Od autora

Od kilkunastu lat zajmuję się programowaniem. Pierwszym językiem programowania przeznaczonym dla mikrokontrolerów, który poznałem, był asembler mikrokontrolerów z rodziny MCS48. Każdy, kto pamięta te mikrokontrolery, pamięta również kłopoty z pisaniem dla nich programów. Z zazdrością patrzyliśmy wówczas na kolegów z Zachodu, którzy pisali programy dla różnych mikrokontrolerów, a nie dla takich, których wolno było użyć, oraz na piszących w językach wysokiego poziomu, jak PL/M czy C. Dla nas były to narzędzia nie do zdobycia. Na szczęście czasy się zmieniły. Obecnie przeciętny mikrokontroler z pamięcią Flash można kupić w niemal każdym sklepie z podzespołami elektronicznymi. W kilka godzin można również znaleźć schemat programatora do niego oraz zdobyć odpowiednie narzędzia absolutnie za darmo. Niestety, za te najlepsze trzeba zapłacić, i to niekiedy bardzo dużo. Dziś sytuacja jest diametralnie inna niż kiedyś – niektórych hobbystów dziwią układy budowane przy użyciu elementów dyskretnych, nikogo natomiast nie dziwi fakt stosowania mikrokontrolerów. Ich ogromny wybór na rynku oraz atrakcyjna cena sprawiają, że na stałe zadomowiły się one również w konstrukcjach amatorskich.

Niniejsza książka to będzie pewna inwestycja – jeśli ją przeczytasz, będzie inwestycją w wiedzę. Zaprocentuje z całą pewnością, ponieważ język C da Ci wolność wyboru. Nie przykuje Cię do jednego, ściśle określonego typu mikrokontrolera, lecz da swobodę wyboru układu odpowiedniego do konkretnego zastosowania. Z łatwością użyjesz dowolnego mikrokontrolera z rodziny na przykład 8051, by później tylko przy minimalnym wysiłku zmienić go na układ z rodziny ST7 lub HC08. O nie, nie ma nic za darmo, ale nie będziesz musiał od podstaw uczyć się nowego języka programowania, tak jak w przypadku asemblera, a niektóre z Twoich programów prawdopodobnie dadzą się przenieść „wprost". To jest absolutnie unikalna cecha języka C. Nie ma jej żaden inny język programowania, a w swojej praktyce poznałem ich wiele.

Język C, tak samo jak asembler, jest bardzo blisko tak zwanego *hardware*, czyli sprzętu, na którym pracuje. Będziesz musiał poznać nazwy rejestrów, rozmiar pamięci, której wolno Ci użyć, prawdopodobnie pewne funkcje będą musiały być poprawione przed ich przeniesieniem na inny mikrokontroler. Jednak praca, którą wykonasz, jest dużo mniejsza niż ta, którą wykonujesz ucząc się nowej listy rozkazów i pisząc program od nowa. Czas jest najcenniejszą wartością. To, jak wykorzystasz swój czas, wpłynie bezpośrednio na to, co zrobisz i osiągniesz. Poświęć swój czas czemuś bardziej wartościowemu, niż znajomość kilku języków asemblera. Nie zrozum mnie jednak źle: nie próbuję powiedzieć, że nie warto uczyć się asemblera, nie to mam na myśli. Warto jest go znać, ale w dzisiejszym świecie to czas jest najcenniejszą wartością.

Książka ta uczy programowania mikrokontrolerów w języku C praktycznie od podstaw. Uczy poprzez przykłady i praktyczne zastosowania języka do budowy konkretnych aplikacji. Nie poucza, nie teoretyzuje, podsuwa Ci tylko pewne pomysły. Możesz je wykorzystać, możesz skrytykować. To nawet bardzo dobrze, gdy stwierdzisz, że coś można zrobić inaczej (lepiej), niż zrobiłem to ja. Oznaczać to bę-

dzie, że jesteś zaawansowanym programistą. Poprowadzi Cię od początku, poprzez składnię języka, konstrukcję wyrażeń, aż do przykładowych aplikacji. Przykłady będą poparte rysunkami, zgodnie z chińskim powiedzeniem, że „jeden obraz wart jest tysiąca słów". Będą się one skupiały nie tyle na różnych sztuczkach programowych, ile na tym, aby były czytelne i zrozumiałe. Jako autor mam nadzieję, że pod koniec lektury tej książki napiszesz być może pierwszy samodzielny program w języku C.

W przykładach programowania zawartych w tej książce, wykorzystałem kompilator RC-51 firmy Raisonance. Dlaczego akurat ten? Są dwa zasadnicze powody: po pierwsze – wersję demonstracyjną możesz sobie pobrać za darmo z Internetu (*http://www.raisonance.com*), po drugie – wersja demonstracyjna umożliwia skompilowanie programu o kodzie wynikowym nie przekraczającym 4 kB. To naprawdę bardzo dużo. Gdy już poznasz mechanizmy języka C, wybierzesz sobie dowolny kompilator taki, jak będzie Ci odpowiadał i na jaki będzie pozwalać zasobność Twojej kieszeni. Możesz również wykorzystywać dostępne darmowe kompilatory języka C, np. GCC.

Starałem się, aby ta książka była inna niż podobne pozycje tego typu. Nie będziemy uczyć się teorii programowania – kurs zawarty w książce opiera się na analizie praktycznych przykładów. Rozpoczniemy od bardzo prostych, które być może nikomu i niczemu nie posłużą. Na początek podłączymy do mikrokontrolera diodę LED, wyświetlacz 7-segmentowy, nauczymy się multipleksować wyświetlanie i wyświetlać różne informacje. Później poznamy wyświetlacz LCD, obsługę klawiatury i różne zagadnienia związane z tzw. interfejsem użytkownika. Zbudujemy przykładowe menu, nauczymy się je wyświetlać oraz dokonywać wyborów. W międzyczasie postaram się przemycić nieco informacji na temat zmiennych, wskaźników, tablic. Moim zamiarem jest przedstawienie nieco innego podejścia do zagadnień związanych z programowaniem aniżeli formy suchego wykładu. Momentami nie unikniemy rozważań teoretycznych, ale... Książek dotyczących teorii jest bardzo dużo, możesz czerpać pełnymi garściami. Ja będę opowiadał o swojej przygodzie związanej z programowaniem i może nie wytłumaczę w tej książce wszystkich zagadnień związanych ze specyfikacją standardu ANSI C, ale z całą pewnością pokażę Ci jak napisać program.

W tym miejscu chciałbym podziękować Piotrowi Zbysińskiemu. Bez niego i jego inspiracji, ta książka nigdy by nie powstała. To jego słowa i jego wiara we mnie spowodowały, że usiadłem i zacząłem pisać. Dziękuję Ci Piotrze.

Jacek Bogusz

Pliki z programami źródłowymi przykładów z książki są dostępne w Internecie pod adresem: *http://www.btc.pl/pliki/pmc.zip*.

Dlaczego piszę programy w języku C?

Język C został rozwinięty przez firmę Bell Labs we wczesnych latach 70. XX wieku. Ideą przyświecającą autorom tego języka było stworzenie języka wysokiego poziomu, umożliwiającego napisanie systemu operacyjnego UNIX. Zadanie było trudne – do tego momentu oprogramowanie systemowe było pisane w asemblerze mikroprocesora, dla którego było przeznaczone. Oprogramowanie to było pisane dla konkretnego sprzętu i otoczenia – nie dawało się łatwo przenosić do innych warunków pracy. Ponadto tworzenie aplikacji w asemblerze, mimo iż programy w nim napisane są bardzo szybkie i efektywnie wykorzystują dostępne zasoby, jest żmudne i czasochłonne. Programy w języku asemblera są bardzo trudne w analizie i wyszukiwaniu błędów. I tutaj maleńka dygresja: mnóstwo programistów piszących w asemblerze nie docenia czasu, który straci na uruchomienie programu. Z mojego doświadczenia wynika, że analiza i weryfikowanie działania programu napisanego w asemblerze pochłania wielokrotnie więcej czasu, niż czas poświęcony na napisanie danego kodu. Wróćmy jednak do języka C. Projektanci systemu UNIX uznali, że będzie im potrzebny język programowania wysokiego poziomu, pozwalający równie efektywnie wykorzystywać zasoby sprzętowe, jak asembler. Efektem tych prac był język C, który powstał na podstawie języków Algol i BCPL.

Język C jest językiem bardzo wydajnym, dzięki swojej wszechstronności i szybkości działania napisanych w nim programów. Jego filozofią jest: *zaufaj programistom*, *oni wiedzą co robią*. Przykładem takiego podejścia może być to, że zmienne nie są ściśle przywiązane do typów. Gdy na przykład deklarujesz zmienną liczbową, a później zdecydujesz się na umieszczenie w niej znaku, to kompilator C na to pozwoli (w większości języków programowania konieczna byłaby jawna konwersja typów). I chociaż łatwo jest z powodu takiego podejścia do zmiennych zgubić wątek w programowaniu, to jednocześnie jest to powodem, dla którego język ten jest elastycznym narzędziem i doskonale nadaje się do stosowania w świecie mikrokontrolerów, gdzie należy bardzo oszczędnie gospodarować pamięcią. Kompilator nie sprawdza za programistę poprawności typów danych, nie ogranicza jego swobody, ale uwaga – to jest zaleta i jednocześnie pułapka i to nie tylko na początkujących programistów. Uważam jednak, że bardzo będziesz sobie cenił tę cechę języka.

Inną kapitalną cechą tego języka są wskaźniki i bezpośrednie operacje na zmiennych i pamięci. Jeśli poznasz dobrze te mechanizmy, będziesz w stanie pisać bardzo szybkie i efektywnie działające programy przy jednoczesnym minimalnym wręcz wykorzystaniu pamięci. Tego nie da Ci żaden inny język programowania.

Warto jeszcze dodać, że współczesne kompilatory tak dobrze optymalizują kod źródłowy, że jest on niewiele większy od tego, który zostanie napisany w asemblerze. Powszechnie uważa się, że jeśli program napisany w asemblerze zajmuje na przykład 4 kB, to program napisany w C zajmie o 100% więcej, to jest około 8 kB. Jednak z moich doświadczeń wynika coś zupełnie innego. Czasami udaje mi się napisać program, którego kod wynikowy, przez to że nie obawiam się bezpośrednich działań na pamięci (w asemblerze trudno jest nad nimi zapanować), jest mniejszy niż funkcjonalny ekwiwalent w asemblerze! Ale to są sporadyczne przypadki.