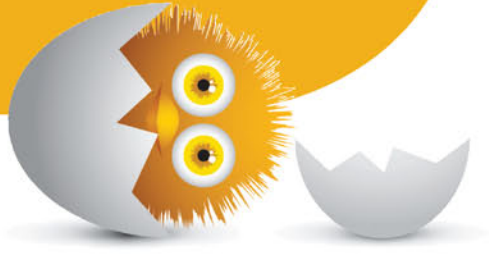
Helion

Język C

PROGRAMOWANIE DLA POCZĄTKUJĄCYCH



Przewodnik dla adeptów programowania!

Wydanie III

Greg Perry, Dean Miller

que

Tytuł oryginału: C Programming Absolute Beginner's Guide, Third Edition

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-1641-6

Authorized translation from the English language edition, entitled: C PROGRAMMING ABSOLUTE BEGINNER'S GUIDE, Third Edition; ISBN 0789751984; by Greg Perry; and by Dean Miller, published by Pearson Education, Inc, publishing as QUE Publishing.

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/jcprpo
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

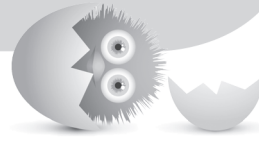
Printed in Poland.

Spis treści

W TYM ROZDZIALE

- Wpisanie pierwszego programu do edytora.
- Sposób użycia funkcji `main()`.
- Rodzaje danych.

2



PIERWSZY PROGRAM W JĘZYKU C

W tym rozdziale zobaczysz pierwszy program w języku C! Ale na razie nie próbuj zrozumieć *każdego* znaku w przedstawianym kodzie źródłowym. Rozluźnij się i postaraj się tylko oswoić z wyglądem i stylem kodu. Wkrótce zaczniesz rozpoznawać poszczególne elementy programów w języku C.

Prosty przykład kodu

W tym podrozdziale przedstawiamy krótki, ale kompletny program w języku C oraz opisujemy inny program, który w całości znajduje się w dodatku B. Oba te programy mają pewne cechy wspólne. Oto pierwszy z nich:

```
/* Drukuje napis na ekranie */
#include <stdio.h>
main()
{
    system("chcp 65001");
    printf("Niewielki krok dla kodera. Gigantyczny skok dla ");
    printf(" programistów!\n");
    return 0;
}
```

Uruchom swoje środowisko programistyczne i wpisz do niego powyższy program. Proste, prawda? Pewnie już używałeś swojego nowego kompilatora. Przy pierwszym uruchomieniu Code::Blocks na ekranie pojawia się podpowiedź dnia. Później może się do czegoś przyda, a na razie zamknij ją, klikając przycisk *Close* (zamknij).

Aby utworzyć program, otwórz menu *File* (plik) i kliknij pozycję *New* (nowy). Z listy opcji w wyświetlonym podmenu wybierz *Empty File* (pusty plik), by utworzyć nowy pusty plik źródłowy, w którym możesz wpisać swój program.

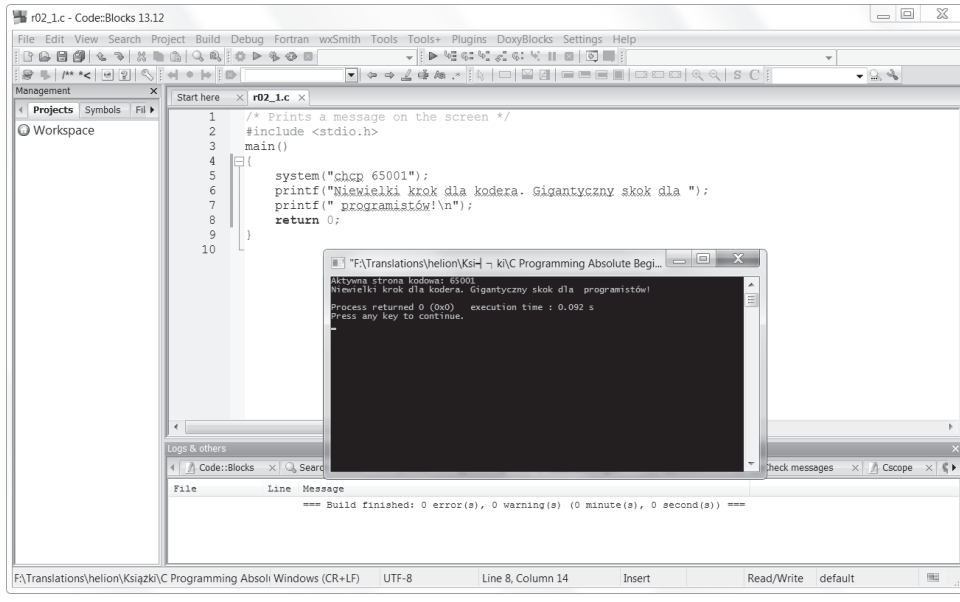
Gdy wpiszesz kod do edytora, musisz go skompilować. W tym celu kliknij znajdującą się na pasku narzędzi małą żółtą ikonę przedstawiającą koło zębate. Jeśli przy wpisywaniu nie popełniłeś żadnego błędu, możesz uruchomić ten program, klikając zieloną strzałkę w prawo znajdującą się obok koła zębatego. (Następna ikona w tym rzędzie, przedstawiająca koło zębate i strzałkę, reprezentuje opcję kompilacji i uruchamiania programu jednocześnie, co pozwala na zmniejszenie liczby kliknięć, jakie musisz wykonać, aby uruchomić program).

Po skompilowaniu i uruchomieniu programu na ekranie powinno pojawić się okno widoczne na rysunku 2.1.

UWAGA Wydrukowanie tego krótkiego napisu w konsoli wymagało sporo pracy! Ale tak naprawdę tylko dwie linijki z powyższego programu rzeczywiście wykonują pracę związaną z tworzeniem danych wyjściowych — te zaczynające się od słowa `printf`. Pozostały kod służy tylko do przeprowadzenia pewnych rutynowych czynności, które wykonuje się we wszystkich programach w języku C.

Jeśli chcesz zobaczyć dłuższy program, zajrzyj do dodatku B. Choć wydrukowana w nim gra w pokera obejmuje kilka stron, znajdują się w niej takie same podstawowe elementy jak w tym krótszym programie.

Przyjrzyj się obu opisanym programom i zwróć uwagę na łączące je podobieństwa. Pierwsze, co może Ci się rzucić w oczy, to klamry ({}), nawiasy (()) i ukośniki (\). Kod źródłowy należy wpisywać do edytora bardzo starannie, ponieważ kompilator języka C jest bardzo drobiazgowy. Nie możesz na przykład wpisać nawiasu prostokątnego ([) w miejscu, w którym powinna być klamra.



RYSUNEK 2.1.

Wynik działania pierwszego programu

OSTRZEŻENIE Szczególną ostrożność zachowaj też przy kopiowaniu kodu do edytora z procesora tekstu. Przedstawiony program napisałem w Wordzie (w ramach tej książki), a potem skopiowałem go do Code::Blocks. Ale podczas kompilacji okazało się, że kod zawiera błędy, ponieważ Word zmienił cudzysłowy w linijkach z instrukcją `printf` (zamiast prostych wstawił zagięte), przez co kompilator ich nie rozpoznał. Gdy usunąłem te cudzysłowy i wpisałem je jeszcze raz już bezpośrednio w edytorze, kompilacja przebiegła pomyślnie. Jeśli więc w Twoim kodzie pojawią się jakieś błędy, sprawdź, czy nie mają przypadkiem związku z cudzysłowami.

Kompilator C nie jest wybredny pod każdym względem. Na przykład większość odstępów, jakie są obecne w programach, ma zwiększać czytelność kodu dla ludzi, ale nie dla kompilatora. Pisząc program, dodawaj puste wiersze i wcinaj sekcje kodu, aby poprawić jego wygląd i ułatwić sobie znajdowanie w nim wybranych części.

WSKAZÓWKA Do wcinania kodu lepiej jest używać tabulatorów niż spacji. W większości edytorów języka C da się ustawić *szerokość tabulatora* (liczbę spacji). Niektóre linijki kodu C są dość długie, dobrym pomysłem jest więc ustawienie tabulatora na trzy spacje. To daje dobre efekty i nie powoduje nadmiernego wydłużania linijek.

Wszystkie polecenia i funkcje standardowe w języku C pisze się małymi literami. (O tym, czym jest funkcja, dowiesz się w następnym podrozdziale). Wielkich liter używa się tylko w wierszach z dyrektywą `#define` i w napisanych do wydrukowania na ekranie.

Funkcja main()

Najważniejszą częścią programu w języku C jest funkcja **`main()`**. Oba opisane wcześniej programy zawierają funkcję `main()`. Wprawdzie w tej chwili nie ma to wielkiego znaczenia, ale warto zaznaczyć, że `main()` to **funkcja**, a nie **polecenie**. Funkcja to procedura wykonująca pewne zadanie. Niektóre funkcje są standardowo dostępne w języku C, a inne programista tworzy samodzielnie. Programy w języku C składają się z jednej lub większej liczby funkcji, ale każdy program *musi* zawierać przynajmniej funkcję `main()`. Od poleceń funkcje różnią się nawiasem za nazwą. Poniżej znajdują się przykłady funkcji:

```
main() calcIt() printf() strlen()
```

A to są przykładowe polecenia:

```
return while int if float
```

W innych książkach o programowaniu w języku C, podręcznikach i na stronach internetowych za nazwami funkcji może nie być nawiasów. Na przykład autor może pisać o funkcji `printf`, a nie `printf()`. Szybko nauczysz się rozpoznawać nazwy funkcji, dlatego nie musisz się tym przejmować. Z drugiej strony większość autorów stara się jak najwyraźniej rozróżniać funkcje i inne konstrukcje, najczęściej więc dodają te nawiasy.

OSTRZEŻENIE W nazwie jednej z przedstawionych na liście funkcji, `calcIt()`, znajduje się wielka litera, mimo że wcześniej napisaliśmy, że tak nie powinno się zdarzyć. Jeśli nazwa składa się z kilku słów, np. `wydrukujTenRaport()`, to zazwyczaj każde słowo oprócz pierwszego pisze się wielką literą w celu zwiększenia czytelności kodu. (W nazwach funkcji nie może być spacji). Podsumowując, nie używaj wielkich liter zawsze, tylko od czasu do czasu.

Funkcja `main()` i wszystkie standardowe funkcje języka C muszą być zapisywane małymi literami. Wielkich liter można używać w nazwach własnych funkcji, ale większość programistów i tak nie korzysta z tej możliwości.

Podobnie jak strona główna stanowi miejsce, od którego zaczyna się przeglądanie witryny internetowej, tak funkcja `main()` jest początkiem wykonywania każdego programu. Jest tak nawet wtedy, gdy przed nią znajdują się jeszcze jakieś inne funkcje. W związku z tym dla czytelności najlepiej jest, aby funkcja ta znajdowała się na początku programu. Programy opisane w kilku następnych rozdziałach zawierają tylko jedną funkcję — `main()`. Gdy zdobędziesz trochę praktyki, dowiesz się, dlaczego dobrze jest dodawać funkcje za funkcją `main()`.W rozdziale 30. natomiast nauczysz się pisać własne funkcje.

Za napisem `main()` zawsze znajduje się otwarcie klamry ({) wyznaczające początek treści funkcji oraz zamknięcie (}) wyznaczające jej koniec. Między tymi znakami mogą znajdować się inne pary klamer. Jeszcze raz spójrz na program w dodatku B. Jego pierwsza funkcja to `main()`, a wewnątrz niej znajdują się inne z własnymi klamrami.

UWAGA W prawie każdym programie C potrzebna jest instrukcja `#include <stdio.h>`, która pomaga w drukowaniu i pobieraniu danych. Na razie zapamiętaj tylko, aby wstawiać ją zawsze gdzieś przed funkcją `main()`. Zastosowanie dyrektywy `#include` i jej znaczenie w programach dokładnie poznasz w rozdziale 7.

Rodzaje danych

Programy C wykorzystują do działania dane składające się z liczb, znaków i słów oraz przetwarzają je w przydatne informacje. Choć istnieje wiele różnych rodzajów danych, poniższe trzy są używane najczęściej w typowych programach w języku C:

- znaki,
- liczby całkowite,
- liczby zmiennoprzecinkowe (w uproszczeniu zwane też **rzeczywistymi**).

WSKAZÓWKA Może z przerażeniem zastanawiasz się, ile matematyki będziesz musiał się nauczyć. Pewnie nie spodziewałeś się takiego obrotu rzeczy. Ale możesz się uspokoić, ponieważ do programowania w języku C nie potrzeba znajomości matematyki. Nie musisz nawet wiedzieć, ile to jest dwa plus dwa. Trzeba jednak rozróżniać typy danych, aby móc bez problemu zdecydować, który w razie potrzeby zastosować.

Znaki w języku C

Znak w języku C to każdy pojedynczy znak, jaki komputer może reprezentować. Twój komputer zna 256 znaków zapisanych w tzw. **tabeli ASCII** (dodatek A). Wszystko, co komputer może reprezentować, może być znakiem. Na przykład wszystkie poniższe elementy są znakami:

```
A a 4 % Q ! + = ]
```

UWAGA Organizacja American National Standards Institute (ANSI), która stworzyła standard ANSI C, jest też autorem kodu karty ASCII.

WSKAZÓWKA Nawet spacja jest znakiem. Kompilator C rejestruje wszystkie litery, cyfry i inne znaki, w tym również wszelkie odstępy, takie jak spacje.

Jak widać, każda litera, cyfra i spacja to w języku C jakiś znak. Oczywiście `4` wygląda jak liczba i czasami pełni taką właśnie funkcję, ale oprócz tego jest też znakiem. Jeśli zaznaczysz, że `4` jest znakiem, nie możesz używać jej w działaniach matematycznych. To samo dotyczy symboli specjalnych. Plus (+) jest znakiem, ale może też być używany jako operator dodawania. (I znów matematyka wraca jak bumerang)!

Dane znakowe w języku C umieszcza się między apostrofami (`'`), przez niektórych zwanymi pojedynczym cudzysłowem. Pozwala to odróżnić znaki od innych rodzajów danych, takich jak liczby i symbole matematyczne. Poniżej znajduje się kilka przykładów danych znakowych języka C:

```
'A' 'a' '4' '%' ' ' '-'
```

Z kolei w następnym przykładzie nie ma danych znakowych, ponieważ nie użyto apostrofów:

```
A a 4 % -
```

WSKAZÓWKA Poniższe przykłady nie są prawidłowymi znakami, ponieważ w apostrofach można umieszczać tylko pojedyncze znaki, nie ich ciągi.

```
'Język C jest fajny.'
'Język C jest trudny.'
'Powinienem teraz żeglować!'
```

Pierwszy opisany w tym rozdziale program zawiera też znak `'\n'`. Na pierwszy rzut oka widać, że `\n` to nie pojedynczy znak, ale jest to jedna z kilku dwuznakowych kombinacji interpretowanych przez kompilator C jako pojedyncze znaki. Dokładniejsze wyjaśnienie tego znajduje się dalej.

Jeśli chcesz użyć więcej niż jednego znaku (nie licząc opisanego powyżej przypadku specjalnych kombinacji dwuznakowych), użyj prostego cudzysłowu podwójnego (`"`). Ciąg znaków nazywa się **łańcuchem**. Oto przykład prawidłowego łańcucha w języku C:

```
"Fajnie jest uczyć się języka C."
```

UWAGA Na razie wystarczy wiedzieć tylko tyle o znakach i łańcuchach znaków. W rozdziałach 4. – 6. nauczysz się ich używać w programach. Kiedy będziesz już potrafił zapisywać znaki w zmiennych, dostrzeżesz też, dlaczego apostrofy i cudzysłowy podwójne są takie ważne.

Liczby w języku C

Choć pewnie nigdy do tej pory się nad tym nie zastanawiałeś, liczby mogą mieć różne rozmiary i formy. Programista musi mieć możliwość zapisywania liczb w programie bez względu na to, jak wyglądają. Służą do tego zmienne typów liczbowych. Zanim przejdziemy do zmiennych, zrobimy krótkie przypomnienie wiadomości o rodzajach liczb.

Liczby całkowite to takie, które nie mają części ułamkowej. W związku z tym każda liczba bez przecinka, a w programie bez kropki, to liczba całkowita. Oto kilka przykładów takich liczb:

```
10 54 0 –121 –68 752
```

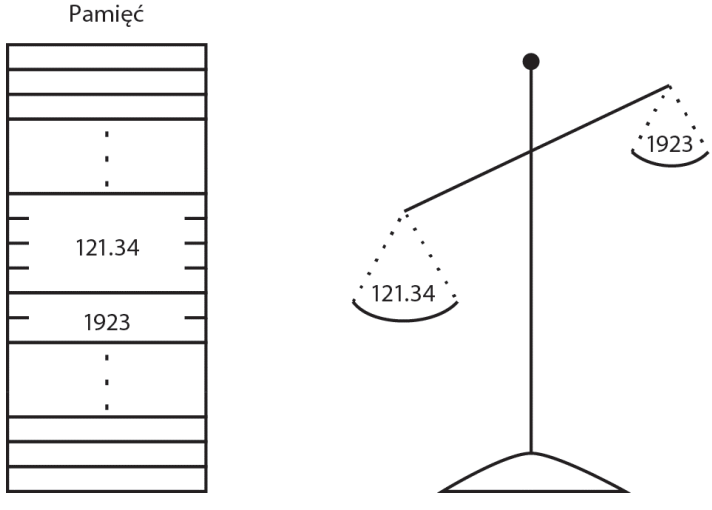
OSTRZEŻENIE Liczba całkowita nie powinna zaczynać się zerem (chyba że *jest* po prostu zerem), ponieważ wówczas kompilator może ją potraktować jako liczbę **szesnastkową** lub **ósemkową**. Liczby szesnastkowe i ósemkowe (nazywane też liczbami o podstawie szesnaście i osiem) to zwykłe liczby, tylko przedstawione w dziwaczny sposób. Na przykład `053` to liczba ósemkowa, a `0x45` to liczba szesnastkowa. Jeśli nie wiesz, o co chodzi, to zapamiętaj tylko, że jeśli będziesz stawiać zero na początku liczb, może Cię spotkać nieszczęście.

Liczby z kropką dziesiętną to **liczby zmiennoprzecinkowe**. Poniżej znajduje się kilka przykładów:

```
547.43  0.0  0.44384  9.1923  –168.470  .22
```

WSKAZÓWKA Jak widać, zero na początku liczb zmiennoprzecinkowych nie powoduje problemów.

Wybór liczb zmiennoprzecinkowych lub całkowitych zależy od rodzaju danych, na których pracuje program. Niektóre wielkości (np. wiek czy ilość) można przedstawić za pomocą liczb całkowitych, a do wyrażenia innych (np. kwot pieniędzy i ciężaru) potrzebne są liczby zmiennoprzecinkowe. Wewnętrzna reprezentacja liczb całkowitych różni się od reprezentacji liczb zmiennoprzecinkowych. Jak widać na rysunku 2.2, wartość zmiennoprzecinkowa z reguły zajmuje dwa razy więcej pamięci niż liczba całkowita. Dlatego jeśli nie jest to konieczne, lepiej jest nie używać wartości zmiennoprzecinkowych.



RYSUNEK 2.2.

Do przechowywania wartości zmiennoprzecinkowych często potrzeba więcej pamięci niż do zapisu liczb całkowitych

UWAGA Z rysunku 2.2 wynika, że liczby całkowite z reguły zajmują mniej pamięci niż liczby zmiennoprzecinkowe bez względu na rozmiar reprezentowanej przez nie wartości. Jednego dnia urząd pocztowy może otrzymać znacznie więcej listów niż innego. Zawartość skrzynki nie ma jednak wpływu na jej pojemność. Podobnie jest z typami liczbowymi w języku C — ich pojemność nie jest uzależniona od wartości liczby.

Różne kompilatory języka C wykorzystują różne ilości pamięci do przechowywania liczb całkowitych i zmiennoprzecinkowych. Później pokażemy Ci, jak sprawdzić, ile dokładnie pamięci przeznacza dany kompilator dla każdego typu danych.

Jeszcze jeden przykład w ramach podsumowania

Celem tego rozdziału było pokazanie, jak wygląda program w języku C, a w szczególności przedstawienie funkcji `main()` jako głównej procedury zawierającej wykonywalne instrukcje. Pokazaliśmy, że język C jest bardzo liberalny pod względem stosowania odstępów w kodzie źródłowym, ale bardzo restrykcyjny, jeśli chodzi o stosowanie wielkich i małych liter w nazwach konstrukcji programistycznych. Małymi literami zapisuje się nazwy wszystkich standardowych poleceń i funkcji języka C, np. `printf()`.

Na razie nie przejmuj się zbytnio szczegółami przedstawionego w tym rozdziale kodu, ponieważ wszystko jest dokładnie wyjaśnione w następnych rozdziałach. Ale bardzo dobrym pomysłem jest przepisanie i przeanalizowanie jak największej liczby programów — dzięki takiej praktyce zdobywa się pewność siebie! Dlatego poniżej przedstawiamy jeszcze jeden program, w którym użyto opisanych wcześniej typów danych:

```
/* Program, w którym wykorzystano znaki, liczby całkowite i liczby zmiennoprzecinkowe */
#include <stdio.h>
main()
{
    system("chcp 65001");
    printf("Uczę się języka programowania %c.\n", 'C');
    printf("Właśnie skończyłem czytać rozdział %d.\n", 2);
    printf("Jestem na %.1f procent gotów do dalszej pracy ", 99.9);
    printf("w następnym rozdziale!\n");
    return 0;
}
```

Program ten tylko drukuje na ekranie trzy napisy, z których każdy zawiera jeden z opisanych wcześniej typów danych: znak (`C`), liczbę całkowitą (`2`) oraz liczbę zmiennoprzecinkową (`99.9`).

UWAGA W pierwszej instrukcji `printf` ciąg `%c` wskazuje programowi miejsce, w którym należy wstawić znak `'C'`. Litera `c` w tym ciągu to skrót od angielskiego wyrazu *character* (znak), a nie odpowiednik `C`. Gdyby to była książka o języku programowania N, to i tak użylibyśmy ciągu `%c`, aby wstawić w wybranym miejscu literę `'N'`.

Funkcja `main()` to jedyna funkcja w tym programie napisana przez programistę. Jej treść musi być objęta klamrą ({}). To samo dotyczy także innych funkcji, jeśli zostaną dodane do programu. Ponadto w kodzie użyto standardowej funkcji języka C o nazwie `printf()`. Poniżej znajduje się wynik jej działania:

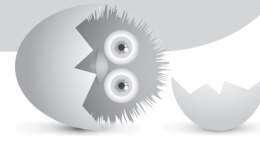
```
Uczę się języka programowania C.
Właśnie skończyłem czytać rozdział 2.
Jestem na 99.9 procent gotów do dalszej pracy w następnym rozdziale!
```

WSKAZÓWKA Pobaw się tym programem, pozmieniaj coś w wyświetlanych napisach. Sprawdź też, co się stanie, gdy popełnisz literówkę, np. zapomnisz średnika na końcu linijki. Dobrze jest wiedzieć, co się dzieje w takim przypadku podczas kompilacji. Nauka na błędach jest bardzo efektywna.

ABSOLUTNE MINIMUM

W tym rozdziale skompilowałeś i uruchomiłeś pierwszy program w języku C i zapoznałeś się z funkcją `main()`. Oto lista najważniejszych informacji do zapamiętania:

- W języku C za nazwą funkcji musi znajdować się nawias. Program C składa się z przynajmniej jednej funkcji. Funkcja `main()` jest zawsze obowiązkowa i jest wykonywana jako pierwsza.
- Stosuj dużo odstępów w kodzie programu, aby uczynić go jak najbardziej czytelnym.
- Na początku liczb całkowitych nie wpisuj zera, chyba że chcesz użyć właśnie zera.
- Pojedyncze znaki umieszczaj między apostrofami. Łańcuchy wpisuje się w podwójnych prostych cudzysłowach. Liczby całkowite to liczby pozbawione części ułamkowej. Liczby zmiennoprzecinkowe mają część ułamkową.



Skorowidz

U

W

Z

Odkryj pasjonujący świat programowania!

Język C ujrzał światło dzienne w 1972 roku i pomimo zaawansowanego wieku wciąż jest powszechnie używany. Na rynku istnieje oczywiście wiele innych języków, zazwyczaj łatwiejszych dla programistów, jednak C wciąż nie ma sobie równych w wielu zastosowaniach. Wszędzie tam, gdzie wymagana jest bezpośrednia kontrola nad sprzętem, najwyższa wydajność oraz przewidywalność czasu wykonania, C jest najczęściej wybieranym rozwiązaniem. Co więcej, jeżeli poznasz ten język, nauka kolejnych nie będzie stanowiła dla Ciebie żadnego problemu!

Dołącz do świata prawdziwych programistów – jeśli przeczytasz tę książkę, zrobisz spory krok w tym kierunku. Znajdziesz tu informacje na temat kluczowych elementów języka C: zmiennych, pętli, instrukcji warunkowych. To podstawowe konstrukcje, które pozwolą Ci napisać Twój pierwszy program. Następnie przejdziesz do poznawania bardziej zaawansowanych zagadnień, takich jak tablice, wskaźniki oraz operacje na plikach. Na sam koniec dowiesz się, jak optymalnie wykorzystać dostępną pamięć, alokować i zwalniać miejsce na stercie oraz przechowywać dane w strukturach. Najnowsze wydanie książki zostało wzbogacone i zaktualizowane o informacje na temat nowego standardu C11. To pozycja obowiązkowa dla każdego adepta programowania.

Dzięki tej książce:

- poznasz składnię języka C
- zadeklarujesz zmienne odpowiedniego typu
- nawiążesz interakcję z użytkownikiem
- podejmiesz decyzję o sposobie wykonania programu
- wykorzystasz struktury do przechowywania danych
- zaczniesz swoją przygodę w świecie programistów

Greg Perry – programista, trener, konsultant. Doświadczony nauczyciel, który wprowadził tysiące osób w świat programowania. Autor 75 książek poświęconych programowaniu (ich łączny nakład przekroczył 2 miliony egzemplarzy). **Dean Miller** – pisarz i redaktor z ponad dwudziestoletnim doświadczeniem w branży wydawniczej. W trakcie swojej kariery wydał wiele bestsellerów, wśród nich całą serię „Teach Yourself in 21 Days", „Teach Yourself in 24 Hours" oraz „Unleashed". Wiele z nich zostało wydanych przez Helion.

Helion

37366 numer katalogowy

księgarnia internetowa
http://helion.pl

zamówienia telefoniczne
0 801 339900
0 601 339900

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

que

ISBN 978-83-283-1641-6

Informatyka w najlepszym wydaniu

cena: 49,00 zł