

KARIERA PROGRAMISTY

**JAK BUDOWAĆ
DOŚWIADCZENIE,
PRZEJŚĆ REKRUTACJĘ
I ZDOBYĆ PRACĘ
MARZEŃ**

JOHN SONMEZ

AUTOR BESTSELLERA

Sprawny programista. Pracuj, zarabiaj i zdobywaj kwalifikacje

Helion 



Tytuł oryginału: The Complete Software Developer's Career Guide:
How to Learn Programming Languages Quickly, Ace Your Programming Interview, and Land Your
Software Developer Dream Job

Tłumaczenie: Wojciech Usarzewicz
ISBN: 978-83-283-4393-1

© 2017 by Simple Programmer. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Polish edition copyright © 2018 by Helion SA
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 biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą 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
WW: <http://helion.pl> (księgarnia internetowa, katalog książek)

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

Printed in Poland.

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

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

SPIS TREŚCI

| | |
|---|-----------|
| Słowem wstępu | 21 |
| Czy ta książka jest dla mnie? | 25 |
| Osoby początkujące lub po prostu zainteresowane nauką programowania | 26 |
| Osoby w połowie kariery | 26 |
| Doświadczeni zawodowcy | 27 |
| Rozdział 1. Jak korzystać z tej książki? | 29 |
| Dlaczego napisałem tę książkę? | 29 |
| Cele tej książki | 30 |
| Jak korzystać z tej książki? | 31 |
| Powtarzanie i działanie | 33 |
| Część I. Jak zacząć karierę programisty? | 35 |
| Rozdział 2. Jak zacząć? | 37 |
| Moje własne początki | 37 |
| Poznajemy zawód | 39 |
| Zrozumienie problemu | 40 |
| Projekt | 40 |
| Pisanie kodu | 41 |
| Testowanie i wdrażanie | 41 |
| Pisanie kodu to coś więcej niż pisanie kodu | 42 |
| Miej plan | 43 |
| Określanie planu | 43 |
| Tworzenie planu | 44 |
| Kto chce być „sportowcem”? | 45 |
| Solidny przykład | 47 |

| | |
|--|-----------|
| Rozdział 3. Umiejętności praktyczne, których potrzebujesz | 49 |
| Umiejętności, którymi zapłacisz rachunki | 49 |
| Jeden język programowania | 50 |
| Opanowanie struktur kodu | 51 |
| Programowanie zorientowane obiektowo | 52 |
| Algorytmy oraz struktury danych | 52 |
| Platforma i powiązane technologie | 54 |
| Framework i narzędzia | 55 |
| Podstawy baz danych | 57 |
| Kontrola wersji | 57 |
| Kompilacja i wdrożenie | 58 |
| Testowanie | 59 |
| Debugowanie | 60 |
| Metodyki tworzenia oprogramowania | 61 |
| Przytłoczony? Niepotrzebnie | 61 |
| Rozdział 4. Jak rozwijać umiejętności praktyczne? | 64 |
| Jak nauczyć się szybko uczyć? | 65 |
| Podstawy procesu | 65 |
| Nauka przez działanie | 67 |
| Jak uczyć się przez działanie? | 68 |
| Przykład uczenia się przez działanie | 69 |
| Jak nauczam umiejętności praktycznych? | 70 |
| Szersze spojrzenie: co możesz zrobić z daną technologią? | 70 |
| Jak zacząć? | 71 |
| 20% tego, co naprawdę musisz wiedzieć, aby działać skutecznie | 72 |
| Czytaj to, co piszą eksperci | 73 |
| Praktyka, praktyka, praktyka | 74 |
| Rozdział 5. Jaki język programowania wybrać? | 75 |
| Język wcale nie ma aż takiego znaczenia | 75 |
| O czym warto pamiętać, wybierając język programowania? | 77 |
| Potencjał pracy i przyszłość | 77 |
| Technologia, która Cię interesuje | 80 |
| Poziom trudności | 81 |
| Dostępne materiały | 81 |
| Zdolność do adaptacji | 82 |
| Kilka słów na koniec | 83 |

| | |
|--|------------|
| Rozdział 6. Uczymy się pierwszego języka programowania | 85 |
| Zacznij od przyjrzenia się działającym zastosowaniom | 86 |
| Znajdź kilka dobrych materiałów lub książek i je przeskanuj | 87 |
| Naucz się pisać „Witaj, świecie” | 88 |
| Poznaj podstawowe elementy składni i przetestuj je na prawdziwych problemach | 89 |
| Poznaj różnicę między elementami języka a bibliotekami | 90 |
| Analiza istniejącego kodu w celu jego zrozumienia | 91 |
| Zacznij tworzyć | 93 |
| Zastosuj język programowania do konkretnej platformy czy technologii | 94 |
| Rozwiązuaj problemy za pomocą algorytmów i wybranego języka | 95 |
| Rozdział 7. Idziemy na studia | 98 |
| Zalety | 99 |
| Wady | 102 |
| Strategia | 106 |
| Rozdział 8. Kursy | 111 |
| O co chodzi z tymi kursami? | 111 |
| Zalety | 112 |
| Wady | 117 |
| Strategia | 120 |
| Rozdział 9. Samodzielna nauka | 124 |
| Zalety samodzielnej nauki programowania | 125 |
| Wady samodzielnej nauki programowania | 128 |
| Strategia | 132 |
| Część II. Jak znaleźć pracę? | 137 |
| Rozdział 10. Staż | 139 |
| Czym jest staż? | 140 |
| Czy powinienem żądać pensji? | 142 |
| Jak dostać staż? | 143 |
| Co czyni z Ciebie dobrego stażystę? | 146 |
| Ze stażu na etat | 147 |
| Rozdział 11. Jak dostać pracę bez doświadczenia? | 149 |
| Największe ryzyko dla firm zatrudniających programistę | 149 |
| Ominięcie problemu | 152 |
| Bądź obecny w sieci | 153 |
| Zbuduj portfolio | 154 |

| | |
|--|------------|
| Stwórz własną firmę | 156 |
| Przygotowanie do rozmów kwalifikacyjnych | 157 |
| Zbuduj sieć kontaktów | 158 |
| Zaoferuj pracę za darmo | 158 |
| Zaoferuj pracę nad małym projektem | 159 |
| Po prostu dostań się do firmy | 160 |
| Zdobądź certyfikaty | 161 |
| Bądź wytrwały | 162 |
| Rozdział 12. Jak znaleźć pracę? | 163 |
| Podejście tradycyjne, czyli brak oryginalności | 164 |
| Loteria | 164 |
| Stwórz dobre CV | 166 |
| Określ liczbę podań składanych każdego dnia | 167 |
| Jak składać podanie o pracę? | 168 |
| Mierz rezultaty | 169 |
| Zmiany w planie | 170 |
| Pracuj bezpośrednio z rekruterem | 171 |
| Odrobina kreatywnego myślenia | 172 |
| Zbuduj sieć kontaktów | 173 |
| Trafnie dobieraj firmy | 175 |
| Stwórz coś przydatnego | 176 |
| Zacznij od góry | 177 |
| Wykorzystaj marketing przychodzący | 178 |
| Porozmawiaj z rekruterem | 179 |
| Bądź wytrwały | 180 |
| Rozdział 13. Tworzymy CV | 181 |
| Krok pierwszy: nie twórz CV | 182 |
| Wybieramy specjalistę od CV | 184 |
| Praca ze specjalistą | 185 |
| Co czyni CV dobrym? | 187 |
| Pisanie swojego CV | 189 |
| Zacznij od LinkedIn | 189 |
| Skup się na oferowanej wartości | 190 |
| Co zrobiłeś, jak to zrobiłeś i co to dało? | 190 |
| Krótko i konkretnie | 192 |
| Sprawdź swój język | 194 |
| Stwórz kilka wersji CV | 194 |
| Uczyń CV wyjątkowym | 195 |

| | |
|---|------------|
| Rozdział 14. Rozmowa kwalifikacyjna | 197 |
| Rodzaje rozmów kwalifikacyjnych | 199 |
| Rozmowa telefoniczna | 199 |
| Rozmowy techniczne online | 200 |
| Standardowa rozmowa techniczna | 201 |
| Dopasowanie do kultury organizacyjnej | 202 |
| Komisja kwalifikacyjna | 202 |
| Rozmowa praktyczna | 203 |
| Rozmowy całodniowe | 203 |
| Co musisz wiedzieć? | 204 |
| Jak rozwiązać problemy programistyczne? | 205 |
| Często spotykane pytania techniczne | 206 |
| Pytania osobiste i psychologiczne | 207 |
| Porady dotyczące rozmowy | 208 |
| Właściwy ubiór | 210 |
| Bądź na czas | 211 |
| Nie kłam | 211 |
| Nie broń się | 212 |
| Rozwijaj temat | 213 |
| Bądź pewny siebie (i nie udawaj) | 213 |
| Zademonstruj ten jeden, najważniejszy przekaz | 214 |
| Ćwicz, ćwicz, ćwicz | 215 |
| Rozdział 15. Płaca i negocjacje | 216 |
| Poznaj swoje możliwości | 217 |
| Otrzymanie oferty | 218 |
| Negocjowanie | 223 |
| Pozycja do negocjacji | 225 |
| Kto pierwszy rzuci liczbę, przegrywa | 227 |
| Nie bój się złożyć kontroferty | 228 |
| Wszystko można negocjować | 230 |
| Nie poddawaj się presji czasu | 231 |
| Wiele ofert | 231 |
| Rozdział 16. Jak odejść z pracy? | 234 |
| Kiedy odejść z pracy? | 234 |
| Jak odejść z pracy? | 236 |
| Nie martw się o „zespół” | 237 |
| Pamiętaj o informacji z wyprzedzeniem | 238 |
| Nie groź odejściem z pracy | 240 |

| | |
|--|------------|
| Nie przesadzaj z okresem wypowiedzenia | 241 |
| Świat jest niesamowicie mały | 242 |
| Przeszkol swojego następcę | 243 |
| Nie mów nic złego na ostatniej rozmowie | 244 |
| Rozdział 17. Zmiana kariery (późne wejście) | 246 |
| Korzyści płynące ze zmiany kariery | 246 |
| Wady i minusy | 248 |
| Jak to zrobić? | 249 |
| Dokonaj przejścia w aktualnej pracy | 249 |
| Szukaj sposobów na wykorzystanie swojego istniejącego doświadczenia | 250 |
| Bądź gotów zacząć na dole | 251 |
| Rozdział 18. Przejście z QA lub podobnej roli do działu programistycznego | 253 |
| Największa przeszkoda | 255 |
| Poinformuj o tym, czego chcesz | 255 |
| Poproś o szansę | 257 |
| Sam szukaj okazji | 257 |
| Wykorzystaj czas własny | 258 |
| Szukaj pracy przejściowej | 259 |
| Przejście do nowej firmy | 260 |
| Moja ostatnia rada | 261 |
| Rozdział 19. Zlecenie kontra etat | 262 |
| Rodzaje zleceń | 263 |
| Zlecenia przez agencję | 264 |
| Niezależni zleceniobiorcy | 265 |
| Wolny strzelec | 266 |
| Praca etatowa | 266 |
| Pieniądze | 267 |
| Analizujemy stawkę przy pracy na zlecenie | 267 |
| Dlaczego zleceniobiorca zarabia więcej? | 269 |
| Wykorzystaj praktyczną wartość dodatków | 270 |
| Otoczenie w pracy | 271 |
| Inne kwestie do przemyślenia | 272 |
| Rozdział 20. Jak działa branża rekrutacyjna? | 274 |
| Typy rekruterów i agencji rekrutacyjnych (i jak się im płaci) | 275 |
| Mali i niezależni rekruterzy | 276 |
| Duże agencje | 277 |

| | |
|--|-----|
| Agencje wewnętrzne | 278 |
| Wewnętrzni rekruterzy | 280 |
| Agenci rekrutacyjni | 280 |
| Co to oznacza dla Ciebie? | 281 |
| Pierwsze podanie | 282 |
| Negocjacje pensji | 283 |
| Korzystanie z usług rekrutera a samodzielne szukanie pracy | 286 |

Część III. Co musisz wiedzieć o tworzeniu oprogramowania? 289

Rozdział 21. Ogólny przegląd języków programowania 291

| | |
|--------------------|-----|
| C | 292 |
| C++ | 292 |
| C# | 293 |
| Java | 294 |
| Python | 294 |
| Ruby | 295 |
| JavaScript | 295 |
| Perl | 296 |
| PHP | 297 |
| Objective-C | 298 |
| Swift | 298 |
| Go | 299 |
| Erlang | 299 |
| Haskell | 300 |
| Dla jasności | 300 |

Rozdział 22. Co to takiego programowanie webowe? 303

| | |
|---|-----|
| Szybki przegląd | 304 |
| Jak działa sieć? | 305 |
| Krótką historia internetu | 307 |
| Główne technologie wykorzystywane w programowaniu webowym | 309 |
| HTML | 309 |
| CSS | 310 |
| JavaScript | 312 |
| Renderowanie po stronie serwera | 313 |
| Renderowanie po stronie klienta | 313 |
| API | 315 |
| Absolutne podstawy | 316 |

| | |
|---|------------|
| Rozdział 23. Programowanie mobilne | 317 |
| Co to takiego programowanie mobilne? | 318 |
| Główne platformy mobilne | 319 |
| iOS | 319 |
| Android | 320 |
| Inne platformy | 321 |
| Jak wygląda programowanie mobilne? | 321 |
| Narzędzia natywne | 322 |
| Frameworki i narzędzia wieloplatformowe | 323 |
| Mobilne aplikacje webowe | 326 |
| Pomyśl o programowaniu mobilnym | 327 |
| Rozdział 24. Programowanie back-endu | 328 |
| Co to jest programowanie back-endu? | 329 |
| Co robią programiści back-endu? | 329 |
| Główne technologie i umiejętności w programowaniu back-endu | 330 |
| Programiści full-stack? | 331 |
| Podsumowanie | 332 |
| Rozdział 25. Kariera w branży gier wideo | 333 |
| Ostrzeżenie | 334 |
| Dyplom | 335 |
| Wymagane umiejętności | 336 |
| Praca dla dużego studia | 338 |
| Twórca niezależny | 339 |
| Zasoby i sugestie | 340 |
| Rozdział 26. DevOps i administratorzy baz danych | 342 |
| DBA, czyli administrator baz danych | 343 |
| Bazy danych wymagają opieki | 344 |
| Czy muszę być DBA? | 344 |
| DevOps — nowa rola | 346 |
| Eksploatacja — jak to robiono dawniej? | 346 |
| Co to jest DevOps? | 347 |
| Co to oznacza dla Ciebie? | 348 |
| Rozdział 27. Metodyka programowania | 350 |
| Tradycyjny model kaskadowy | 351 |
| Czy pilnujesz SDLC? | 352 |
| Agile | 355 |
| Manifest Agile | 355 |

| | |
|---|------------|
| Agile to nie do końca metodyka | 357 |
| Problemy z modelem kaskadowym | 357 |
| Scrum | 358 |
| Role w Scrum | 359 |
| Jak działa Scrum? | 359 |
| Problemy ze Scrumem | 360 |
| Kanban | 361 |
| Programowanie ekstremalne (XP) | 363 |
| Inne metodyki i modele programowania | 365 |
| Rozdział 28. Testowanie i podstawy QA | 367 |
| Podstawowe założenie testowania | 368 |
| Typowe rodzaje testowania | 369 |
| Proces testowania | 375 |
| Jak działa testowanie w zespołach stosujących Agile? | 377 |
| Testowanie i Ty, programista | 378 |
| Rozdział 29. Programowanie oparte na testach i testy jednostkowe | 380 |
| Czym są testy jednostkowe? | 382 |
| Co jest czasem nazywane testami jednostkowymi? | 384 |
| Wartość testów jednostkowych | 385 |
| Co to takiego programowanie oparte na testach (TDD)? | 386 |
| Co jest celem TDD? | 387 |
| Typowy tok pracy TDD | 389 |
| To tylko podstawy | 390 |
| Rozdział 30. Kontrola wersji | 394 |
| Czym jest kontrola wersji? | 396 |
| Dlaczego to takie ważne? | 396 |
| Podstawy kontroli wersji | 398 |
| Repozytoria | 398 |
| Pobranie kodu | 398 |
| Rewizje | 399 |
| Branching, czyli odgałęzienia | 400 |
| Scalanie | 401 |
| Konflikty | 402 |
| Typowe rozwiązania | 403 |
| Systemy scentralizowane | 403 |
| Systemy rozproszone (DVCS) | 404 |
| Szybki przegląd najpopularniejszych systemów kontroli wersji | 405 |
| CVS | 405 |

| | |
|---|------------|
| Subversion | 406 |
| Git | 406 |
| Mercurial | 407 |
| Coś jeszcze? | 408 |
| Rozdział 31. Ciągła integracja | 409 |
| Jak kiedyś budowaliśmy kod? | 410 |
| I nastąpiła era serwerów kompilacji | 412 |
| Nareszcie, ciągła integracja | 414 |
| Przykład toku pracy z ciągłą integracją | 415 |
| Serwery i oprogramowanie CI | 419 |
| Rozdział 32. Debugowanie | 423 |
| Co to takiego debugowanie? | 424 |
| Pierwsza zasada debugowania: nie używaj programu do debugowania | 425 |
| Odtwórz błąd | 426 |
| Usiądź i pomyśl | 428 |
| Przetestuj hipotezy | 429 |
| Sprawdź swoje założenia | 431 |
| Dziel i rządź | 432 |
| Jeśli naprawisz błąd, zrozumiesz go | 433 |
| Sztuka i nauka | 435 |
| Rozdział 33. Utrzymanie kodu | 436 |
| Większość kariery spędzisz na utrzymywaniu kodu | 437 |
| Wybitni programiści piszą kod, który można utrzymać | 438 |
| Zasada skauta | 439 |
| Najważniejsza jest czytelność | 440 |
| Refaktoryzacja w celu ulepszenia kodu | 441 |
| Automatyzacja jest niezbędna | 443 |
| Pisz dobre komentarze | 443 |
| Materiały dla chcących dowiedzieć się więcej | 444 |
| Rozdział 34. Praca i nazwy stanowisk | 447 |
| Tytułatura nie ma znaczenia... .. | 448 |
| ... ale powinieneś dostać najlepszy tytuł | 448 |
| Często spotykana tytułatura | 449 |
| Jeden tytuł, którego trzeba unikać | 451 |
| Podstawowe role czy stanowiska? | 452 |
| Tytułatura w wielkich firmach technologicznych | 454 |
| Sporo hałasu o tytuł | 455 |

| | |
|---|------------|
| Rozdział 35. Rodzaje pracy | 457 |
| Pisanie kodu | 458 |
| Naprawianie błędów | 459 |
| Projekty i architektura | 459 |
| Spotkania | 460 |
| Uczenie się | 461 |
| Eksperymentowanie i odkrywanie | 462 |
| Testowanie | 462 |
| Myślenie | 463 |
| Interakcja z klientami i interesariuszami | 464 |
| Szkolenie innych, mentoring | 465 |
| I tyle... | 465 |

Część IV. Praca w roli programisty **467**

| | |
|--|------------|
| Rozdział 36. Współpracownicy | 469 |
| Liczy się pierwsze wrażenie | 471 |
| Staraj się być pomocny | 472 |
| Unikaj scen i nieprzyjemnych sytuacji | 473 |
| Ale nie unikaj konfliktów | 474 |
| Polityka i religia | 476 |
| Współpracownicy, którzy nie pracują | 477 |
| Gadatliwi współpracownicy | 480 |
| Osoby toksyczne | 482 |
| A co z X? | 483 |
| Rozdział 37. Jak radzić sobie z szefem? | 484 |
| Zrozumieć szefa | 485 |
| Odpowiedzialność na Twoich barkach | 487 |
| Co ułatwia pracę Twojemu szefowi? | 487 |
| Wroży szefowie | 489 |
| Mikromenedżer | 490 |
| Dręczyciel | 491 |
| Ignorant | 494 |
| Pan niewolników | 497 |
| Nie zawsze możesz wybrać swojego szefa | 499 |

| | |
|--|------------|
| Rozdział 38. Praca z działem QA | 500 |
| QA nie jest wrogiem | 500 |
| Wiedz, co jest testowane | 501 |
| Najpierw sam testuj swój kod | 503 |
| Unikaj cyklu błędów i poprawek | 504 |
| Pomagaj w automatyzacji | 505 |
| Co zrobić z tym jednym typkiem? | 506 |
| | |
| Rozdział 39. Równowaga między życiem a pracą | 508 |
| Równowaga między życiem a pracą to mit | 509 |
| Nadgodziny rzadko coś wnoszą | 511 |
| Ale to nie powód, by nie pracować ciężko | 512 |
| Najpierw płać sobie | 513 |
| Niech dbanie o siebie będzie Twoim priorytetem | 514 |
| Ostrożnie dobieraj relacje | 517 |
| Życie tu i teraz | 518 |
| Prawdziwa równowaga | 519 |
| | |
| Rozdział 40. Praca w zespole | 520 |
| Wspólny sukces, wspólna porażka | 521 |
| Zespoły mają wspólne cele | 522 |
| Bądź odpowiedzialny za zespół | 524 |
| Komunikacja i współpraca | 525 |
| Bądź szczery, ale taktowny | 526 |
| | |
| Rozdział 41. Przekonywanie do swoich pomysłów | 528 |
| Dlaczego przekonywanie do pomysłów jest takie ważne? | 529 |
| Nie kłóć się | 530 |
| Bądź przekonujący | 531 |
| Prowadzenie do celu | 532 |
| Umiejętna komunikacja | 533 |
| Pożycz autorytet | 533 |
| Stwórz autorytet | 534 |
| Edukuj | 536 |
| Dojdź do wprawy | 537 |
| | |
| Rozdział 42. Ubiór | 538 |
| Wygląd ma znaczenie | 539 |
| Ubieraj się lepiej, niż jest to wymagane | 542 |
| Podążaj za liderem | 543 |
| Charyzma i sprzeczności | 544 |

| | |
|--|------------|
| Ubieraj się, by... zmienić osobowość? | 547 |
| Symbole statusu | 547 |
| Fryzura, makijaż i podstawowa higiena | 548 |
| A jeśli mnie to nie interesuje? | 549 |
| Rozdział 43. Ocena pracownicza | 552 |
| Jak dobrze wypadłem w ocenie pracowniczej? | 553 |
| Nie czekaj na ostatnią chwilę | 555 |
| Miej konkretne cele, informuj o nich | 556 |
| Śledź i dokumentuj postępy | 557 |
| Przygotuj argumenty | 558 |
| Odwołuj się, jeśli trzeba | 559 |
| Pułapka oceny samego siebie | 560 |
| Oceny współpracownicze | 561 |
| Technika wymuszonego rozkładu | 563 |
| Rozdział 44. Uprzedzenia | 566 |
| Zaakceptuj, że ludzie nieświadomie są uprzedzeni | 567 |
| Staraj się unikać uprzedzeń | 568 |
| Nie segreguj się | 571 |
| Bądź pewny siebie | 572 |
| Ignoruj, co się da | 574 |
| Zgłoś to, czego nie możesz zignorować | 575 |
| Uprzedzenia są okropne | 577 |
| Rozdział 45. Lider | 578 |
| Czym jest przywództwo? | 579 |
| Jak skutecznie przewodzić? | 580 |
| Przywództwo we wszystkich obszarach | 581 |
| Wymagaj od siebie więcej | 582 |
| Jesteś odpowiedzialny za swój zespół | 583 |
| Ufaj swoim ludziom, deleguj pracę | 584 |
| Prowadź! | 586 |
| Rozdział 46. Podwyżki i awanse | 587 |
| Zawsze wybieraj odpowiedzialność, nie wypłatę | 588 |
| Podejmij inicjatywę | 590 |
| Zainwestuj w edukację | 591 |
| Nie ukrywaj swoich celów | 592 |
| Zyskaj wartość poza firmą | 594 |
| Zostań cennym atutem | 595 |

| | |
|---|-----|
| Poproś o konkretne sumy | 597 |
| Bez gróźb | 598 |
| Nie mów, dlaczego potrzebujesz pieniędzy | 599 |
| Jeśli nic innego nie pomoże, poszukaj innej pracy | 600 |

Rozdział 47. Kobiety w świecie technologii 602

| | |
|---|-----|
| Stereotypy i uprzedzenia | 603 |
| Dlaczego faceci nękają kobiety? | 604 |
| Rady dla kobiet | 606 |
| Staraj się nie być taka urażona | 607 |
| Nie ignoruj prawdziwych problemów | 608 |
| Nie staraj się być jedną z nich | 609 |
| Wykorzystaj swoje zalety | 610 |
| Negocjuj | 613 |
| Rady dla mężczyzn | 615 |
| Nie traktuj kobiet protekcjonalnie | 616 |
| Kobiety nie są facetami | 619 |
| Nie wyładowuj frustracji na kobietach | 620 |
| Zachowuj się normalnie | 621 |
| Szczerze wierzę, że to pomoże | 622 |

Część V. Rozwój kariery 623

Rozdział 48. Budowanie reputacji 625

| | |
|-----------------------------------|-----|
| Korzyści płynące ze „sławy” | 627 |
| Styl i treść | 628 |
| Tworzenie marki osobistej | 630 |
| Jak zostać „znanym”? | 633 |
| Twórz wartość dla innych | 635 |
| Bądź cierpliwy | 636 |

Rozdział 49. Networking 638

| | |
|----------------------------------|-----|
| Błędny networking | 639 |
| Poprawny networking | 640 |
| Gdzie nawiązywać kontakty? | 642 |
| Tworzenie grup | 645 |
| Networking to nic trudnego | 646 |

| | |
|--|------------|
| Rozdział 50. Dbanie o swoje umiejętności | 648 |
| Brak planu to też plan, po prostu okropny | 648 |
| Czytanie blogów | 650 |
| Czytanie książek | 650 |
| Wybieranie nowej rzeczy do opanowania | 652 |
| Ucz się szybko | 654 |
| Uczestniczenie w wydarzeniach | 655 |
| Czytanie newsów | 655 |
| Koduj — dużo | 656 |
| Unikaj strefy komfortu | 657 |
| Rozdział 51. Umiejętności ogólne kontra specjalizacja | 658 |
| Moc specjalizacji | 659 |
| Aby się specjalizować, musisz mieć szerokie podstawy wiedzy | 660 |
| Pamiętaj o zasadzie T | 661 |
| Ale wszyscy mówią, by szukać ludzi o wiedzy ogólnej | 662 |
| Dziś nawet nie można znać się na wszystkim | 663 |
| A jeśli wybiorę złą specjalizację? | 664 |
| Co powinieneś zrobić? | 665 |
| Rozdział 52. Prelekcje i konferencje | 667 |
| Uczestniczenie w konferencjach | 667 |
| Ale konferencje są drogie | 668 |
| Co robić na konferencji? | 669 |
| Występy | 671 |
| Jak zacząć? | 673 |
| Pokonanie tremy | 675 |
| Kilka praktycznych rad | 677 |
| Przygotowanie materiałów | 678 |
| Występy płatne | 679 |
| Bierz się do pracy | 682 |
| Rozdział 53. Tworzenie bloga | 683 |
| Dlaczego blog to wciąż najlepszy pomysł? | 684 |
| Jak stworzyć bloga? | 686 |
| Wybieranie tematu | 688 |
| Jak blogować? | 689 |
| Potęga konsekwencji | 692 |
| Pozyskiwanie ruchu | 693 |
| Znajdź swój styl | 696 |
| Pisz | 699 |

| | |
|--|------------|
| Rozdział 54. Freelancing i własna firma | 701 |
| Czy naprawdę chcesz to zrobić? | 702 |
| Czym jest freelancing? | 704 |
| Jak zacząć? | 705 |
| Wspomniałeś coś o prostszym sposobie? | 707 |
| Jak ustalić stawkę? | 708 |
| Otwieranie firmy | 712 |
| Nie dąż do perfekcji | 713 |
| Nie rzucaj pracy | 714 |
| Najpierw zbuduj bazę odbiorców | 716 |
| Naucz się sprzedawać | 718 |
| Znajdź pomoc | 719 |
| Biznes jest trudny, ale jest też tego wart | 720 |
| Rozdział 55. Ścieżki kariery | 721 |
| Trzy typy programistów | 722 |
| Możliwości dla tradycyjnej kariery | 723 |
| Programista webowy | 724 |
| Programista mobilny | 724 |
| Programista desktopowy | 725 |
| Gry wideo | 725 |
| Systemy wbudowane | 726 |
| Data science | 727 |
| Narzędzia i oprogramowanie dla firm | 727 |
| Oprogramowanie w chmurze | 728 |
| Automatyzacja | 728 |
| Pokonanie ograniczeń | 729 |
| Zarządzanie czy ścieżka techniczna? | 729 |
| Zawsze myśl o tym, dokąd zmierzasz | 731 |
| Rozdział 56. Gwarancja zatrudnienia i poczucie stabilizacji | 733 |
| Nie ma stabilizacji i to jest w porządku | 734 |
| Bezpieczeństwa zatrudnienia nie zyskasz, skrywając wiedzę | 736 |
| Rób odwrotnie | 737 |
| Zastąp stabilizację umiejętnościami | 738 |
| Stwórz własną sieć bezpieczeństwa | 738 |
| Nie bój się niepewności | 742 |

| | |
|---|------------|
| Rozdział 57. Szkolenia i certyfikaty | 745 |
| Czy certyfikaty są warte swojej ceny? | 746 |
| Dlaczego więc wyrobiłeś certyfikaty, Johnny? | 747 |
| Jak uzyskać certyfikat? | 748 |
| A co ze szkoleniami? | 749 |
| Jakiego rodzaju szkolenia są dostępne? | 751 |
| W pełni wykorzystaj szkolenie | 753 |
| Przekonanie pracodawcy do sfinansowania szkolenia | 755 |
| Zostanie szkoleniowcem | 757 |
| Wszystko zależy od tego, co dasz od siebie | 758 |
| Rozdział 58. Projekty po godzinach | 759 |
| Zawsze powinieneś robić coś po godzinach | 762 |
| Wybieramy projekt dodatkowy | 763 |
| Projekt dodatkowy powinien realizować dwa cele | 765 |
| Zaczynamy | 767 |
| Działaj konsekwentnie | 769 |
| Kończ, co zacząłeś | 770 |
| Zarabianie na projektach dodatkowych | 772 |
| Zacznij działać! | 773 |
| Rozdział 59. Książki, które warto przeczytać | 774 |
| Pisanie dobrego kodu | 775 |
| Co musisz wiedzieć? | 776 |
| Praca z zastanym kodem | 777 |
| Rozwój zawodowy | 779 |
| Rozwój osobisty | 780 |
| Zgłębianie tematów | 781 |
| Rozrywka i zabawa | 783 |
| Wyrwałość i motywacja | 785 |
| Czytaj dalej, przyjacielu... .. | 786 |
| Rozdział 60. Na pożegnanie | 788 |
| Prośba na koniec | 791 |
| Przypisy końcowe | 793 |

ROZDZIAŁ 2.

JAK ZACZAĆ?

Kiedy dopiero zaczynałem przygodę z programowaniem, nie miałem pojęcia, co w ogóle robię.

Byłem też sfrustrowany. **Nic nie miało sensu** i myślałem, że nigdy tego nie „załapię”.

Mówię to dlatego, że pewnie czujesz się tak samo, skoro postanowiłeś przeczytać tę książkę.

Nie martw się, **to normalne. To wręcz całkowicie naturalne.**

Pozwól, że od razu wyjaśnię pewną sprawę: by zostać programistą, **nie musisz być geniuszem**, nie musisz nawet mieć **ponadprzeciętnej inteligencji**.

Jeśli dopiero zaczynasz działanie w dziedzinie programowania i nie odczuwasz przy tym przytłoczenia⁶, nie czujesz się, jakbyś właśnie wskoczył do wody z nogami zatopionymi w betonie, to albo robisz coś nie tak, albo nie jesteś człowiekiem — albo jedno i drugie.

Tak czy inaczej, powinieneś się spodziewać, że odczujesz trudy i zagubienie, kiedy dopiero się uczysz, ale nie będzie to trwało wiecznie — obiecuję.

MOJE WŁASNE POCZĄTKI

Pamiętam, kiedy uczyłem się programować. Nie miałem dostępu do tych wszystkich materiałów, które są dostępne dziś. Tak naprawdę, **w ogóle nie miałem materiałów**.

Ściągnąłem kod źródłowy popularnego **MUD**-a (ang. *Multi-User Dungeon*). To taki *World of Warcraft*, ale bez grafiki — sam tekst. Tak, to dawne czasy, kiedy do internetu się jeszcze „dzwoniło”.

Nie wiedziałem nawet, czego szukam. Wiedziałem tylko, że chcę stworzyć swoją wersję MUD-a i dodać własne elementy, a to wymagało znalezienia igły w stogu niezrozumiałych linii kodu.

Zacząłem się bawić. Wprowadzałem różne wartości zmiennych. Szukałem jakiegoś fragmentu kodu, który zdawał się kontrolować szanse zadania krytycznego ciosu wymierzonego w przeciwnika. Zmieniałem kod, kompilowałem i sprawdzałem efekty tych działań.

Czasami się udawało. **Czasami programu nie udawało się nawet skompilować.** Uczyłem się, patrząc na to, co działało, a co nie.

Wciąż nie wiedziałem, co robię, ale mniej więcej po tygodniu grzebania w kodzie udało mi się stworzyć wersję MUD-a z kilkoma elementami, które chciałem w niej umieścić.

Daleko było mi do dobrego programisty, ale był to jakiś początek — wszyscy musimy jakoś zacząć.

Opowiadam Ci moją historię, bo **uważam, że to, jak ja zaczynałem, to najlepszy sposób, by zacząć programować. Ważniejszy niż przeczytanie podręcznika, pójdzie na studia czy zapisanie się na kurs.**

Musisz zacząć grzebać w kodzie i patrzeć, co działa, a co nie (uważam po prostu, że to najlepszy sposób nauki). Zobacz odpowiednią sekcję na temat efektywnego uczenia się w *Sprawnym programiście*.

Uczenie się kodowania i uczenie się tego, jak zacząć w świecie programowania, to dwie zupełnie osobne sprawy.

Tak, musisz się nauczyć kodowania, ale nauka programowania to coś więcej niż tylko kodowanie. Rozdział ten omawia właśnie to „coś więcej”.

POZNAJEMY ZAWÓD

Przede wszystkim musisz nauczyć się czegoś o tworzeniu oprogramowania.

Jest to zarówno łatwiejsze, jak i trudniejsze, niż mógłbyś myśleć.

Spora część tej książki poświęcona jest temu, co musisz wiedzieć o tworzeniu oprogramowania, ale już teraz w skrócie to omówię.

Tworzenie oprogramowania to nie tylko programowanie. Programowanie to spora część tego procesu, ale sama umiejętność kodowania daleko Cię nie zaprowadzi, zwłaszcza kiedy interesuje Cię kariera w tej branży.

Ideą przyświecającą większości projektów związanych z oprogramowaniem jest **automatyzacja pewnych manualnych procesów** albo stworzenie nowego zautomatyzowanego procesu robienia czegoś, co jest zbyt trudne, by robić to ręcznie.

Pomyśl o edytorze tekstu, którego teraz używam, pisząc tę książkę. Wpisuję kolejne słowa, korzystając z Google Docs.

Bez tego czy innego edytora tekstu musiałbym pisać tę książkę na maszynie do pisania albo wręcz robić to za pomocą pióra.

Gdybym chciał sformatować dokument do druku, musiałbym ręcznie ułożyć czcionki w prasie drukarskiej.

Gdybym chciał poprawić błędy — zwłaszcza ortograficzne — musiałbym mieć pod ręką całą butlę korektora (a obok zapewne też sporą butelkę szkockiej).

Ale Google Docs oferuje wszystkie te rzeczy automatycznie. Mam do dyspozycji wiele programów, dzięki którym mogę zautomatyzować proces pisania książki. Myślę, że wiesz już, co mam na myśli.

Pozwól, że podkreślę kluczową koncepcję, której powinieneś się nauczyć tak szybko, jak to tylko możliwe, wkraczając w świat programowania.

Musisz wiedzieć, jak zrobić coś ręcznie, zanim zaczniesz to automatyzować.

ZROZUMIENIE PROBLEMU

Zbyt wielu początkujących — lub doświadczonych — programistów próbuje pisać programy komputerowe, nie mając pojęcia, co one tak naprawdę mają zrobić. Programiści chcą po prostu zabrać się do pracy, zacząć kodować (to odpowiednie podejście, jeśli chcesz się nauczyć programowania — jak opisałem to w przypadku MUD-a — ale nie jest dobre, kiedy naprawdę musisz stworzyć konkretny program).

To oczywiście, że **jesteś mądrzejszy**, bo przecież czytasz tę książkę.

Proces tworzenia oprogramowania zawsze zaczyna się od zrozumienia problemu, który chcemy rozwiązać. Co tak naprawdę automatyzujesz?

Różne metodyki rozwiązują tę kwestię w różny sposób, ale to akurat nie jest w tej chwili istotne. Teraz ważne jest to, że musisz w ten czy inny sposób zgromadzić jakieś wymogi co do programu i zrozumieć problem, który chcesz rozwiązać, zanim w ogóle zaczniesz pisać pierwszą linijkę kodu.

Może to być bardzo proste — możesz na przykład porozmawiać z potencjalnym klientem i omówić, czego oczekuje i jak to coś powinno działać. Albo możesz obrać bardziej formalny kierunek i stworzyć całą udokumentowaną specyfikację projektu.

PROJEKT

Kiedy odpowiednio rozumiesz już założenia programu, który trzeba stworzyć, zaczynasz pracę nad jakimś rodzaju projektem, określającym, w jaki sposób problem zostanie rozwiązany za pomocą rzeczywistego kodu — etap ten następuje, zanim w ogóle zaczniesz pisać kod.

Pomyśl o tym jak o projekcie architektonicznym dla skryptów. Ponownie, każda metodyka inaczej do tego podchodzi, ale naprawdę istotny jest tu fakt, że **musisz posiadać jakiś projekt, zanim zaczniesz pisać kod.**

Dotyczy to zarówno dużych, jak i małych projektów. Niektórzy programiści uczący się podejścia Agile do programowania (omówię to szczegółowo w późniejszym rozdziale) uważają, że nie musisz nic projektować, że możesz od razu zacząć kodować. Choć Agile, nasze programowanie zwinne, nie skupia się aż tak bardzo na stworzeniu planu zawczasu, to **plan i projekt wciąż są potrzebne.**

Nie budujesz domu, losowo kładąc pustaki jeden na drugim.

PISANIE KODU

Kiedy masz już koncepcję projektu swojego programu, nadchodzi czas na rozpisanie testów określających, co program ma robić (nazywamy to też *Test-Driven Development*, w skrócie *TDD*, czyli programowanie oparte na testach), albo na rozpoczęcie kodowania (TDD omówimy w późniejszym rozdziale).

Pisanie kodu to sztuka sama w sobie, więc nie będę go tutaj omawiał, ale polecę dwie świetne książki poświęcone kodowaniu, które powinieneś przeczytać.

Pierwszą z nich jest *Kod doskonały*⁷ autorstwa Steve'a McConnella. To dobrze znana książka, którą powinien przeczytać każdy programista.

Drugą jest *Czysty kod*⁸ autorstwa Roberta Martina, kolejna dobrze znana książka, która pomoże Ci w pisaniu lepszego kodu.

Te dwie prace pomogą Ci nauczyć się, jak porządkować kod i jak go pisać, aby był przejrzysty i łatwy w utrzymaniu.

Obydwie książki miały duży wpływ na moje własne umiejętności programowania, zwłaszcza w zakresie przejrzystości i projektowania.

TESTOWANIE I WDRAŻANIE

Kiedy kod jest już gotowy, dostarczamy go klientowi, prawda?

Nie. **Teraz nadchodzi czas testów**. Ponownie, istnieją różne metodyki i różne podejścia⁹. Ogółem mówiąc, przed przekazaniem kodu użytkownikowi końcowemu musimy w jakiś sposób kod ten przetestować.

Przykładowo, w tradycyjnych projektach tworzonych w modelu kaskadowym testowanie następuje na samym końcu, natomiast w projektach w modelu Agile testowanie odbywa się w trakcie każdej iteracji, trwającej mniej więcej dwa tygodnie.

Kiedy kod zostanie przetestowany, można go wdrożyć, co samo w sobie jest osobnym procesem¹⁰.

Nie będziemy się jeszcze wdawać w szczegóły — poświęcę temu osobny rozdział — ale wdrożenie to proces wgrzania gotowego programu na serwer, publikowania aplikacji w sklepie czy udostępniania oprogramowania w dowolny inny sposób jemu użytkownikom (i proces ten może być dość złożony).

Gdzieś po drodze kod może — a raczej powinien — zostać wprowadzony w repozytoria źródeł, gdzie przechowuje się różne wersje tego samego programu wraz z kolejnymi zmianami.

W najbardziej złożonych aplikacjach, obsługujących jakiegoś rodzaju dane, **z pewnością będziemy też mieć do czynienia z bazą danych.**

Baza danych zazwyczaj przechowuje dane użytkowników związane z aplikacją albo ustawienia konfiguracyjne i również musi być aktualizowana, na równi z kodem źródłowym.

Wiele zespołów tworzących oprogramowanie korzysta z jakichś procesów ciągłej integracji, która automatyzuje kompilowanie kodu, kiedy kolejni programiści wprowadzają do niego zmiany.

PISANIE KODU TO COŚ WIĘCEJ NIŻ PISANIE KODU

W końcu nie zapomnijmy o procesie debugowania. Będąc programistą, sporą część swojego czasu spędzisz na dumaniu, dlaczego Twój kod — albo kod kogoś innego — nie działa.

Jak więc widzisz, tworzenie oprogramowania to coś więcej niż po prostu pisanie kodu.

Będziesz musiał pamiętać o wszystkich powyższych kwestiach, zanim otrzymasz pracę na stanowisku programisty. Miejmy nadzieję, że w niektórych z tych aspektów tworzenia oprogramowania będziesz też miał odrobinę doświadczenia i umiejętności.

Ale nie obawiaj się. **Celem tej książki jest przygotować Cię na wszystkie te ewentualności.** A przynajmniej skierować Cię do odpowiednich źródeł wiedzy. Być może będziesz musiał spakować swój plecak samodzielnie, ale przynajmniej powiem Ci, co powinieneś do niego schować.

MIEJ PLAN

No dobrze, John, rozumiem to całe tworzenie oprogramowania, wiem, że to coś więcej niż tylko kodowanie i że sporo godzin poświęcę na szukanie błędów. Ale wciąż mi nie powiedziałeś, jak mam zacząć. No więc?

Tak, w pełni Cię rozumiem, ale wiesz co? Mam dobrą wiadomość:

Już zacząłeś. Gratuluję!

Biorąc do ręki książkę, taką jak choćby ta, i faktycznie próbując zrozumieć, że tworzenie oprogramowania to coś więcej niż tylko kodowanie, masz **lepsy start niż większość innych programistów**.

No dobrze, dobrze. Wiem, że to tylko takie miłe gadanie, ale to jednocześnie prawda. Pewnego dnia, kiedy będziesz już takim zrzedliwym starym programistą jak ja, będziesz mówił te same rzeczy.

A teraz, trochę poważniej i praktyczniej, potrzebujesz planu.

Tak, planu. Prawdziwego, rzeczywistego planu, w którym nie lejesz wody. Powinien pokazywać, jak masz zmienić się z osoby, która kompletnie nic nie wie o programowaniu, w doświadczonego programistę.

Do tego punktu możesz dotrzeć wieloma drogami — niektóre z nich omówię w kolejnych rozdziałach — ale przede wszystkim pamiętaj, że nieważne, którą drogę wybierzesz. Ważne jest to, byś którąś drogę wybrał i się jej trzymał.

OKREŚLANIE PLANU

Porozmawiajmy o tym, co powinien zawierać Twój plan.

Zacznijmy od tego, że musisz **szczerze ocenić**, gdzie jesteś w tym momencie i czego będziesz się musiał nauczyć.

Czy masz doświadczenie w programowaniu?

Czy znasz jakiś język programowania?

Czy kiedykolwiek stworzyłeś jakiś program, a może zaczynasz kompletnie od zera?

Co z innymi umiejętnościami, opisanymi wcześniej?

Czy któreś z nich posiadasz?

Czy wiesz cokolwiek o bazach danych, kontroli wersji, TDD, testowaniu, debugowaniu czy metodykach tworzenia oprogramowania?

Zapytaj też sam siebie, **jakiego rodzaju oprogramowanie chcesz tworzyć.**

Pewnie, wszyscy chcą robić gry komputerowe¹¹, ale czy jest to praktyczne? I czy od tego chcesz rzeczywiście zacząć? Czy jesteś gotów poświęcić długie godziny i walczyć z konkurencją, którą napotkasz na swojej drodze?

Wiele osób obiera w życiu jakiś kierunek, zapominając, by wszystko dokładnie przemyśleć.

Poświęć chwilę na udzielenie odpowiedzi na te pytania, abyś mógł dzięki temu opracować dobry plan.

Nie zrozum mnie źle. Pomogę Ci w miarę moich możliwości, ale **nie zrobię wszystkiego za Ciebie.**

Mogę przekazać Ci wszystkie informacje, których potrzebujesz, aby zostać dobrym, a nawet świetnym programistą, ale Ty będziesz musiał je **uporządkować i stworzyć plan działania** dopasowany do Ciebie. A potem *osobiście* będziesz musiał się nim kierować.

TWORZENIE PLANU

Kiedy już przemyślałeś odpowiedzi na wcześniejsze pytania, możemy zabrać się za właściwy plan.

Najlepszym sposobem na stworzenie planu jest **opracowanie go wstecz, od miejsca, do którego chcesz dotrzeć.**

Zamiast „nauczyć się programowania” albo „zostać programistą”, powinieneś postawić sobie **konkretny cel, określić, jakiego rodzaju programistą chcesz zostać.**

W części „Co musisz wiedzieć o tworzeniu oprogramowania?” omówię różne rodzaje ról pełnionych przez programistów, które może zechcesz rozważyć. Ale sam też możesz zbadać temat i określić, co Ci najbardziej odpowiada.

Chcesz być **możliwie najbardziej konkretny**, dzięki czemu będziesz wiedział, czego dokładnie musisz się nauczyć, jak przygotować CV i portfolio, jakimi szkołami czy kursami powinieneś się zainteresować i jakie stanowiska Cię interesują.

Wiem, że trudno jest podjąć decyzję, zobowiązać się do czegoś, ale muszę podkreślić, że to bardzo istotny etap!

Im konkretniejszy będziesz w związku z tym, jakim programistą chcesz zostać, **tym łatwiejsza będzie Twoja droga.**

Będziesz wiedział, czego musisz się nauczyć i co musisz zrobić na każdym etapie drogi.

KTO CHCE BYĆ „SPORTOWCEM”?

Pomyśl o tym w ten sposób: załóżmy, że chcesz zostać „sportowcem”.

To bardzo szeroki termin. Co powinieneś trenować, żeby zostać „sportowcem”?

Może powinieneś podnosić ciężary i biegać, a może trenować pływanie. Może powinieneś praktykować zamach raketą tenisową.

Może lepiej trenować wszystkie te rzeczy, żebyś był przygotowany do dowolnej dyscypliny, którą ostatecznie się zajmiesz.

Widzisz, jakie to niepoważne?

Tak samo niepoważnie — a może i bardziej — brzmi stwierdzenie, że ktoś chce być „programistą”.

Powinieneś raczej **wybrać konkretną dyscyplinę.**

Kiedy już ją wybrałeś, możesz trenować to, czego trzeba, a to znacznie ułatwi Ci życie — uwierz mi.

Najpierw określ cel, a potem przemyśl, czego potrzebujesz, aby go osiągnąć.

Kiedy już to zrobisz, będziesz mógł stworzyć plan.

Początek planu powinien skupiać się na tym, jakiej wiedzy i umiejętności będziesz potrzebował. Musisz zrobić dwie ważne rzeczy: określić, w jakiej kolejności powinieneś opanować pewne zagadnienia, oraz zastanowić się, w jaki sposób te umiejętności i wiedzę posiadasz.

Potem powinieneś pomyśleć, czego będziesz potrzebował, aby znaleźć pracę.

W końcu będziesz potrzebował planu, żeby tę pracę rzeczywiście dostać. Gdzie jej poszukasz? Co będziesz robił? **Jakiej pracy w ogóle szukasz?**

Sam dodałbym jeszcze plan określający, jak będziesz się dalej rozwijał, kiedy już dostaniesz pracę,

Może to trochę przytłaczające, wiem, ale nie martw się tym teraz. Książkę tę napisałem, **żeby Ci to wszystko ułatwić.**

W kilku kolejnych rozdziałach pomogę Ci ustalić, czego potrzebujesz i jak możesz zdobyć wiedzę, zaś w kolejnych częściach omówię to, w jaki sposób znaleźć pracę w branży.

Na razie możesz zacząć od myślenia, **jak będzie wyglądał Twój plan.** A potem pomyśl, **jakim programistą chcesz zostać.**

HEJ, JOHN!

Ale jakim programistą chcę zostać?

Dobre pytanie. Jeśli dopiero zaczynasz, możesz nawet nie wiedzieć, co masz do wyboru. No, może poza programistą gier komputerowych.

Na szczęście, to wcale nie takie trudne do określenia — choć może wymagać zgłębienia tematu.

W tej książce, głównie w części „Co musisz wiedzieć o tworzeniu oprogramowania?”, omówię niektóre typy programistów, ale sam możesz poszukać informacji, na przykład w internecie.

Jeśli znasz jakichś programistów, zapytaj ich, jakie istnieją rodzaje programowania oraz jakim oni sami się zajmują.

Pomyśl, co chciałbyś stworzyć, i poszukaj informacji na temat powiązanych z tym technologii i języków programowania.

Jest sporo różnych technologii i rozwiązań, na których możesz się skupić jako programista.

Czy chcesz tworzyć aplikacje internetowe? Aplikacje mobilne? Może chcesz pisać kod, który sprawia, że Twoja lodówka poprawnie chłodzi? A może chcesz pisać programy, które wysyłają astronautów na orbitę?

Pomyśl, poczytaj, zgłęb temat. Jeśli zadaje się właściwe pytania, odpowiedzi przychodzą z łatwością.

SOLIDNY PRZYKŁAD

Przykłady zawsze się przydają, spójrzmy więc na realistyczny scenariusz dla kogoś, kto chce zostać programistą stron internetowych skupiającym się na Node.js:

Cel: zostać programistą Node.js.

Plan

Etap nauki

- Opanować podstawy JavaScript.
- Nauczyć się tworzenia stron internetowych i odpowiednich języków, takich jak HTML i CSS.
- Poznać podstawy Node.js.
- Nauczyć się pisać proste aplikacje internetowe w Node.js.
- Poznać różne frameworki i technologie, których używa się do tworzenia aplikacji Node.js.
 - Lepiej poznać niektóre z tych frameworków.
- Opanować jakieś rozwiązania z zakresu baz danych, by wykorzystywać je na potrzeby Node.js.
- Opanować podstawy informatyki:
 - Algorytmy.
 - Struktury danych.
- Poznać dobre praktyki dotyczące pisania kodu.
- Nauczyć się, jak projektować architekturę aplikacji Node.js.

Etap szukania pracy

- Zacząć szukać ofert pracy dla programistów Node.js w mojej okolicy, popatrzeć, jakie pracodawcy mają wymagania.
- Sporządzić listę miejscowych firm, w których chciałbym dostać pracę.
- Zacząć uczęszczać na spotkania branżowe w okolicy.
- Zacząć nawiązywać kontakty z innymi miejscowymi programistami Node.js.
- Zatrudnić specjalistę, żeby pomógł mi stworzyć dobre CV.
- Przećwiczyć pytania z rozmów kwalifikacyjnych związane z kodowaniem.
- Przećwiczyć przykładowe rozmowy.
- Zbudować portfolio zawierające kilka aplikacji, by zademonstrować swoje umiejętności.

Etap zdobycia pracy

- Skontaktować się z osobami z mojego kręgu kontaktów. Poinformować je, co umiem robić i czego szukam.
- Zacząć składać podania o staż czy etat dla młodszych programistów.
- Postarać się odpowiadać na dwie oferty dziennie.
- Po rozmowach kwalifikacyjnych przemyśleć je, żeby określić, nad czym jeszcze trzeba popracować.

Twój plan nie będzie na początku zbyt wyrafinowany, ale z czasem odkryjesz, czego musisz się nauczyć i co musisz robić, a wtedy będziesz mógł go rozbudować.

To ważne, by mieć na początku jakiś rodzaj planu. Zawsze możesz go zmienić, dostosować do zmieniającej się sytuacji, ale jeśli w ogóle nie masz planu, będziesz działał na oślep. I zapewne pojawi się wtedy frustracja, która może sprawić, że się poddasz.

W kolejnym rozdziale pomogę Ci dopracować plan. Omówimy umiejętności praktyczne, które musisz opanować, jeśli chcesz zostać programistą.

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**

PRZEPIS NA KARIERĘ I SUKCES!

Dobry programista powinien tworzyć udane aplikacje, pisać poprawny kod i znać kilka uznanych języków programowania. Jeśli spełnia te kryteria, na pewno znajdzie pracę marzeń. Tak przynajmniej tak sądzi wiele osób, które postanowiły rozpocząć karierę w tym zawodzie. Prawda jest jednak taka, że udana kariera musi zostać przemyślana i zaplanowana, a poza wiedzą techniczną koniecznie trzeba rozwijać w sobie tak zwane kompetencje miękkie. W ten sposób można uniknąć poważniejszych zawirowań i spokojnie kroczyć ścieżką kariery developera.

Dzięki tej książce pewnie wkroczysz w branżę tworzenia oprogramowania. Znajdziesz tu mnóstwo przydatnych informacji o specyfice pracy developera. Dowiesz się, jak napisać dobre CV i uniknąć pułapek rekrutacji, na przykład jak zacząć, jeśli brakuje Ci doświadczenia. Poradzisz sobie także z wymagającym przełożonym, uprzedzeniami w miejscu pracy i rozwiążesz problemy z pracą zespołową. Nauczysz się utrzymywać równowagę pomiędzy życiem zawodowym a osobistym. Zawarte w tej książce praktyczne wskazówki i porady pozwolą Ci zaplanować i poprowadzić najlepszą dla Ciebie karierę zawodową programisty! To publikacja idealna zarówno dla świeżo upieczonych programistów, jak i dla senior developerów.

Dzięki tej książce dowiesz się:

- Jak zacząć karierę programisty i znaleźć pierwszą pracę
- Jakich języków programowania warto się nauczyć na startcie
- Jak wybrać odpowiedni dla siebie model zatrudnienia i rodzaj specjalizacji
- Jak zarządzać relacjami z szefem i ze współpracownikami
- Czy i kiedy niezbędne jest wykształcenie informatyczne
- Jak wyłapać luki w wiedzy i je uzupełniać

JOHN SONMEZ

— programista i konsultant specjalizujący się w automatyzacji testów i metodyce Agile. Spektakularne sukcesy zawodowe zaczął odnosić dopiero wtedy, gdy zrozumiał, że wybitna wiedza o tym, jak rozwijać oprogramowanie, to za mało, by stać się doskonałym programistą. Opublikował wiele kursów online dotyczących projektowania i budowy aplikacji. Jest też autorem świetnie sprzedających się książek.

| | | |
|--|---|--|
|  | <i>Sprawdź nasze szkolenia!</i> | KOD KORZYŚCI Słęgnij po więcej ▶  |
|  helion.pl |  AKADEMIA IT & BUSINESS | ISBN 978-83-283-4393-1 |
|  HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl | WWW.SZKOLENIA.HELION.PL |  9 788328 343931 |
| INFORMATYKA W NAJLEPSZYM WYDANIU | | Cena: 99,00 zł |