CORY ALTHOFF

PROGRAMISTA *samouk*

Profesjonalny
przewodnik
do samodzielnej
nauki kodowania

Helion

Tytuł oryginału: The Self-Taught Programmer: The Definitive Guide to Programming Professionally
Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-3950-7

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)

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

Printed in Poland.

Spis treści

Rozdział

1

Wprowadzenie

„Większość dobrych programistów programuje nie dlatego,
że oczekują zapłaty lub powszechnego uwielbienia, ale dlatego, że jest to świetna zabawa”.

— Linus Torvalds

Studiowałem nauki polityczne na Uniwersytecie Clemson. Zwróciłem wtedy uwagę na informatykę i nawet zapisałem się na pierwszym roku na kurs „Wprowadzenie do programowania”, ale szybko z niego zrezygnowałem. Był dla mnie zbyt trudny. Kiedy jednak po zakończeniu studiów zamieszkałem w Dolinie Krzemowej, uznałem, że muszę się nauczyć programowania. Po roku pracowałem jako programista drugiego stopnia w eBay (to lepiej niż programista pierwszego stopnia i gorzej niż starszy programista). Nie chciałbym jednak sprawiać wrażenia, że to było łatwe. Było to niezwykle wymagające, ale w przerwach pomiędzy rzucaniem czym popadnie po ścianach miałem sporo frajdy.

Swoją przygodę z nauką programowania rozpocząłem od języka Python, popularnego języka programowania. Jednak celem tej książki nie jest jedynie nauczenie czytelników programowania w konkretnym języku, choć —oczywiście — chodzi także to. Jej celem jest przedstawienie wszystkiego, czego nie można znaleźć w standardowych źródłach informacji. Traktuje ona o rzeczach, których sam nauczyłem się po drodze, zostając programistą. Książka nie jest przeznaczona dla kogoś poszukującego zwyczajnego wprowadzenia do zagadnień programowania, by potem pisać kod w ramach hobby. Napisałem ją specjalnie z myślą o osobach, które pragną zajmować się programowaniem zawodowo. Niezależnie od tego, czy celem jest zostanie programistą, przedsiębiorcą, czy też wykorzystanie nowych umiejętności programistycznych w jeszcze innej profesji, tę książkę napisałem właśnie dla Ciebie.

Nauczenie się nowego języka programowania to tylko część całej batalii. Są także inne umiejętności, które trzeba posiąść, by porozumiewać się w języku informatyków. Nauczę Cię wszystkiego, czego sam nauczyłem się na drodze od zupełnego amatora do profesjonalnego programisty. Napisałem tę książkę, by pokazać osobom marzącym o zostaniu programistami zarys tego, co muszą umieć. Jako programista samouk nie wiedziałem, czego mam się uczyć. Książki zawierające wprowadzenie do programowania wszystkie są takie same. Uczą podstaw programowania w językach Python lub Ruby i pozostawiają nas samym sobie. Osoby, które przeczytały takie książki, często pytały: „Co mam teraz zrobić? Jeszcze nie jestem programistą i nie wiem, czego mam się uczyć w następnej kolejności". Ta książka jest moją odpowiedzią na to pytanie.

Struktura książki

Wielu zagadnieniom poruszanym w pojedynczym rozdziale tej książki można by poświęcić odrębną pozycję — i w wielu przypadkach tak się właśnie dzieje. Moim celem jednak nie jest opisywanie każdego szczegółu każdego zagadnienia, które może Ci się przydać. Moim celem jest przedstawienie mapy — zarysu wszystkich umiejętności, które będziesz musiał zdobyć, by zawodowo zajmować się programowaniem. Książka została podzielona na pięć części.

Część I. — Wprowadzenie do programowania. Swój pierwszy program napiszesz tak szybko, jak to tylko możliwe. Miejmy nadzieję, że będzie to już dziś.

Część II. — Wprowadzenie do programowania obiektowego. W tej części opisuję różne paradygmaty programowania, koncentrując się na programowaniu obiektowym. Napiszesz w tej części grę, która pokaże ogromne możliwości, jakie daje programowanie. Jestem pewny, że po przeczytaniu tej części książki będziesz zafascynowany programowaniem.

Część III. — Wprowadzenie do narzędzi programistycznych. W tej części książki pokażę różne narzędzia programistyczne, które przeniosą wydajność pracy na wyższy poziom. Na tym etapie jesteś już zapewne zafascynowany programowaniem i chcesz być coraz lepszy. W tym miejscu dowiesz się nieco więcej o systemach operacyjnych, o możliwościach stosowania wyrażeń regularnych w celu poprawy wydajności pracy, o instalowaniu programów napisanych przez innych i zarządzaniu nimi oraz o współpracy z innymi programistami przy wykorzystaniu systemów kontroli wersji.

Część IV. — Wprowadzenie do informatyki. Ta część zawiera krótkie wprowadzenie do informatyki i jest poświęcona dwóm podstawowym zagadnieniom, czyli algorytmom i strukturom danych.

Cześć V. — Zdobywanie pracy. Ostatnia część książki jest poświęcona najlepszym praktykom programistycznym, zdobywaniu pracy w charakterze programisty, pracy w zespole oraz samodoskonaleniu programistów. Zawiera porady, jak przejść techniczną rozmowę kwalifikacyjną na stanowisko programisty i jak pracować w zespole oraz informacje o tym, jak dalej poszerzać swoje umiejętności.

Jeśli nie dysponujesz jeszcze żadnymi umiejętnościami w zakresie programowania, podczas lektury wszystkich części książki jak najwięcej samodzielnie ćwicz programowanie. Nie staraj się przeczytać tej książki zbyt szybko. Warto jej używać jak przewodnika i pomiędzy lekturą kolejnych rozdziałów jak najwięcej ćwiczyć.

Zacznijmy od końca

Sposób, w jaki nauczyłem się programować, jest dokładnie odwrotny od standardowego sposobu nauki informatyki i programowania, a struktura tej książki odpowiada mojemu tokowi nauki. W tradycyjnym sposobie nauki na początku wiele czasu poświęca się na poznanie teorii — w rzeczywistości przeznacza się na to tak wiele czasu, że wiele osób, które skończyły kursy informatyki, wcale nie potrafi pisać programów. Jeff Atwood na swoim blogu, w poście „Why Can't Programmers… Program?"[1] pisze: „Autor, podobnie jak ja, nie może zrozumieć faktu, że 199 na 200 starających się o posadę programisty nie potrafi pisać kodu. Powtarzam: w ogóle nie potrafi napisać żadnego kodu". To spostrzeżenie skłoniło Atwooda do stworzenia programistycznego wyzwania o nazwie **FizzBuzz** — programistycznego testu używanego na rozmowach kwalifikacyjnych w celu eliminacji kandydatów. Większość kandydatów oblewa ten test i właśnie z tego powodu podczas lektury tej książki tak wiele czasu poświęcimy na naukę umiejętności, które będą niezbędne w praktyce. Jednak nie masz się czego obawiać: nauczysz się także wszystkiego, czego potrzeba, by zdać test FizzBuzz.

Bohater filmu *Szachowe dzieciństwo*, w swojej książce *Sztuka nauki*, opisuje, jak nauczył się grać w szachy w sposób odwrotny do standardowo przyjętego. Zamiast studiować otwarcia, zaczął od nauki gry końcowej (w której na planszy pozostaje jedynie kilka figur). Taka strategia pozwoliłam mu lepiej zrozumieć grę i wygrać wiele zawodów. Na podobnej zasadzie uważam, że lepsze efekty daje nauczenie się najpierw programowania i poznawanie teorii później, kiedy już będziemy chcieli wiedzieć, jak wszystko działa. Właśnie z tego względu teoretyczne zagadnienia informatyki wprowadzam dopiero w czwartej części książki, a w dodatku przedstawiam je w jak najkrótszej postaci. Choć teoria jest ważna, jednak jej znajomość stanie się jeszcze cenniejsza, kiedy zostanie poparta praktycznymi doświadczeniami programistycznymi.

Nie jesteś sam

Nauka programowania poza szkołami czy uczelniami staje się coraz popularniejsza. W 2015 roku serwis Stack Overflow (internetowa społeczność programistów) przeprowadził ankietę, z której wynikało, że ponad 48% respondentów nie ma żadnego wykształcenia związanego z programowaniem lub informatyką[2].

Zalety samodzielnej nauki

Kiedy zostałem zatrudniony w eBay, znalazłem się w zespole programistów mających tytuły magistrów informatyki zdobyte na uniwersytetach Stanforda, Cal, Duke, były w nim nawet dwie osoby po

[1] Dlaczego programiści nie mogą… programować? — *przyp. tłum.*

[2] *http://www.infoworld.com/article/2908474/application-development/stack-overflowsurvey-finds-nearly-half-have-no-degree-in-computer-science.html*

doktoratach. Dla mnie, osoby w wieku 25 lat, było krępujące, że moi 21-letni koledzy wiedzieli o programowaniu i informatyce 10 razy więcej niż ja.

Niezależnie od tego, jak onieśmielające byłoby pracowanie z osobami z tytułami licencjatów, magistrów czy też nawet doktorów informatyki, nigdy nie zapominaj, że będziesz dysponował tym, czym dysponowałem ja — „zaletą samodzielnej nauki". Nie sięgnąłeś po tę książkę, bo kazał Ci to zrobić nauczyciel czy wykładowca, ale dlatego, że pragniesz się czegoś nauczyć, a pragnienie poznania czegoś jest największą zaletą i przewagą, jaką można dysponować. Poza tym nie wolno zapominać, że wiele z osób, które w świecie komputerów odniosło największe sukcesy, było właśnie samoukami. Steve Wozniak, współzałożyciel firmy Apple, jest programistą samoukiem. Podobnie jak Margaret Hamilton, która otrzymała Prezydencki Medal Wolności za swoją pracę w NASA nad misją Apollo, David Karp — założyciel Tumblr, Jack Dorsey — założyciel Twittera oraz Kevin Systrom — założyciel Instagrama.

Dlaczego należy programować?

Programowanie może pomóc w karierze niezależnie do profesji. Uczenie się programowania wzmacnia. Uwielbiam wymyślanie nowych pomysłów i zawsze chodzi mi po głowie jakiś nowy projekt, który chciałbym rozpocząć. Kiedy nauczyłem się programowania, mogłem siąść i realizować swoje pomysły bez poszukiwania kogoś, kto mógłby to zrobić dla mnie.

Programowanie sprawi, że można być lepszym we wszystkim, co się robi. Nie ma wielu zagadnień, które nie skorzystałyby na umiejętności dokładnego przeanalizowania i określenia rozwiązania problemu. Ostatnio musiałem wykonać bardzo żmudne zadanie polegające na poszukiwaniu lokum na witrynie Craiglist. Potrafiłem jednak napisać program, który robił to za mnie i wysyłał mi wyniki e-mailem. Nauka programowania może więc uwolnić od konieczności wykonywania powtarzających się czynności.

Jeśli ktoś chce stać się programistą, warto wiedzieć, że zapotrzebowanie na nich jest coraz większe i nie ma dostatecznie wielu odpowiednio wykwalifikowanych kandydatów, by zaspokoić potrzeby. Szacuje się, że do 2020 roku ponad milion wakatów na programistów pozostanie nieobsadzonych[3]. Jeśli nawet zostanie zawodowym programistą nie jest Twoim celem, umiejętność programowania jest także ceniona w takich branżach jak nauka oraz finanse.

Konsekwencja i zainteresowanie

By poprawiać swoje umiejętności programistyczne, należy programować codziennie. Jedyną rzeczą, jaka ewentualnie mogłaby powstrzymać Cię przed codziennym ćwiczeniem, jest brak zainteresowania programowaniem. Poznaj zatem dwie metody, by do tego nie dopuścić.

Jeśli nie masz żadnego doświadczenia programistycznego, a rozpoczęcie tej podróży napawa Cię obawą, jestem pewny, że i tak sobie poradzisz. Istnieje wiele nieprawdziwych opinii dotyczących

[3] *http://www.wsj.com/articles/computer-programming-is-a-trade-lets-act-like-it-1407109947?mod=e2fb*

programowania, takich jak ta, że wymaga ono doskonałej znajomości matematyki. Otóż nie wymaga. Wcale nie trzeba być świetnym z matematyki, by się uczyć programowania, choć jej znajomość może pomóc. A zatem bardzo wiele zagadnień opisywanych w tej książce będzie łatwiejszych do opanowania, niż można przypuszczać.

Kiedy zaczynałem, używałem listy zadań do wykonania, która zapewniała, że programowałem każdego dnia i zachowywałem koncentrację.

Jeśli potrzebujesz dodatkowej pomocy, Tim Ferriss, specjalista do spraw efektywności, zaleca stosowanie następującej techniki pozwalającej na zachowanie wysokiego poziomu motywacji: daj przyjacielowi lub jakiemuś członkowi rodziny pewną sumę pieniędzy wraz z instrukcją, by Ci ją oddał, kiedy zrealizujesz cele w zadanych ramach czasowych, bądź też, by przekazał jakiej nielubianej organizacji, jeśli Ci się to nie uda.

Postać książki

Każdy kolejny rozdział tej książki bazuje na poprzednich. Starałem się unikać w niej wielokrotnego wyjaśniania tych samych pojęć, więc warto starać się je zapamiętywać. Ważne terminy w miejscu ich pierwszego wystąpienia są zapisywane pogrubioną czcionką. Na końcu każdego rozdziału znajduje się słowniczek, w którym wszystkie te ważne terminy są definiowane. Na końcu każdego rozdziału znajdują się także zadania, których celem jest poprawienie umiejętności programistycznych, tam także znajdują się odnośniki do rozwiązań tych zadań.

Technologie stosowane w książce

Książka uczy pewnych technologii, by zapewnić Ci możliwość zdobycia jak największej liczby praktycznych umiejętności programistycznych. Staram się jednak zachować neutralność w doborze technologii, koncentrując się nie na nich, lecz raczej na ogólnych koncepcjach.

W niektórych przypadkach musiałem jednak wybrać jedną spośród wielu dostępnych technologii. W rozdziale 19., „Kontrola wersji” (jeśli nie wiesz, czym ona jest, wyjaśnię o co chodzi nieco później), przedstawiam podstawowe informacje o stosowaniu systemu Git — popularnego systemu kontroli wersji. Wybrałem Git, gdyż uważam że stanowi przemysłowy standard kontroli wersji. W większości przykładów programistycznych używam języka Python, gdyż jest bardzo popularnym językiem do nauki programowania, a jego kod jest łatwy do czytania i analizy nawet dla osób, które nigdy wcześniej go nie używały. Oprócz tego osoby znające Pythona są bardzo poszukiwane niemal na każdym rynku pracy.

Oczywiście, do wykonywania przykładów prezentowanych w tej książce konieczny będzie komputer. Każdy komputer ma **system operacyjny** — program pełniący rolę pośrednika pomiędzy komponentami sprzętowymi komputera a człowiekiem. To, co widzimy, patrząc na ekran komputera, jest nazywane **graficznym interfejsem użytkownika** (w skrócie GUI od angielskich słów: *graphical user interface*) i stanowi jeden z elementów systemu operacyjnego.

Istnieją trzy najpopularniejsze systemy operacyjne używane na komputerach biurkowych i laptopach; są to **Windows**, **Unix** oraz **Linux**. Windows jest systemem stworzonym przez firmę Microsoft. Unix z kolei powstał już w latach 70. ubiegłego wieku. Obecny system operacyjny używany przez firmę Apple bazuje właśnie na Uniksie. Od tej pory będę używał nazwy Unix także wtedy, gdy będę miał na myśli system operacyjny stosowany na komputerach Apple. Z kolei Linux — system rozpowszechniany jako **oprogramowanie otwarte** (typu *open source*) — jest używany na ogromnej większości istniejących obecnie **serwerów**. Serwer to komputer lub program komputerowy wykonujący jakieś zadanie, na przykład udostępniający witrynę WWW. Oprogramowanie otwarte (nazywane także oprogramowaniem typu *open source*) to program, który nie jest własnością żadnej firmy ani osoby i który może być rozpowszechniany i modyfikowany bez żadnych ograniczeń. Linux oraz Unix są **systemami uniksowymi**, co oznacza, że są do siebie bardzo podobne. W tej książce zakładam, że używasz komputera działającego pod kontrolą systemów Windows, Unix lub Ubuntu (popularnej wersji systemu operacyjnego Linux).

Słownictwo

FizzBuzz. Test programistyczny stosowany do odsiewania kandydatów.

Graficzny interfejs użytkownika (GUI). Ta część systemu operacyjnego, którą użytkownik może oglądać na ekranie komputera.

Linux. System operacyjny typu *open source* używany na większości serwerów na świecie.

Oprogramowanie typu *open source*. Oprogramowanie, które nie jest własnością żadnej firmy ani konkretnej osoby; nad jego rozwojem pracuje grupa ochotników.

Serwer. Komputer lub program komputerowy wykonujący konkretne zadanie, takie jak udostępnianie witryny WWW.

System operacyjny. Program pełniący rolę pośrednika pomiędzy fizycznymi komponentami komputera a człowiekiem.

Uniksowy system operacyjny. Unix lub Linux.

Unix. System operacyjny stworzony w latach 70. ubiegłego wieku. Bazuje na nim obecny system używany na komputerach firmy Apple.

Windows. System operacyjny firmy Microsoft.

Wyzwanie

1. Przygotuj codzienną listę rzeczy do wykonania, obejmującą ćwiczenie programowania.

Rozwiązania można znaleźć w przykładach dołączonych do książki, które możesz pobrać z serwera FTP wydawnictwa Helion *ftp://ftp.helion.pl/przyklady/proprs.zip*.

Skorowidz

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

JEŚLI CHCESZ NAUCZYĆ SIĘ PROGRAMOWANIA, ZACZNIJ OD RAZU!

Nie wystarczy znajomość jednego języka programowania, aby zostać programistą. W rzeczywistości trzeba opanować dość szeroki zakres pojęć i paradygmatów, a także zrozumieć zagadnienia związane z algorytmami. Trzeba być na bieżąco z nowymi technologiami i narzędziami. Należy również poznać i zacząć stosować dobre praktyki programistyczne oraz przyswoić sobie zasady pracy w zespole. Przede wszystkim jednak priorytetem jest sama praktyka, ponieważ wielu programistów wciąż ma problem z pisaniem poprawnego kodu.

Jeśli chcesz być profesjonalistą i postanowiłeś nauczyć się wszystkiego, co jest do tego potrzebne, to wziąłeś do ręki właściwą książkę. Zawiera ona znacznie więcej informacji niż proste wprowadzenie do konkretnego języka programowania. Opisano tu najpotrzebniejsze technologie, elementy kodu i zasady ich stosowania, sporo miejsca poświęcono obiektowości. Zapoznasz się również z najważniejszymi narzędziami programistycznymi i nauczysz się dobrych praktyk programistycznych. To wszystko czyni z tej książki świetną mapę umiejętności, dzięki którym szybko i sprawnie staniesz się prawdziwym profesjonalistą!

W tej książce między innymi:

- wprowadzenie do **programowania**
- **struktury** danych i algorytmy
- **pakiety i kontrola** wersji
- **testowanie i bezpieczeństwo** tworzonych aplikacji
- **zasady pracy** zespołu programistów

CORY ALTHOFF

jest programistą samoukiem i autorem książek. Pracował dla eBaya i kilku innych firm w Dolinie Krzemowej. Biegle posługuje się Pythonem, Javą, JavaScriptem i kilkoma innymi językami programowania.

ISBN 978-83-283-3950-7
9 788328 339507
Cena: 39,90 zł