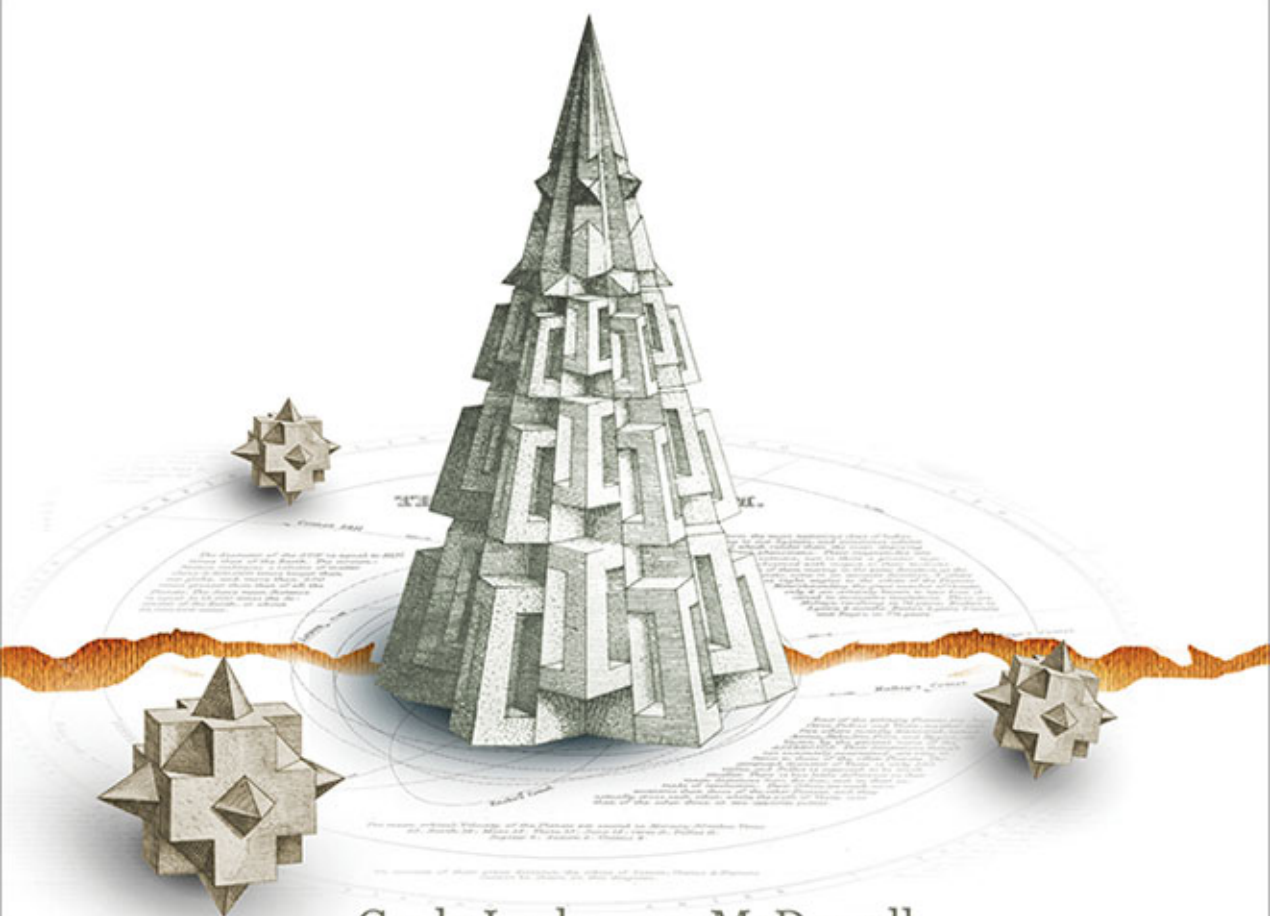


ROZMOWA REKRUTACYJNA DLA PROGRAMISTÓW

Przewodnik do sukcesu



Gayle Laakmann McDowell



Helion

Tytuł oryginału: Cracking the Coding Interview: 150 Programming Questions and Solutions

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-9332-0

Copyright © 2008 – 2013 by Gayle Laakmann McDowell.

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means, including information storage and retrieval systems, without permission in writing from the author or publisher, except by a reviewer who may quote brief passages in a review.

No part of this book may be used or reproduced in any manner without written permission except in the case of brief quotations in critical articles or reviews.

© Helion SA 2014.
All rights reserved.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/rorepr.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/rorepr>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to!» Nasza społeczność](#)

Przedmowa	7
Wprowadzenie	9
I Proces rekrutacji	11
Wprowadzenie	12
Jak wybierane są pytania?	13
Oś czasu i mapa przygotowań	14
Proces oceny	16
Błędne odpowiedzi	17
Zasady dotyczące ubioru	18
Dziesięć najczęściej popełnianych błędów	19
Często zadawane pytania	21
II Rekrutacja od kuchni	23
Rekrutacja w Microsoftzie	25
Rekrutacja w Amazonie	26
Rekrutacja w Google'u	27
Rekrutacja w Apple'u	28
Rekrutacja w Facebooku	29
Rekrutacja w Yahoo!	30
III Wyjątkowe sytuacje	31
Doświadczeni kandydaci	32
Testerzy i inżynierowie testów oprogramowania	33
Menedżerowie programów i produktów	34
Liderzy i menedżerowie zespołów programistów	36
Początkujące firmy	37
IV Przed rozmową	39
Zdobywanie odpowiedniego doświadczenia	40
Budowanie sieci kontaktów	41
Pisanie świetnych CV	43

V	Pytania behawioralne	45
	Przygotowania do pytań behawioralnych	46
	Odpowiadanie na pytania behawioralne	49
VI	Pytania techniczne	51
	Przygotowania techniczne	52
	Odpowiadanie na pytania techniczne	55
	Pięć metod tworzenia algorytmów	58
	Jakie cechy ma dobry kod?	62
VII	Otrzymałeś ofertę i co dalej?	67
	Postępowanie w sytuacji przyjęcia i odrzucenia podania	68
	Ocenianie oferty	69
	Negocjacje	71
	Na nowym stanowisku	72
VIII	Pytania z rozmów rekrutacyjnych	73
	Rozdział 1. Struktury danych	75
	Rozdział 2. Tablice i łańcuchy znaków	77
	Rozdział 2. Listy powiązane	81
	Rozdział 3. Stosy i kolejki	85
	Rozdział 4. Drzewa i grafy	89
	Różne zagadnienia i algorytmy	93
	Rozdział 5. Manipulowanie bitami	95
	Rozdział 6. Łamigłówki	99
	Rozdział 7. Matematyka i rachunek prawdopodobieństwa	103
	Rozdział 8. Projektowanie obiektowe	109
	Rozdział 9. Rekurencja i programowanie dynamiczne	113
	Rozdział 10. Skalowalność i ograniczenia pamięci	117
	Rozdział 11. Sortowanie i wyszukiwanie	123
	Rozdział 12. Testy	129
	Zagadnienia wymagające konkretnej wiedzy	137
	Rozdział 13. Języki C i C++	139
	Rozdział 14. Java	147

Spis treści

Rozdział 15. Bazy danych	153
Rozdział 16. Wątki i blokady	159
Dodatkowe problemy przeglądowe	167
Rozdział 17. Umiarkowanie trudne	169
Rozdział 18. Trudne	173
IX Rozwiązania	175
Struktury danych	177
Rozdział 1. Tablice i łańcuchy znaków	177
Rozdział 2. Listy powiązane	189
Rozdział 3. Stosy i kolejki	207
Rozdział 4. Drzewa i grafy	225
Zagadnienia i algorytmy	245
Rozdział 5. Manipulowanie bitami	245
Rozdział 6. Łamigłówki	261
Rozdział 7. Matematyka i rachunek prawdopodobieństwa	267
Rozdział 8. Projektowanie obiektowe	283
Rozdział 9. Rekurencja i programowanie dynamiczne	315
Rozdział 10. Skalowalność i ograniczenia pamięci	339
Rozdział 11. Sortowanie i wyszukiwanie	357
Rozdział 12. Testy	375
Zadania oparte na wiedzy	383
Rozdział 13. Języki C i C++	383
Rozdział 14. Java	397
Rozdział 15. Bazy danych	405
Rozdział 16. Wątki i blokady	413
Zadania dodatkowe	425
Rozdział 17. Umiarkowanie trudne zadania	425
Rozdział 18. Trudne zadania	455
Podziękowania	485
Skorowidz	487
O autorce	493

1

Tablice i łańcuchy znaków

Mam nadzieję, że wszyscy czytelnicy tej książki potrafią używać tablic i łańcuchów znaków, dlatego nie będę zanudzać was tymi zagadnieniami. Zamiast tego skoncentruję się na często używanych technikach i kwestiach związanych z tymi strukturami danych.

Zwróć uwagę na to, że w zadaniach tablice często można zastąpić łańcuchami znaków i na odwrót. Oznacza to, że jeśli w książce w pytaniu występuje tablica, na rozmowie zamiast niej mogą pojawić się łańcuchy znaków i vice versa.

Tablice z haszowaniem

Tablica z haszowaniem to struktura danych, w której kluczom przyporządkowane są wartości, co pozwala na bardzo wydajne wyszukiwanie elementów. W bardzo prostej implementacji tablic z haszowaniem używane są tablica i *funkcja haszująca*. Gdy chcesz wstawić obiekt i odpowiadający mu klucz, funkcja haszująca przekształca klucz na liczbę całkowitą, która określa indeks w tablicy. Następnie obiekt jest zapisywany za pomocą tego indeksu.

Jednak zwykle rozwiązanie to nie działa poprawnie. W tej implementacji wartość skrótu dla każdego klucza musi być niepowtarzalna — w przeciwnym razie dane mogą zostać przypadkowo nadpisane. Aby zapobiec tego typu „kolizjom”, tablica musi być bardzo duża (jej wielkość powinna odpowiadać liczbie wszystkich możliwych kluczy).

Zamiast tworzyć bardzo dużą tablicę i zapisywać obiekty za pomocą indeksu `funkcja_haszujaca ↪ (klucz)`, można zbudować znacznie mniejszą tablicę i przechowywać obiekty na liście powiązanej za pomocą indeksu `funkcja_haszujaca(klucz) % dlugosc_tablicy`. W celu pobrania obiektu o określonym kluczu należy wyszukać ten klucz na odpowiedniej liście.

Inna możliwość to zaimplementowanie tablicy z haszowaniem za pomocą binarnego drzewa poszukiwań. Pozwala to zagwarantować czas wyszukiwania na poziomie $O(\log n)$, ponieważ drzewo może pozostawać zrównoważone. Dodatkowo pozwala to zmniejszyć zużycie pamięci, ponieważ nie trzeba alokować dużej tablicy na początku pracy programu.

Zalecam, aby przed rozmową przećwiczyć implementowanie i używanie tablic z haszowaniem. Jest to jedna ze struktur danych najczęściej używanych w trakcie rozmów rekrutacyjnych. Jest niemal pewne, że w trakcie jednej z rozmów natrafisz na tę strukturę.

Poniżej znajduje się prosty przykładowy kod w Javie ilustrujący używanie tablicy z haszowaniem.

```
1 public HashMap<Integer, Student> buildMap(Student[] students) {
2     HashMap<Integer, Student> map = new HashMap<Integer, Student>();
3     for (Student s : students) map.put(s.getId(), s);
4     return map;
5 }
```

Zauważ, że choć czasem rekruter bezpośrednio prosi o zastosowanie tablicy z haszowaniem, to częściej musisz sam wywnioskować, że powinieneś wykorzystać tę strukturę do rozwiązania problemu.

Lista tablicowa (dynamicznie rozszerzana tablica)

Lista tablicowa (inaczej typ `ArrayList` lub dynamicznie rozszerzana tablica) jest tablicą, która sama zmienia rozmiar, a jednocześnie zapewnia dostęp w czasie $O(1)$. Typowa implementacja działa tak, że gdy tablica jest pełna, jej długość zostaje podwojona. Za każdym razem wymaga to $O(n)$ czasu, jednak operacja ta jest potrzebna na tyle rzadko, że złożoność i tak wynosi $O(1)$.

```
1 public ArrayList<String> merge(String[] words, String[] more) {
2     ArrayList<String> sentence = new ArrayList<String>();
3     for (String w : words) sentence.add(w);
4     for (String w : more) sentence.add(w);
5     return sentence;
6 }
```

Bufory łańcuchów znaków (typ `StringBuffer`)

Załóżmy, że chcesz scalić listę łańcuchów znaków w pokazany poniżej sposób. Jaka jest złożoność tego kodu? Dla uproszczenia przyjmij, że wszystkie łańcuchy znaków mają tę samą długość (x) i jest n takich łańcuchów.

```
1 public String joinWords(String[] words) {
2     String sentence = "";
3     for (String w : words) {
4         sentence = sentence + w;
5     }
6     return sentence;
7 }
```

Przy każdym scalaniu tworzona jest nowa kopia łańcucha znaków i dwa łańcuchy są kopiowane znak po znaku. Pierwsza iteracja wymaga skopiowania x znaków. W drugiej iteracji kopiowanych jest $2x$ znaków, w trzeciej — $3x$ znaków itd. Tak więc łącznie złożoność wynosi $O(x + 2x + \dots + nx)$, czyli $O(xn^2)$. Dlaczego nie jest to $O(xn^n)$? Ponieważ $1 + 2 + \dots + n$ wynosi $n(n+1)/2$, czyli $O(n^2)$.

Typ `StringBuffer` pozwala uniknąć tego problemu. Tworzy on tablicę wszystkich łańcuchów znaków i kopiuje dane z tablicy z powrotem do łańcucha tylko wtedy, gdy jest to konieczne.

```
1 public String joinWords(String[] words) {
2     StringBuffer sentence = new StringBuffer();
3     for (String w : words) {
4         sentence.append(w);
5     }
6     return sentence.toString();
7 }
```


Dobrym ćwiczeniem z zakresu łańcuchów znaków, tablic i ogólnych struktur danych jest zaimplementowanie własnej wersji typu `StringBuffer`.

Pytania z rozmów rekrutacyjnych

- 1.1.** Zaimplementuj algorytm określający, czy łańcuch zawiera tylko niepowtarzające się znaki. Jak wykonasz to zadanie, jeśli nie można stosować dodatkowych struktur danych? s. 178
- 1.2.** Zaimplementuj w języku C lub C++ funkcję `void reverse(char* str)`, która odwraca znaki w łańcuchu zakończonym znakiem `null`. s. 179
- 1.3.** Napisz metodę, która przyjmuje dwa łańcuchy znaków i określa, czy jeden jest permutacją drugiego. s. 179
- 1.4.** Napisz metodę, która zastępuje wszystkie spacje w łańcuchu znaków sekwencją `%20`. Przyjmij, że na końcu łańcucha dostępne jest miejsce na dodatkowe znaki, a także że znasz prawdziwą długość łańcucha. Jeśli implementujesz rozwiązanie w Javie, zastosuj tablicę znaków, aby móc wykonać operację w miejscu (bez kopiowania łańcuchów).
PRZYKŁAD:
Dane wejściowe: "Pan Jan Nowak ", 13
Dane wyjściowe: "Pan%20Jan%20Nowak" s. 181
- 1.5.** Zaimplementuj metodę, która przeprowadza prostą kompresję łańcuchów znaków opartą na zliczaniu powtarzających się liter. Na przykład metoda ma przekształcać łańcuch `aabccccaaa` na `a2b1c5a3`. Jeśli „skompresowany” łańcuch znaków nie jest mniejszy od wyjściowego, metoda powinna zwracać pierwotny łańcuch. s. 182
- 1.6.** Dany jest rysunek reprezentowany przez macierz o wymiarach $N \times N$, w którym każdy piksel jest reprezentowany za pomocą czterech bajtów. Napisz metodę, która rotuje rysunek o 90 stopni. Czy potrafisz wykonać tę operację w miejscu? s. 184
- 1.7.** Napisz algorytm, który jeśli jakiś element macierzy o wymiarach $M \times N$ ma wartość zero, ustawia cały wiersz i kolumnę tego elementu na 0. s. 186
- 1.8.** Załóżmy, że istnieje metoda `isSubstring`, która sprawdza, czy dane słowo jest podłańcuchem innego. Napisz kod, który przyjmuje dwa łańcuchy znaków (`s1` i `s2`) oraz za pomocą tylko jednego wywołania metody `isSubstring` sprawdza, czy `s2` jest rotacją słowa `s1` (na przykład `1ajkonik` jest rotacją słowa `nik1ajko`). s. 187

Dodatkowe zadania: Manipulowanie bitami (5.7), Projektowanie obiektowe (8.10), Rekurencja (9.3), Sortowanie i wyszukiwanie (11.6), C++ (13.10), Umiarkowanie trudne (17.7, 17.8, 17.14).

Skorowidz

A

aktualizowanie wyników, 355
algorytm układający puzzle, 300
algorytmy, 52, 93
 rekurencyjne, 115
 rozwiązania, 245, 261, 267, 283, 315, 339,
 357, 375
 sortowania, 123
 wyszukiwania, 126
analizowanie relacji, 110
atak siłowy, 248, 474

B

bazy danych, 153, 405
 nieznormalizowane, 153
 znormalizowane, 153
BFS, breadth first search, 90
binarne drzewo poszukiwań, 89, 233
bit, 95, 245, 249, 251
blok
 finally, 398
 try-catch, 398
blokady, 161, 163, 413
błędne odpowiedzi, 17
błędy, 19, 133
brak testów, 20
budowanie
 relacji, 72
 sieci kontaktów, 41, 42
bufor, 190
bufory łańcuchów znaków, 78
burza mózgów, 60

C

cechy dobrego kodu, 62–65
CV, 37, 43
cykliczna lista powiązana, 448

Ć

ćwiczenie przy komputerze, 19

D

debugowanie, 133
definiowanie obiektów, 110
denormalizacja, 157
 wady, 409
 zalety, 409
destruktor, 140
destruktor wirtualny, 141
DFS, depth first search, 90
dobry kod, 62–65
dodatkowe usprawnienia, 356
dopasowywanie do wzorca, 58, 369
doświadczenie, 32, 40
drzewa, 89, 225
 binarne, 89
 kompletne, 89
 niezrównoważone, 89
 pełne, 89
 trie, 90
 zrównoważone, 89
dziedziczenie, 139
dzielenie, 272
dzielenie danych, 118

E

encja, 410

F

FIFO, 86, 222

flaga, 239, 466

funkcja

free, 392

hasWon, 428

haszująca, 77, 385

malloc, 392

popAt(), 214

funkcje

wirtualne, 140

z wyrównaniem, 392

G

generowanie liczb pierwszych, 104

głowa listy powiązanej, 448

gra mastermind, 433

graf, 89, 225

I

implementacja

dobra, 63

zła, 62

implementowanie

klasy CircularArray, 402

kolejki, 86

stosu, 85

instrukcja

FULL OUTER JOIN, 408

INNER JOIN, 408

JOIN, 407

LEFT OUTER JOIN, 408

OUTER JOIN, 408

RIGHT OUTER JOIN, 408

instrukcje SQL-a, 154

interfejs Iterator, 403

inżynierowie testów oprogramowania, 33

iteracje, 192, 202, 458

J

język

C, 139, 383

C++, 139, 383

Java, 147, 397

języki programowania, 21, 44, 53

K

kandydat, 17

klasa, 139

CircularArray, 402

Rectangle, 481

Singleton, 110, 307

WordGroup, 482

klucz główny, 154

kod, 62–65

kolejki, 85, 207

kolejność in-order, 231

kombinatoryka, 323

konstruktor, 140, 398

kopia pamięci podręcznej, 354

L

liczby

Catalana, 337

pierwsze, 103

liderzy, 36

LIFO, 85

lista

jednokrotnie powiązana, 81

podwójnie powiązana, 446

powiązana, 81, 189–193, 199

tablicowa, 78

Ł

łamigłówki, 99, 261

łańcuchy znaków, 77, 177

M

manipulowanie bitami, 245

mapa przygotowań, 14, 15

matematyka, 103, 267

mechanizm refleksji, 401
 menedżerowie
 programów i produktów, 34, 35
 zespołów programistów, 36
 metoda finalize(), 149, 399
 metody
 fabryczne, 111
 tworzenia algorytmów, 58–61
 Microsoft SQL Server, 153
 mnożenie, 271
 modułowość, 64

N

najczęstsze błędy, 19
 najgorszy przypadek, 100
 najmniej znaczący bit, 256
 negocjacje, 71
 niechlujny kod, 20
 niedbałe poprawianie błędów, 20

O

ocenie oferty, 69, 70
 odejmowanie, 271
 odmiany SQL-a, 153
 odpowiadanie na pytania, 52
 behawioralne, 49, 50
 techniczne, 55, 56
 odpowiedzi ustrukturyzowane, 49
 odrzucanie
 oferty, 68
 aplikacji, 21
 podania, 68
 oferta, 67–72
 ogon listy powiązanej, 448
 ograniczenia pamięci, 117, 339
 operator %, 403
 opieranie się na przykładach, 58
 optymalizacja, 344
 organizacja pracy, 129
 oś czasu, 14, 15

P

pakiet finansowy, 69
 pamięć podręczna, 352

permutacja, 180
 pętla, 199, 200
 pętla while, 444
 pisanie
 CV, 43, 44
 kodu, 57
 pseudokodu, 56
 plan działań, 72
 pobieranie bitu, 96
 początkujące firmy, 37
 poczucie szczęścia, 70
 podejście SDE, 50
 pośpiech, 20
 potęgi dwójki, 53
 powtórne wykorzystanie kodu, 63
 pozwolenie na pracę, 37
 praktyczność, 130
 prawdopodobieństwo
 A i B, 106
 A lub B, 106
 proces
 aplikowania, 37
 oceny, 16
 programowanie dynamiczne, 114, 315, 453, 474
 projektowanie
 algorytmu, 56
 bazy danych, 156, 157
 obiektowe, 109, 283
 przechodzenie po drzewie, 231
 przeciążanie
 metod, 149
 operatorów, 143
 przesłanianie metod, 149
 przeszukiwanie
 w głąb, 91
 wszerz, 91
 przygotowania
 do pytań behawioralnych, 46–48
 techniczne, 52–54
 przyjęcie podania, 68
 przypadek bazowy, 59
 pytania
 behawioralne, 19, 45
 techniczne, 51, 55–61

R

rachunek prawdopodobieństwa, 103, 267
referencje, 143
rekrutacja
 w Amazonie, 26
 w Apple'u, 28
 w Facebooku, 29
 w Google'u, 27
 w Microsoftzie, 25
 w Yahoo!, 30
rekruter, 16
rekurencja, 82, 203, 315, 321, 457
 dół – góra, 113
 góra – dół, 113
rezygnowanie, 20
rozmowa próbna, 19
rozwiązania, 175
 bazy danych, 405
 blokady, 413
 drzewa, 225
 grafy, 225
 język Java, 397
 języki C i C++, 383
 kolejki, 207
 listy powiązane, 189
 łamigłówki, 261
 łańcuchy znaków, 177
 manipulowanie bitami, 245
 matematyka, 267
 ograniczenia pamięci, 339
 programowanie dynamiczne, 315
 projektowanie obiektowe, 283
 rachunek prawdopodobieństwa, 267
 rekurencja, 315
 skalowalność, 339
 sortowanie, 357
 stosy, 207
 tablice, 177
 testy, 375
 wątki, 413
 wyszukiwanie, 357
 zadań trudnych, 455
 zadań umiarkowanie trudnych, 425
rozwiązywanie
 problemów na głos, 19
 zadań technicznych, 55
rozwój kariery, 69
równoważenie drzew, 90

S

SDET, Software Development Engineer in Test, 33
sieć kontaktów, 41
sito Eratostenesa, 104
skalowalność, 117, 339
słowo kluczowe
 final, 147, 398
 finalize, 398
 finally, 148, 398
sortowanie, 123, 357, 463
 kubelkowe, 126
 łańcuchów znaków, 180
 przez scalanie, 124
 przez wybieranie, 123
 szybkie, 125
 zewnętrzne, 362
SQL, 153
stabilność firmy, 70
sterta, 463
stos, 85, 207, 220
stosowanie
 bufora, 190
 struktur danych, 62
struktura
 typu FIFO, 86
 typu LIFO, 85
struktury danych, 52, 62, 75, 447
 rozwiązania, 177, 189, 207, 225
synchronizacja, 161
synchronizowane
 bloki kodu, 163
 metody, 162
szablon, 144, 399

T

tablica, 177
 dynamicznie rozszerzana, 78
 z haszowaniem, 77, 349, 385
technika
 biegacza, 82
 pre-order, 91
techniki algorytmiczne, 101
termin przyjęcia oferty, 68
testerzy, 33

- testowanie, 57
 - codziennych obiektów, 130
 - funkcji, 132
 - oprogramowania, 131
- testy, 129, 375
 - automatyczne, 381
 - drugiego typu, 378
 - pierwszego typu, 377
 - ręczne, 381
- tworzenie
 - algorytmów, 58–61
 - klasy pochodnej, 160, 161
 - listy powiązanej, 81
 - nakładki, 192
- typ
 - ArrayList, 78, 150
 - HashMap, 150
 - LinkedList, 150
 - StringBuffer, 78
 - Vector, 150

U

- ubiór, 18
- uczciwość, 21
- uczenie się rozwiązań, 19
- uogólnianie, 59, 243, 369
- upraszczanie, 59, 242, 343, 369
- ustalanie operacji, 110
- ustawianie bitu, 96, 97
- usuwanie węzła, 81

W

- wartości domyślne, 142
- wartość
 - null, 240
 - typu void**, 394
- wątki, 413
- wątki w Javie, 159
- wektor bitowy, 347
- węzeł, 200, 234, 238, 389
- wiedza, 129, 137, 383, 397, 405, 413
- wieloznaczność, 109
- wizy, 37

- wskaźnik, 143, 199
- wskaźnik z modyfikatorem, 388
- wstępne przetwarzanie, 472
- wybór pytań, 13
- wydajność, 471–476
- wyjątkowe sytuacje, 31
- wykrywanie błędów, 66
- wymagania, 302
- wymagania systemowe, 352
- wyszukiwanie, 123, 357
 - binarne, 365
 - dokumentów, 119
 - ścieżki, 317
- wyznaczenie
 - następnej wartości, 252
 - poprzedniej wartości, 253
- wzajemne wykluczanie się, 107
- wzorce projektowe, 110
- wzór na liczby Catalana, 337

Z

- zadania
 - trudne, 173, 455
 - umiarkowane trudne, 169, 425
- zadania dodatkowe
 - rozwiązania, 425, 455
- zadania oparte na wiedzy
 - rozwiązania, 383, 397, 405, 413
- zadawanie pytań, 55
- zagadnienia, 52, 53
 - rozwiązania, 245, 261, 267, 283, 315, 339, 357, 375
- zakleszczenie, 164, 418
- zapisywanie danych, 351
- zapytanie UPDATE, 407
- zasady dotyczące ubioru, 18
- zdarzenia niezależne, 107
- zdobywanie odpowiedniego doświadczenia, 40
- zerowanie bitu, 97
- zestaw kolekcji, 150
- złączenia, 407, 408
- zmienne z modyfikatorem, 388

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

TWOJA PRZEPUSTKA DO NOWEJ FIRMY!

stresuje każdego, kto stara się o pracę. W przypadku naboru na stanowisko programisty składa się ona z kilku etapów, a w jednym z nich poruszane są tematy techniczne. Daje to rekruterom duże pole do popisu. Szczegółowe pytania o znane technologie, sprawdzian z wiedzy ogólnej, łamigłówki, rekurencja — to tylko część zagadnień, o które możesz zostać zapytany. Pytania z zakresu rachunku prawdopodobieństwa czy struktur danych również nie powinny Cię zdziwić. Co możesz zrobić, by dobrze wypaść? Solidnie przygotuj się do rozmowy i nie daj się zaskoczyć!

Mnóstwo cennych wskazówek znajdziesz w tej fantastycznej książce. Sięgnij po nią i poznaj typowy proces rekrutacji, dziesięć najczęściej popełnianych błędów oraz ulubione pytania rekruterów. W kolejnych rozdziałach znajdziesz opisy rekrutacji w takich firmach jak Google, Microsoft, Yahoo! czy Facebook. Te opisy wraz z najlepszymi sposobami na przygotowanie się do rozmowy rekrutacyjnej oraz przykładowymi zadaniami technicznymi (z rozwiązaniami) pomogą Ci opanować stres i zrozumieć tok myślenia osób rekrutujących. Książka ta jest obowiązkową lekturą dla wszystkich programistów szukających pracy!

Dzięki tej książce:

- ▶ poznasz dogłębnie proces rekrutacji,
- ▶ zobaczysz, jak prowadzą ją giganci rynku IT,
- ▶ solidnie przygotujesz się do rozmowy rekrutacyjnej,
- ▶ unikniesz typowych błędów,
- ▶ zdobędziesz pracę w wybranej firmie!.

helion.pl
księgarnia
internetowa

Nr katalogowy: 23566

Księgarnia internetowa:
<http://helion.pl>

Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

Sprawdź najnowsze promocje:
• <http://helion.pl/promocje>
Książki najchętniej czytane:
• <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
• <http://helion.pl/nowosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-9332-0



9 788324 693320

cena: 79,00 zł

Informatyka w najlepszym wydaniu