato de dado (TAD), como 376–378, 426–427
 - topo de 322
 - vazia 323
- pilha de execução
- esgotamento de 169
 - registros de ativação de 167. *V. também* registro de ativação stack overflow e 169
- PilhaIdxCheia()**, função 325
- PilhaIdxVazia()**, função 324
- PilhaVazia()**, função 518
- PilhaVaziaG()**, função 474
- piso, função (matemática) 560–561
- PolarParaCartesiano()**, função 151
- polinômio 556–557
- lista indexada, representado usando 296–298
 - lista simplesmente encadeada, representado usando 434–440
- ponteiro 69
- aritmética de 122–123
 - array e 123–124
 - compatibilidade de 70
 - constante 124–125
 - para constante 124–125
 - decremento de 123
 - definição de 69
 - para função 464–468. *V. também* ponteiro para função

ponteiro (*continuação*)
 genérico 366
 incremento de 123
 indireção de 70–71
 inválido 71
 nulo 71
 opaco 223
 como parâmetro 85–87
 soma com inteiro de 122–123
 subtração com inteiro de 123
 subtração de 123

ponteiro para função 464
 atribuição de valor a 465–466
 chamada de função com 466
 definição de 464
 como parâmetro de função 467–468
 retorno de 466–467

ponto e vírgula (símbolo)
 declaração, uso como terminador de 58
for, uso com instrução 60, 61
 instrução, uso como terminador de 58
 instrução vazia, uso como 58

ponto, operador 135–136

ponto (símbolo)
 acesso, uso como operador de 139
 campo de estrutura, uso em acesso a 135–136

porcentagem (símbolo) especificador de formato, usado em 55–56

posição (índice) de elemento de lista 280

Posiciona(), função 179

pow(), função 437

precedência de operador 48, 553

PreencheSudoku(), função 204–205

pré-processor 92–96

PrimeiroElemento(), função 261

primos entre si 116

PRIMOS_POR_LINHA, constante simbólica 389

printf(), função 54–55
 sequência de escape, usada com 47
 string de formatação de 54

Prioridade(), função 489

problema de satisfação de restrições 190

ProdutoInterno(), função 147

produto, regra do (análise de algoritmo) 250

ProfListaG(), função 471

programa
 com argumento 132–133
 -cliente 218, 219–220, 324–325
 depuração de 53
 executável 68–69
 fluxo de execução de 59
 hospedado 68, 132
 multiarquivo 98. *V. também* programa multiarquivo simples 68

programa multiarquivo
 construção 98
 editor com, uso de 68, 99

make e 100
 makefile e 100

prólogo (de função) 170

prompt 55

propriedade de operador
 associatividade 553
 curto-circuito 49, 50
 efeito colateral 48, 50–51, 52–53
 ordem de avaliação de operandos 49
 precedência 48–59, 553
 resultado 48

propriedades de logaritmos 559

protótipo de função 87

ProximoListaSEC(), função 420

ProximoListaSE(), função 408

ProximoToken(), função 488

pseudocódigo 221

pseudolinguagem 221

putchar(), função 55

Q

qsort(), função 252, 476–478

QUADRADO, macro 92

quebra de linha 88

R

RADIANO_POR_GRAU, constante simbólica 149

rainhas, problema das n 174–189

raiz quadrada (método de Newton e Raphson) 109

rand(), função 54

realloc(), função 365–366
 custo de 253, 380, 398
 uso de 372, 380, 384

recomendação sobre
 array como parâmetro formal, escrita de 125
 buffer, esvaziamento de 88–89
 constante de enumeração, escrita de 57
 constante simbólica, uso de 574
const, uso de 126, 127
 identificador, escrita de 573–576
 interação com usuário 89–92
 leitura de string 130
 operador vírgula, uso de 67
 retorno de endereço 126
scanf(), leitura com 56
 usuário, interação com 89–92

ReconstróiArvoreBin(), função 517

recorrência. *V.* relação de recorrência

recursão 164. *V. também* recursão
 análise temporal de 254–255
 árvore de 255–260
 busca binária usando 291–292
 de cauda 170–173
 clareza de 193
 eficiência de 193

- recursão (*continuação*)
 - exibição de arquivo, uso em 202–204
 - exponenciação por quadratura, uso em 201
 - Fibonacci (número ou sequência de) usando 198–200
 - funcionalidade de 193
 - infinita 169
 - iteração, convertida em 333–338
 - iteração vs. 193
 - legibilidade de 193
 - pensando em 190–192
 - pilha de execução e 168–170
 - quando usar 192–194
 - recomendações práticas sobre uso de 194
 - torres de Hanói usando 195–198
- recursividade. *V.* recursão
- redo (refazer) 343–352
- refazer (redo) 343–352
- Refaz()**, função 350
- refinamentos sucessivos 221
- registro (de arquivo) 308
- registro de ativação 167–170
 - endereço de retorno e 167
 - da função **main()** 167
 - parâmetro formal e 167
 - valor de retorno e 167
 - variável de duração automática e 167
- registro variante 138–139
- regra (análise de algoritmo)
 - de produto 250
 - de soma 250
 - de transitividade 250
- relação de recorrência 254–255, 564–566
 - árvore de recursão e 255–260
 - condição inicial de 254
 - conjectura de 255
 - forma fechada de 564
 - homogênea 565–566
 - interpretação algorítmica de 255
 - método de substituição para 255
 - notação de 564
 - representação gráfica de 255–260
 - solução de 564
 - tentativa e erro em resolução de 255
- remoção
 - em fila 326
 - em lista 280
 - em lista duplamente encadeada 415–416
 - em lista duplamente encadeada circular 422–423
 - em lista duplamente encadeada circular com cabeça 425
 - em lista indexada 285–286
 - em pilha 322
- remoção de caractere no buffer de entrada padrão 88–89
- RemoveDuplicatas()**, função 315
- RemoveItem()**, função 305
- RemoveListaDEC()**, função 423
- RemoveListaDE()**, função 416
- RemoveListaIdxD2()**, função 375
- RemoveListaIdxD()**, função 372, 380
- RemoveListaIdx()**, função 285, 292, 293, 295, 315
- RemoveListaSEC()**, função 420
- RemoveListaSE()**, função 404
- RemoveVogais()**, função 200
- RemoveZerosIniciais()**, função 445
- representação
 - de caractere 47
 - gráfica de caractere 47
 - de matriz esparsa 298–301
 - em memória. *V.* representação em memória
 - de polinômio 296–301
- representação em memória
 - de estrutura 137
 - de parâmetro 83, 86–87
 - de união 138
- ResultadoOperacao()**, função 491, 492
- resultado (propriedade de operador) 48
- RetiraC()**, função 332
- retorno de função 83–84
 - de estrutura 137–139
 - expressão, usado em 84
 - return** (instrução) e 83–84
 - de **scanf()** 56
 - tipo de 82, 83–84
 - com valor 83–84
 - sem valor 83
 - void**, do tipo 83
 - zero como 68
- retrocesso 173–190
 - problema das n rainhas e 174–189
 - problema de satisfação de restrições e 190
 - problema propício ao uso de 190
 - Sudoku 204–214
- return 0**, instrução 68
- return**, palavra-chave 83–84
- robustez 55, 89–92
- Rodadas()**, função 263
- rotação de ordem k 160
- rotFilaIdxD**, rótulo 379
- rotFilaSE**, rótulo 428
- rotListaIdxD2**, rótulo 374
- rotListaIdxD**, rótulo 370
- rotNoArvoreBin**, rótulo 511
- rotNoArvoreCost**, rótulo 519
- rotNoArvoreHuff**, rótulo 528
- rotNoJosephus**, rótulo 441
- rotNoLDECCab**, rótulo 424
- rotNoLDE**, rótulo 414
- rotNoListaG**, rótulo 469
- rotNoLSE**, rótulo 400, 428, 528
- rotNoPolI**, rótulo 435
- rotPilhaIdxD**, rótulo 377
- rotPilhaSE**, rótulo 426
- rótulo de estrutura 134, 574. *Procure um rótulo de estrutura específico pelo nome dele*

rótulo de instrução 66

S

%s, especificador de formato 55, 56

SaoIguaisListasG(), função 471

SaoSemelhantesArvoresBin(), função 513

satisfação de restrições 190, 204

scanf(), função 56–58, 87–88

seleção, instrução de 63–66

sequência de escape 47

'\0' 129

'\n' 88

sequência de instruções 58

série de Taylor para cálculo de seno 107–109

série telescópica 556

seta, operador 135–136

simulação de fila de banco 352–355

sin(), função 148

sistema de execução 68

sistema polar de coordenadas 148

site deste livro: www.ulysseso.com/ed1 xli

sizeof, operador 67, 90–91

size_t, tipo 67, 363

Sobrevivente(), função 441

SomaAteN2(), função 165, 194, 198

SomaAteN3(), função 168

SomaAteN(), função 164

SomaComplexos(), função 226, 228

soma de ponteiro com inteiro 122–123

SomaDigitos(), função 450

SomaDigitosQuadrados(), função 433

SomaInteiros(), função 448, 450

soma (operação aritmética) 49

SomaPol(), função 297

soma, regra da (análise de algoritmo) 250

soma telescópica 556

somatório 555–556

SomaVetores(), função 147

sqrt(), função 148

SqrtNewton(), função 109

srand(), função 54

stack overflow 169, 193, 194

static, palavra-chave 91–92

<stdlib.h>, cabeçalho 54

strcat(), função 130

strchr(), função 131

strcmp(), função 131, 310, 311

strcoll(), função 131

strcpy(), função 130

strcpy(), implementação da função 253

string 129

array, armazenado em 129–132

array de 132–133

busca de caractere em 131

busca de string em 131

caractere em, busca de 131

casamento de 131

comprimento recursivo de 200

concatenação de 130

concatenação múltipla de 389–391

constante 130–132

conversão em número de 131–132

cópia de 130

endereço, representado como 130–132

de formatação 54

ocorrência de caractere em 131

partes (tokens) de, separação em 131

remoção de vogal em 200–201

string em, busca de 131

tokens (partes) de, separação em 131

<string.h>, cabeçalho 54, 130–161

strlen(), função 130, 200

strrchr(), função 131

strstr(), função 131

strtod(), função 131

strtok(), função 131, 312

strtol(), função 131, 491

struct, palavra-chave 134

subárvore 500

subprograma 82

subtração 49

inteira de ponteiro 122–123

entre ponteiros 123

subtração (símbolo)

usado em expressão 479

usado em identificador 46, 92, 95–96, 573–574

SubtraiDigitos(), função 452

SubtraiInteiros(), função 452

SubtraiMenor(), função 451

sucessor 280

Sudoku 204–207

sufixo de identificador 574

switch-case, instrução 63–65

T

tAcao, tipo 344

TAD (tipo abstrato de dado) 223–225. *V. também* tipo abstrato de dado (TAD)

tAluno, tipo 309, 363, 370, 373, 400

TAMANHO_BLOCO, constante simbólica 384, 391

tamanho de array 120

em definição de array 120

iniciação e 121, 129

multidimensional 127

parâmetro, recebido como 125–126

tamanho de entrada 237

tamanho de resultado de expressão 67–68

tamanho de tipo 67–68

tamanho de variável 67–68

estrutura 138–139

fator de escala e 122

sizeof, calculado com 67

- tamanho de variável (*continuação*)
 size_t e 67
 união 138–139
- TAM_INCREMENTO_PILHA**, constante simbólica 474
- tArvoreBin**, tipo 511
- tArvoreCost**, tipo 519
- tArvoreHuff**, tipo 528
- tAtomoOuLista**, tipo 469
- Taylor, série de 107–109
- tComparacao**, tipo 447
- tComplexo2**, tipo 227
- tComplexo**, tipo 223, 225
- tConteudoListaG**, tipo 469
- tCostura**, tipo 519
- tData**, tipo 136, 144
- tDesfazRefaz**, tipo 344
- tDigitosPtr**, tipo 443
- tDigitos**, tipo 443
- tDisponibilidade**, tipo 179
- tElemento**, tipo 282, 370
- teorema
 6.1 [$f(n) \in \theta(g(n))$ se e somente se $g(n) \in \theta(f(n))$] 243, 566
 6.2 (análise assintótica de polinômio) 243, 566
 6.3 (constantes multiplicativas e aditivas) 249, 567
 6.4 (regra da soma) 250, 567
 6.5 (regra do produto) 250, 567
 6.6 (função polinomial e função exponencial) 250, 568
 6.7 (regra de transitividade) 250, 568
 6.8 (custo de relação de recorrência) 260, 568
 12.1 (número máximo de nós de árvore binária) 503, 569
 12.2 (número de folhas e de nós de grau dois) 503, 569
 12.3 (relação entre número de nós e profundidade de árvore estritamente binária) 503, 570
 12.4 (número de nós de árvore estritamente binária) 503, 570
 12.5 (número de nós de grau dois de árvore binária perfeita) 503, 570
 12.6 (profundidade de árvore binária completa) 505, 571
 12.7 (índices de pai e filhos em árvore binária completa) 505, 571
- mestre 260
- terminação de identificador 574
- terminal de instrução 58
- tEstado**, tipo 334
- teto, função (matemática) 560–561
- tFilaC**, tipo 329
- tFilaIdxD**, tipo 378
- tFilaIdx**, tipo 327, 353
- tFilaSE**, tipo 428
- time()**, função 54
- <time.h>**, cabeçalho 54
- tipo. *Procure um tipo específico pelo nome dele. V. também* tipo de dado
- tipo abstrato de dado (TAD) 219–220
 fila 378–380, 428–430
 lista indexada 369–376
 número complexo 223–225, 227–229
 pilha 376–378, 426–427
- tipo de dado 218
 abstrato 219. *V. também* tipo abstrato de dado (TAD)
 agregado 218
 char 46
 char* 130
 definição de 133
 definido pelo programador 133
 derivado 133
 double 46
 embutido 46, 218
 de estrutura 134
 estruturado 218
 incompleto 223
 int 46
 long int 131
 long long 443
 opaco 220
 primitivo 46, 218
 de registro variante 138–139
 size_t 67
 transparente 220, 225–227
 typedef em definição de, uso de 133
 de união 138–139
 void 82
- tItemCompra**, tipo 302
- tItemFila**, tipo 327, 329, 378
- tItemPilha**, tipo 324
- tLetraFreq**, tipo 528
- tListaAlunos**, tipo 309
- tListaCompras**, tipo 302
- tListaDECCab**, tipo 424
- tListaDE**, tipo 414
- tListaG**, tipo 469
- tListaIdxD2**, tipo 373
- tListaIdxD**, tipo 370
- tListaIdx**, tipo 282
- tListaJosephus**, tipo 441
- tListaSE**, tipo 400, 428, 528
- tMatricula**, tipo 309
- tMoeda**, tipo 524
- tNoArvoreBin**, tipo 511
- tNoArvoreCost**, tipo 519
- tNoArvoreHuff**, tipo 528
- tNoJosephus**, tipo 441
- tNoLDECCab**, tipo 424
- tNoListaDE**, tipo 414
- tNoListaG**, tipo 469
- tNoListaSE**, tipo 400, 428, 528
- tNome**, tipo 309
- tNoPol**, tipo 435
- tNumeroPtr**, tipo 443
- tNumero**, tipo 443
- tolower()**, função 132
- tOperacao**, tipo 512
- tOperador**, tipo 485

torres de Hanói 195–198, 238, 264–266, 333–338
TORRESDEHANÓI, algoritmo 196
TorresDeHanoi(), função 197, 333
TorresIterativas(), função 334
TorresRecursivas(), função 333–334
toupper(), função 132
tPilhaGen, tipo 473
tPilhaIdxD, tipo 376
tPilhaIdx, tipo 324
tPilhaSE, tipo 426
tPolI, tipo 435
tPonto, tipo 148
tPrioridade, tipo 485
traço (símbolo)
 uso em acesso a campo de estrutura 135–136
 uso em operador de acesso 139–140
 uso em operador de decremento 52–53
tralha (símbolo) 57
transitividade, regra de (análise de algoritmo) 250
Transpoe(), função 300
tratamento de exceção 293–296
 macro **ASSEGURA** usada em 294
 precondição em 293
três pontos (símbolo), uso como parâmetro de 56
troca de valores entre variáveis 85–87
tSinal, tipo 443
tSistema, tipo 148
tTermo, tipo 435
tTipoDeDado, tipo 367
Tudor.txt, arquivo 307
tVetor, tipo 146
typedef 133

U

UltimoNo(), função 416
undo (desfazer) 343–352
união 138–139
 compartilhamento de memória de 138–139
 definição de 138
 iniciação de 138
 registro variante e 138–139
union, definidor de tipo 138–139

V

validação de dado
 esvaziamento de buffer e 88–89
 laço de repetição e 89–92
 número inteiro 89–92, 104–105
 número real 106
ValorNumerico(), função 437
variável
 agregada 218
 alusão de 97
 const, definida com 124–125
 de contagem 61

definição de 97
duração de 90–91
endereço de 69
escopo de 91
estrutura 134–139
estruturada 218
extern, declarada com 97
global 96
heterogênea 218
homogênea 218
iniciação de 50
local 91
não iniciada 90
nome de 573
nomenclatura usada com 573
ponteiro para 69–70
static, definida com 90–92, 91–92
união 138–139
vetor 146–148
VetorVezeConstante(), função 146
vírgula, operador 67
vírgula (símbolo)
 operador vírgula, uso como 67
 separador de declarações, uso como 50
 separador de parâmetros, uso como 84, 87
 separador de variáveis em definição, uso como 50
(void), operador de conversão 52
void*, tipo 363, 366–367
void, tipo 82–83
 alusão, uso em 87
 main(), usado com 87, 133
 como parâmetro de função 83
 como tipo de retorno de função 82–84

W

while, instrução 59–62, 65–66
www.ulysseso.com/ed1, site deste livro xli

Z

zumbi
 pilha de execução, habitante da 126–127

