Wstęp

O autorze, o książce

Kiedy miałem siedem lat, zostałem dumnym posiadaczem swojego pierwszego, własnego komputera Atari 800XL wraz z magnetofonem oraz dwoma dżojstikami; do zestawu był również dołączony zestaw kaset z przeróżnymi grami, takimi jak *River Ride*, *Bruce Lee* czy *Archon*. Wczytanie gry trwało zazwyczaj od pięciu do dziesięciu minut, a więc całą wieczność dla niecierpliwego dziecka. Na szczęście jedna z gier nie wymagała kasety i była dostępna od razu po włączeniu komputera – nazywała się *Atari BASIC* i była grą w programowanie. Co więcej, była do niej dołączona instrukcja w formie zdobytej cudem kserokopii książki, która dla osoby rozpoznającej litery, ale niekoniecznie umiejącej złożyć je w wyrazy, stanowiła zagadkę, tak samo jak *BASIC*. Niemniej jednak umiejętność czytania nie jest potrzebna, by przepisywać kolejne znaki z przykładowych listingów, co też z upodobaniem czyniłem – najpierw z kserówek, a potem z kolejnych numerów czasopisma „Bajtek", po które co miesiąc udawałem się wraz z którymś z rodziców do pobliskiego kiosku.

Na samym przepisywaniu listingów się nie kończyło – jeszcze większą radość przynosiło mi nanoszenie rozmaitych zmian we wprowadzonych programach i obserwowanie ich skutków. W efekcie okazało się, że parametry rozkazu `REM` były bez znaczenia (z czym wiązała się inna świetna wiadomość – nie trzeba było ich przepisywać!), parametry komendy `PRINT` były wypisywane na ekran, po instrukcji `GOSUB` musiało znaleźć się `RETURN`, a `GRAPHICS` na spółkę z `PLOT` i `DRAWTO` pozwalały rysować kolorowe linie. Z każdym przepisanym listingiem poznawałem coraz więcej wzorców, a z każdą wprowadzoną zmianą coraz lepiej rozumiałem to, co właściwie działo się w programie.

Bardzo szybko zacząłem również pisać własne, proste programiki, które z biegiem czasu stawały się coraz dłuższe i bardziej złożone. Wkrótce potem w domowym salonie stanął komputer kompatybilny z IBM PC, wyposażony w procesor 80286, 1 MB RAM, stację dyskietek 5,25" i kartę graficzną Hercules oferującą jedynie monochromatyczne tryby graficzne i tekstowe. Wraz z nim nadszedł czas nauki nowego wariantu znanego mi już języka – stworzonego przez Microsoft GW–BASIC. Wkrótce potem komputer został doposażony w dysk twardy o pojemności 50 MB, co umożliwiło zainstalowanie pełnego

systemu MS-DOS 5 wraz z interpreterem języka QBasic, a później także Windows 3.1 i potężnym IDE języka Visual Basic 1.0. Czas leciał, leciwy 80286 został wymieniony na wielokrotnie szybszy 80486, a z języka Visual Basic przeniosłem się na Turbo Pascal. W kolejnych latach pojawiły się procesory z rodziny Pentium, pierwsze akceleratory 3D do kart graficznych (wtedy jeszcze jako osobne urządzenia), a także ogólnodostępny Internet za sprawą numeru 0–20 21 22. Wraz z nim uzyskałem dostęp do sieci IRC, grup dyskusyjnych i hobbystycznie prowadzonych stron internetowych poświęconych różnym językom programowania. Tymczasem trafiłem do liceum, komputer przeszedł kolejną metamorfozę i został wyposażony w procesor Intel Celeron taktowany z częstotliwością 333 MHz oraz kartę graficzną z wbudowaną akceleracją 3D, a ja za namową znajomego porzuciłem Turbo Pascal na rzecz C++ i OpenGL; moje zainteresowanie wzbudził też temat inżynierii wstecznej. Potem nadeszły studia na Politechnice Wrocławskiej, kolejne języki programowania w niemałej liczbie i coraz bardziej skomplikowane projekty. Na drugim roku rozpocząłem pierwszą pracę jako programista i *reverse engineer* dla jednej z polskich firm antywirusowych, coraz bardziej interesując się również zagadnieniami związanymi z bezpieczeństwem komputerowym. Wkrótce przeszedłem do hiszpańskiej firmy Hispasec, skończyłem studia inżynierskie, by ostatecznie trafić do firmy Google, w której pracuję do dzisiaj.

Spoglądając w przeszłość, mogę powiedzieć, że moja fascynacja wszystkim, co wiąże się z komputerami, a w szczególności programowaniem, nigdy nie minęła, pomimo że zaczęła się ponad ćwierć wieku temu – jednym z jej efektów było napisanie książki, którą właśnie, czytelniku, trzymasz w ręku. Podczas jej tworzenia starałem się uchwycić preferowane przeze mnie podejście do programowania, które jest zabarwione ciekawością, chęcią zrozumienia tego, jak każdy z używanych mechanizmów działa od środka, a także zamiłowaniem do niskopoziomowych zakątków informatyki. Co za tym idzie, książkę pisałem z myślą o pasjonatach, dla których programowanie jest jednocześnie wyzwaniem i świetną zabawą, bez względu na to, czy programują jedynie dla przyjemności, czy też pracują w zawodzie.

Dobierając zawartość książki, starałem się przede wszystkim wybrać tematy, o które często pytają początkujący i średnio zaawansowani programiści – stąd m.in. obecność rozdziałów o wątkach, gniazdach sieciowych czy plikach binarnych. Do spisu treści trafiły również tematy moim zdaniem istotne dla każdego programisty, takie jak niskopoziomowe kodowanie danych, synchronizacja procesów wielowątkowych czy choćby korzystanie z wiersza poleceń. Chciałem jednak również wskazać, że programowanie nie kończy się na aplikacjach narzędziowych lub „usługowych", stąd obecność ostatniego rozdziału książki zatytułowanego „Programowanie dla zabawy". Jednocześnie zdecydowałem się pominąć pewne, skądinąd bardzo istotne, zagadnienia, takie jak algorytmy, wzorce projektowe czy szczegółowe opisy konkretnych języków programowania – są one opisane w wielu innych, bardziej wyspecjalizowanych publikacjach.

Przykłady w książce zostały napisane w różnych językach programowania – są to przede wszystkim Python, C, C++ oraz Java. Ta nietypowa różnorodność brała się z chęci podkreślenia, że w większości środowisk występują te same podstawowe mechanizmy, których używa się w bardzo podobny sposób niezależnie od wybranego języka programowania, a różnice często sprowadzają się wyłącznie do nazw bibliotek, funkcji, klas itp. Jednocześnie z uwagi na specyfikę poszczególnych języków oraz ich umowną klasyfikację jako

nisko- bądź wysokopoziomowe istnieją zadania łatwiejsze i trudniejsze do osiągnięcia w każdym z nich – na przykład korzystanie z zaawansowanych funkcji oferowanych przez system operacyjny jest łatwiejsze w C i C++, ale już przetwarzanie danych tekstowych jest zdecydowanie prostsze w językach pokroju Python. Ostatecznie nie ma języka idealnego – warto więc, by programista znał ich kilka i zgodnie z zasadą *use the right tool for the right job* oraz własnymi preferencjami wybierał język adekwatny do danego zadania. Tak jak nie istnieje idealny język programowania, tak nie ma również idealnego systemu operacyjnego, stąd tematy poruszane w książce omawiam w kontekście dwóch popularnych rodzin systemów: Microsoft Windows oraz GNU/Linux (w szczególności Ubuntu).

Kontynuując temat przykładowych programów, starałem się również, by listingi miały wewnętrznie spójny styl, ale jednak zróżnicowany pomiędzy sobą – jednym z celów książki było zaprezentowanie fragmentów istniejącego ekosystemu programistycznego, w którym różni programiści stosują odmienne style tworzenia kodu. Warto więc jak najwcześniej nabrać pewnej elastyczności w tej kwestii – choć każdy projekt powinien konsekwentnie kierować się zasadami konkretnego stylu, jego wybór jest często kwestią osobistych preferencji programisty.

Nie w każdym przykładowym kodzie zawarłem pełne sprawdzanie błędów – wynikało to z chęci zwiększenia czytelności listingów i redukcji ich długości, która w przeciwnym razie mogłaby być przytłaczająca dla bardziej początkujących czytelników. W przypadku rozwijania kodu produkcyjnego (tj. o wysokiej jakości) pełne sprawdzanie błędów jest w zasadzie obowiązkowe, choć oczywiście nie każdy tworzony kod musi przystawać do takich standardów. W trakcie pracy nad pomocniczymi narzędziami jednorazowego użytku czy rozwiązaniami zadań podczas konkursów trwających kilka lub kilkadziesiąt godzin obranie drogi „na skróty" może nierzadko zaoszczędzić sporo czasu i okazać się ostatecznie strategią najkorzystniejszą.

W książce oprócz przykładowych listingów kodu występuje również znaczna liczba ramek oznaczonych jako [VERBOSE] oraz [BEYOND]. Celem tych pierwszych jest dokładniejsze przedyskutowanie zagadnień, które pojawiają się w tekście, i są skierowane przede wszystkim do bardziej początkujących czytelników. Zawartość ramek [BEYOND] natomiast znacznie rozwija tematykę i często wykracza poza średni poziom skomplikowania danego rozdziału – jestem przekonany, że nawet zaawansowani programiści mogą w nich znaleźć coś nowego i ciekawego.

Chciałbym również zachęcić do odwiedzenia oficjalnego serwisu „Zrozumieć Programowanie", w którym można znaleźć przykładowe kody źródłowe, erratę, jak i niewielkie forum stworzone z myślą o dyskusji na tematy poruszane w książce, w tym również o ćwiczeniach i flagach. Znajduje się on pod adresem:

http://gynvael.coldwind.pl/book/

Podziękowania

W powstaniu i nadaniu ostatecznego kształtu tej książce swój udział miało wiele osób, którym chciałbym w tym miejscu podziękować. Mojej żonie Arashi Coldwind za wsparcie okazywane podczas całego czasu trwania tego, bądź co bądź, niemałego projektu, jakim jest napisanie książki. Autorowi przedmowy, a jednocześnie redaktorowi merytorycznemu i mojemu dobremu przyjacielowi Mateuszowi Jurczykowi, który na edycję i korektę tej książki poświęcił ogromną ilość własnego czasu. Tomaszowi Łopuszańskiemu, który wraz ze mną przesiadywał do późnych godzin nocnych, przeglądając i szlifując każdy kolejny nadesłany rozdział. Łukaszowi Łopuszańskiemu, który przekonał mnie do napisania tej książki i doprowadził do jej wydania. Sebastianowi Rosikowi za genialny projekt okładki. Recenzentom merytorycznym, którzy wychwycili znaczą liczbę niedociągnięć obecnych w pierwotnej wersji, a byli to: Paweł „KrzaQ" Zakrzewski, Robert „Jagger" Święcki, Mariusz Zaborski, Unavowed, Michał Leszczyński, Michał Melewski, Karol Kuczmarski, Adam „pi3" Zabrocki, Sergiusz „q3k" Bazański, Tomasz ‚KeiDii' Bukowski. Testerom, którzy wskazali dodatkowe błędy i nieścisłości, w składzie: Łukasz „Stiltskin" Głowacki, Nism0, Łukasz „Lord Darkstorm", Trzeciakiewicz, Alan Cesarski. A także Ange Albertiniemu, Mikołajowi Koprasowi, Mattowi Moore'owi, Ferminowi Sernowi oraz Michałowi Zalewskiemu.

Mam nadzieję, Drogi Czytelniku, że książka ta będzie dla Ciebie ciekawą i inspirującą lekturą.

Gynvael Coldwind
Zurych, wrzesień 2015 r.

Zgłaszanie błędów i errata

W idealnym wszechświecie napisałbym książkę bez błędów, w której nie byłoby miejsca na literówki czy brakujące przecinki. Niestety, książka ta pochodzi z naszego wszechświata, więc wbrew wszelkim staraniom błędy na pewno się pojawią.

Z tego względu chciałbym zachęcić Czytelników do zaglądania od czasu do czasu na stronę z erratą, którą będę aktualizował przy okazji każdego znalezionego lub zgłoszonego błędu merytorycznego. Erratę można znaleźć pod adresem:

http://gynvael.coldwind.pl/book/errata

Chciałbym również zachęcić czytelników do zgłaszania wszelkiego rodzaju błędów, zarówno merytorycznych, jak i językowych – wszystkie zauważone niedociągnięcia będą

eliminowane w kolejnych wydaniach książki. Informacje na temat wykrytych błędów proszę zgłaszać pod poniższym adresem:

http://gynvael.coldwind.pl/book/bugbounty

Dodatkowo, naśladując niejako Donalda Knutha[1], postaram się wysłać pamiątkową pocztówkę każdemu czytelnikowi, który jako pierwszy zgłosi dany błąd merytoryczny. Szczegóły tego swoistego *bug bounty* można również znaleźć na wspomnianej stronie internetowej.

[1] Autor znakomitej serii książek pt. „Sztuka programowania" (org. *The Art of Computer Programming*), a także twórca TeX – języka oraz oprogramowania do składu tekstu, które stworzył właśnie na potrzeby swoich publikacji. Znany również z nagradzania osób, które znalazły błędy w jego książkach, pamiątkowymi czekami wystawionymi na jednego „heksadecymalnego dolara" (tj. 256 centów) lub odpowiednio mniejszymi wartościami w przypadku drobniejszych potknięć.