

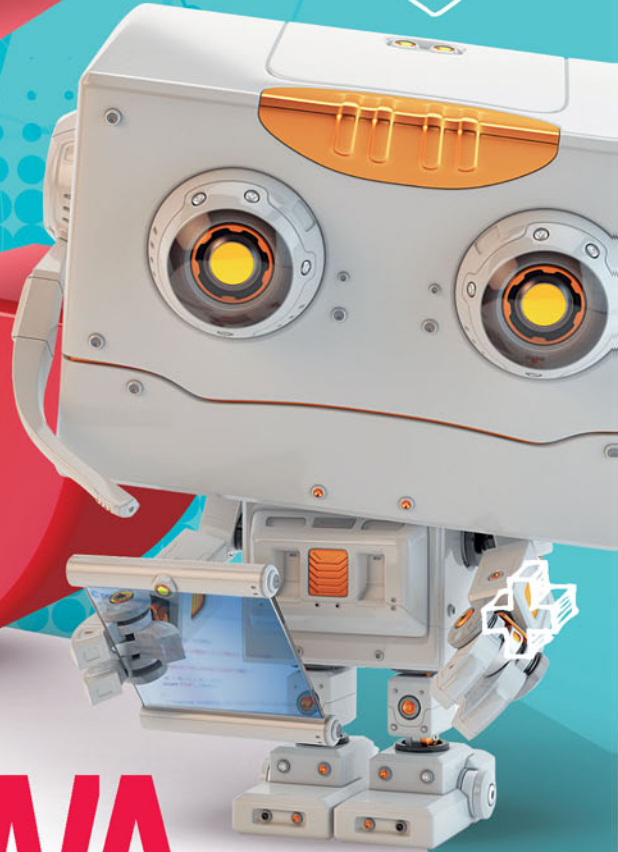
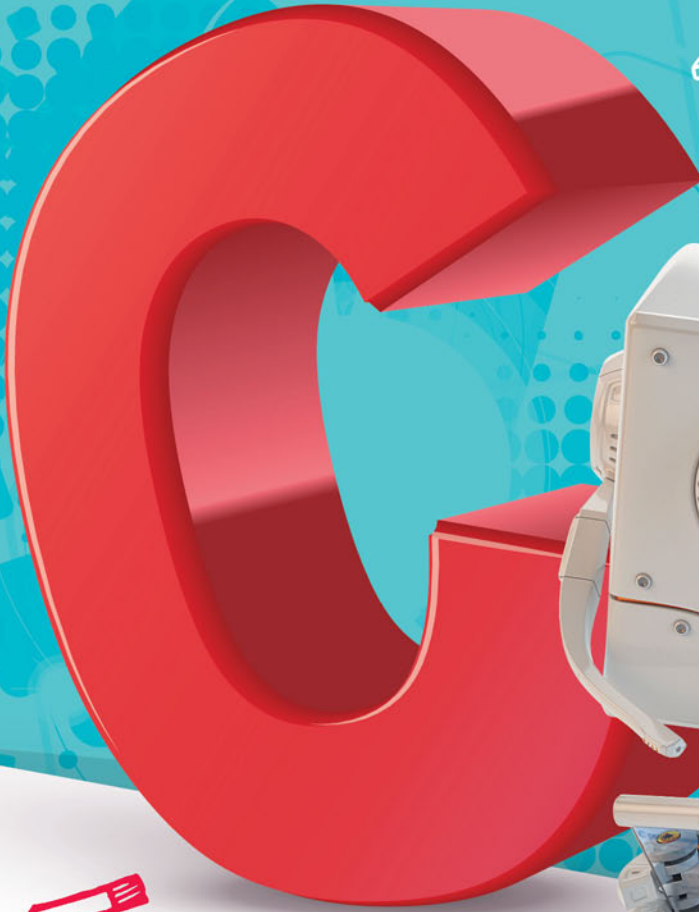
Michał Wiszniewski

NAPISZ SOBIE SWÓJ WŁASNY PROGRAM!

Zanim rozpoczniesz prace, czyli jak zainstalować odpowiednie programy

Bez tych klocków ani rusz, czyli co składa się na język C i jak tego używać

Im dalej w las, czyli jak stosować bardziej zaawansowane mechanizmy



ZABAWA w programowanie

JEZYK C DLA NASTOLATKÓW



Helion

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.

Redaktor prowadzący: Michał Mrowiec

Projekt okładki: Jan Paluch

Fotografia na okładce została wykorzystana za zgodą Shutterstock.com

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/zaprcn>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-1491-7

Copyright © Helion 2016

Printed in Poland.

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

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

Spis treści

Rozdział 1. O co tu chodzi? — opanujesz to w tydzień :)	5
Rozdział 2. Rozpoczęcie pracy — zainstaluj odpowiednie programy	7
Rozdział 3. Pierwszy program — już programujesz!	13
Rozdział 4. Biblioteki i funkcje — jak to działa?	19
Rozdział 5. Zmienne i stałe — niezbędne elementy	23
Rozdział 6. Trochę więcej o funkcjach i zmiennych — zaczynamy rozwijać skrzydła	27
Rozdział 7. Tablice — bardzo przydatna sprawa	33
Rozdział 8. Instrukcje warunkowe — ważne i proste	37
Rozdział 9. Operatory — tak naprawdę znasz to już od podstawówki	41
Rozdział 10. Podstawy logiki. Nuda? Wcale nie!	45
Rozdział 11. Pętle. Po co to? Po co to? Po co to?	51
Rozdział 12. Operacje wejścia i wyjścia oraz podstawowa obsługa błędów. Nareszcie!	57
Rozdział 13. enum i typedef — przydatne bajery	63
Rozdział 14. Struktury i unie. Liczyłem, że coś takiego wymyślono	67
Rozdział 15. Wskaźniki — pora na małe wyzwanie	71
Rozdział 16. malloc i free oraz stos i sarta, czyli kilka słów o pamięci operacyjnej	75

Rozdział 17. Operacje na plikach i parametry wejściowe programu	
— zawsze chciałem to zrobić	79
Rozdział 18. Preprocesor, kompilator i linker — to tylko groźnie brzmi	83
Rozdział 19. Pliki nagłówkowe oraz static i extern,	
czyli jak dzielić program i dlaczego?	89
Rozdział 20. Na koniec: to dopiero początek wspaniałej przygody! :)	95
Skorowidz	97

Rozdział 3.

Pierwszy program — już programujesz!

Masz już wszystko, co jest potrzebne do programowania w języku C, więc pora zaczynać!

Uruchom edytor tekstu:

1. Naciśnij ikonę w lewym górnym rogu ekranu, tuż pod napisem *Ubuntu Desktop*. Po pojawieniu się nowego okna wpisz słowo `gedit`, a następnie poniżej wybierz lewym przyciskiem myszy program o nazwie Text Editor.
2. W programie `gedit` wybierz *File*, a następnie *Save As...* Teraz zapisz plik w katalogu `/home/user` pod nazwą `program031.c`
3. W treści pliku wpisz kod z listingu 3.1.

Listing 3.1. Plik `program031.c`. Pierwszy program

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     printf("Wlasnie zostales programista!\n");
6.     return 0;
7. }
```

Nie zapominaj, aby za każdym razem, gdy w programie wprowadzisz jakiegokolwiek zmianę, zapisywać plik poprzez naciśnięcie przycisku *Save*.

Właśnie napisałeś swój własny program w języku C. Niestety komputer nie wie, co z nim zrobić. Język C jest zrozumiały dla ludzi, ale nie dla maszyn. To tak jak np. Francuz nie porozumie się z Polakiem bez znajomości choćby jednego wspólnego języka. Potrzebujemy tłumacza i takiego tłumacza posiadamy. Nazywa się on kompilatorem, a wspomagają go dodatkowo preprocesor i linker. Jeszcze nie musisz rozumieć tych nazw. Poznasz je na końcu książki. Teraz musisz jedynie wiedzieć, jak poprosić swego „tłumacza” o pomoc.



Uwaga

Pamiętaj, że żaden znaczek nie jest tu przypadkowy. Żadna kropka, średnik czy spacja. Komputer jest urządzeniem niezwykle precyzyjnym, więc programując, musisz być bardzo ostrożny. Zauważ również, że w kodzie programu nie używamy polskich liter, ponieważ czasami zamiast nich możemy otrzymać dziwne znaczki. Z tego powodu będziemy trzymać się tej zasady aż do końca książki. Pamiętaj również, że wszystkie programy w tej książce zostały przeze mnie przetestowane i na 100% działają. Jeżeli u Ciebie występuje jakiś błąd, to przyczyną prawdopodobnie jest niedokładnie przepisany program.

W tym celu:

1. Naciśnij ikonę w lewym górnym rogu ekranu, tuż pod napisem *Ubuntu Desktop*. Po pojawieniu się nowego okna wpisz słowo `terminal`, a następnie poniżej wybierz lewym przyciskiem myszy program o nazwie Terminal.
2. W Terminalu wpisz poniższą komendę:


```
cd /home/user
```

 a następnie wciśnij *Enter*, aby wykonać tę komendę.
3. Wpisz w Terminalu komendę `ls`, naciśnij *Enter* i sprawdź, czy znajduje się tam nasz plik `program031.c`. Jeśli go tam nie ma, upewnij się, że wszystkie wcześniejsze komendy wykonałeś poprawnie.
4. Jeśli wszystko przebiegło poprawnie, wpisz w Terminalu:

```
gcc program31.c -o prog
```

i oczywiście naciśnij *Enter*.



Uwaga

Pamiętaj, że możesz zmieniać program `gedit` na Terminal i odwrotnie za pomocą kombinacji klawiszy `Alt+Tab`. W tym celu naciśnij i przytrzymaj klawisz *Alt*, a następnie naciśnij jeden raz lub kilka razy klawisz *Tab*. Powinieneś również wiedzieć, że czasami system Ubuntu zablokuje ekran (jeśli będziesz zbyt długo nieaktywny) i poprosi o wpisanie hasła w celu jego odblokowania. Hasło podawaliśmy przy instalacji systemu i brzmi ono *user*.

Wspaniale! Udało Ci się przetłumaczyć Twój pierwszy program napisany w języku C na język komputera. Twój tłumacz o nazwie `gcc` przetłumaczył plik `program031.c` na zrozumiałą dla komputera wersję o nazwie `prog`. Teraz rozumiesz już znaczenie powyższej linii.



Uwaga

Jeśli zmienisz cokolwiek w swoim programie, to znowu musisz poprosić „tłumacza”, czyli program `gcc`, o ponowne tłumaczenie. To tak, jakby polski kucharz wydał nową wersję swojej książki kucharskiej. Francuz, który ma starą wersję książki, nie dowie się o zmianach, chyba że tłumacz ponownie ją przetłumaczy.

Nasz program jest gotowy do uruchomienia! Aby go odpalić, musimy wpisać w Terminalu:

```
./prog
```

i potwierdzić klawiszem *Enter*.

W tym momencie na ekranie powinna pojawić się następująca informacja:

```
user@computer:~$ ./prog
Wlasnie zostales programista!
user@computer:~$ ~$ ./prog
```

Jak to się stało? Za parę sekund wszystko zrozumiesz. Wy tłumaczę Ci to nie jak inni programiści, lecz jak osoba, która kiedyś próbowała to zrozumieć tak jak Ty.

Przeanalizujemy, jak to zrobiłeś. Spójrz na *program031.c*. Zaczniemy od tej linijki:

```
printf("Wlasnie zostales programista!\n");
```

Słowo `printf` informuje komputer, że chcesz wypisać na ekranie tekst. Wybrany tekst wpisujesz w okrągłym nawiasie i w cudzysłowie, w ten sposób:

```
printf ( "wybrany tekst" );
```

Na końcu stawiamy średnik. W języku C stawiamy średnik na końcu większości linii. Informuje to nasz kompilator, że w tym miejscu kończy się polecenie. Kiedy je przeanalizuje, przejdzie do kolejnych linii.

Znak `"\n"` jest komendą dla komputera, która każe mu przejść do nowej linii. Gdybyśmy usunęli go z naszego programu, efekt byłby mało czytelny:

```
gzp@gzp:~$ ./prog
Wlasnie zostales programista!gzp@gzp:~$ ./prog
```

Niestety komputer nie rozumie sam z siebie słówka `printf`. W tym celu potrzebna jest mu „encyklopedia”, która wytłumaczy mu, co ma zrobić, kiedy wpiszesz to słowo. Taką encyklopedią w naszym programie jest linijka:

```
#include <stdio.h>
```

Informuje ona komputer, że wszystkie potrzebne tłumaczenia znajdzie w „encyklopedii” o nazwie *stdio.h*. Część `#include<nazwa_encyklopedii>` informuje komputer, że wskazujemy na encyklopedię o nazwie *nazwa_encyklopedii*. W informatyce nie używamy jednak słowa „encyklopedia”, lecz „biblioteka”. Natomiast takie słowa jak `printf` nazywamy funkcjami. To znaczy, że funkcja `printf` wypisuje na naszym ekranie oczekiwany tekst, a informację, jak ma to zrobić, odczytuje z biblioteki o nazwie *stdio.h*.

Wyobraź sobie, że nie masz pojęcia o mechanice samochodowej, a kolega prosi, abyś wymienił w samochodzie filtr oleju. Aby dowiedzieć się, jak to zrobić, poszedłbyś po poradę do jakiegoś doświadczonego mechanika samochodowego i wtedy poradziłbyś sobie bez problemu. Biblioteka *stdio.h* jest naszym doświadczonym mechanikiem, a funkcja `printf` to prośba kolegi.

Na koniec zostaje nam jeszcze fragment:

```
int main()
{
    (...)
    return 0;
}
```

Teraz wystarczy, że zapamiętasz, iż w ten sposób będziemy pisali większość naszych programów. W miejscu, gdzie znajduje się (...), będziemy pisali wszystkie instrukcje, które komputer powinien wykonać. Dokładniej wytłumaczę Ci to w późniejszych rozdziałach.

Myślę, że nadszedł dobry moment na parę słów na temat ogólnych zasad programowania. Są to tak zwane dobre praktyki. Zauważyłeś na pewno, że program, który przed chwilą napisałeś, dziwnie wygląda. `int main` oraz nawiasy klamrowe zaczynają się maksymalnie od lewej strony ekranu, a kolejne linijki zaczynają się nieco dalej. Są to tak zwane wcięcia i są one najwycyżajniej w świecie kilkoma spacjami (przeważnie czterema). Programiści piszą w ten sposób, ponieważ bardzo poprawia to czytelność kodu. To samo dotyczy wstawiania w niektórych miejscach czystych linii (w naszym przypadku tak właśnie wygląda linia druga). Jest to bardzo dobry sposób na odseparowanie poszczególnych części programu. Dla komputera wszystkie takie elementy są niewidoczne. Powyższy program mogliśmy napisać równie dobrze w taki sposób:

```
#include <stdio.h>
int main(){printf("Wlasnie zostales programista!\n");return 0;}
```

i zadziałałby on dokładnie tak samo! Musisz jednak przyznać, że jego czytanie byłoby niezbyt przyjemne, a spróbuj wyobrazić sobie program zawierający kilkaset linii kodu! Dla programisty byłaby to prawdziwa katorga! Kolejne dobre praktyki programistyczne będą pojawiać się w kolejnych rozdziałach wraz z nowymi pojęciami.

Na koniec tego rozdziału chciałem wspomnieć o komentarzach w kodzie. Są to specjalne miejsca, gdzie możemy wpisywać, co nam się podoba, a kompilator nie zwróci na to uwagi. Komentarze są przydatne, gdy chcemy zapisać jakąś ważną notatkę niebędącą częścią programu. Aby do programu wstawić komentarz, należy zrobić to w następujący sposób:

```
// tekst komentarza lub /* tekst komentarza */
```

Spójrz na przykład:

```
1. #include <stdio.h>
2. // to jest pierwszy komentarz
3. int main()
4. {
5.     printf("Wlasnie zostales programista!\n");
6.     return 0;
7.     /*
8.     Tutaj wpisuję kolejny komentarz,
9.     który zajmuje kilka linijek.
10.    */
11. }
```


Pierwszy sposób komentowania pozwala nam na wpisanie komentarza od znaku `//` do końca linii. Kolejny sposób pozwala na dużo bardziej rozbudowane komentarze. Wystarczy, że na początku komentarza wpiszemy `/*`, a na jego końcu `*/`. Sprawi to, że wszystko pomiędzy tymi znakami zostanie zignorowane przez kompilator.

Zadania

1. Spróbuj zmienić plik `program31.c` tak, aby komputer wyświetlał każde słowo w nowej linii. Efekt powinien wyglądać tak:

```
user@computer:~$ ./prog
```

```
Wlasnie
```

```
zostales
```

```
programista!
```

```
gzp@gzp:~$
```

Podpowiedź: Znaku `"\n"` możesz używać dowolną liczbę razy.

Skorowidz

B

bezpieczeństwo, 91
biblioteka, 19, 89
 curl, 86
 glib, 86
 math.h, 19, 20
 niestandardowa, 86
 sqlite3, 86
 stdbool.h, 48
 stdio.h, 15
 stdlib.h, 60, 76
 string.h, 35
 tworzenie, 90
błąd, 84, 85
 obsługa, *Patrz:* obsługa błędów

D

danych wyciek, *Patrz:* pamięć wyciek
dekrementacja, 41
dyrektywa
 #define, 83, 91
 #endif, 91
 #error, 83
 #ifdef, 83
 #ifndef, 83, 91
 #include, 20, 83, 91
 #pragma, 84
 #undef, 83
 #warning, 83

F

falsz, 45, 46, 78
funkcja, 19
 argument, 29
 atof, 60

atoi, 60
ciało, 29
definicja, 27
fclose, 80
fgets, 80
fopen, 80
fprintf, 80
free, 77
getchar, 59
gets, 59
ls, 14
main, 29
malloc, 76, 77
nazwa, 31
printf, 15, 20, 57
prototyp, 28
putchar, 57
puts, 57
scanf, 58, 59
sqrt, 19
streat, 35
stcmp, 35
strlen, 35
terminal, 14
tworzenie, 27
wartość, 77

G

gcc, 14
gedit, 11, 13, 14

H

hasło, 10

I

inkrementacja, 41
 instrukcja
 pętli, *Patrz:* pętla
 warunkowa, 37, 40, 45, 83
 if else, 37, 39
 switch, 39

K

komentarz, 16, 93
 kompilator, 13, 84, 85
 komputer wirtualny, 8

L

linker, 13, 86
 Linux, 7
 logika, 45

M

makro, 84

O

obsługa błędów, 60
 operacja wejścia i wyjścia, 57
 operator
 &&, 47
 *, *Patrz:* operator wyłuskania
 ., *Patrz:* operator kropka
 ||, 48
 ->, 72
 alternatywy, 47, 48
 bitowy, 42
 dekrementacji, 41
 dodawania, 41
 dzielenia, 41
 inkrementacji, 41
 kolejność, 41, 43
 koniunkcji, 47
 kropka, 72
 logiczny, 42, 47
 mnożenia, 41
 modulo, 41
 odejmowania, 41
 porównania, 42
 przypisania, 41
 sizeof, 75, 76
 trójargumentowy, *Patrz:* operator warunkowy

warunkowy, 39
 wyłuskania, 72

P

pamięć
 adres, 71
 alokacja dynamiczna, 77
 fragmentacja, 77
 wyciek, 77, 80
 zwalnianie, 77
 pętla
 do while, 51
 for, 52
 przerwanie, 54
 while, 52
 zagnieżdżanie, 53
 plik, 79
 .c, 89
 .h, 89
 .o, 90
 nagłówkowy, 89, 90, 91, 93
 ścieżka, 80
 polecenie, *Patrz:* funkcja
 postinkrementacja, 41
 prawda, 45, 46
 preinkrementacja, 41
 preprocesor, 13, 83, 84
 program
 dobre praktyki, 16, 25, 40, 43, 48, 55, 60, 65,
 78, 86, 93
 gcc, *Patrz:* gcc
 gedit, *Patrz:* gedit
 Terminal, *Patrz:* Terminal

R

rzutowanie, 76

S

słowo kluczowe
 break, 54
 const, 25
 continue, 54
 default, 39
 else, 37
 else if, 38
 error, 85
 extern, 91, 93
 main, 28
 malloc, 76

- note, 85
- NULL, 77
- return, 29
- sizeof, 75
- static, 91, 93
- switch, 39
- typedef, 64, 65, 68
- void, 29, 30
- warning, 85
- while, 51
- specyfikator, 20
 - %c, 20
 - %d, 20, 58
 - %f, 20
 - %p, 20
 - %s, 20
- stała, 25
- standard C99, 48
- sterta, 77
- stos, 77
- struktura, 68, 72
- system
 - autotools, 86
 - cmake, 86
 - operacyjny wirtualny, 11
 - zarządzania procesami kompilacji, 86

T

- tablica, 33
 - deklarowanie, 33
 - element, 34
 - rozmiar, 76
 - typ
 - char, 34
 - int, 34
 - znakowa, 34, 35
 - długość, 35
 - łączenie, 35
 - porównywanie, 35
- Terminal, 11, 14
- typ
 - bool, 49
 - char, 24, 29, 34
 - definiowanie, 64
 - enum, 63, 64
 - float, 24, 29
 - int, 24, 29, 34
 - void, 29, 30
 - wyliczeniowy, *Patrz:* typ enum

U

- Ubuntu
 - instalowanie, 10
 - pobieranie, 7
- unia, 68, 72
- użytkownik nazwa, 10

V

- Virtualbox
 - instalowanie, 8
 - pobieranie, 7

W

- wskaźnik, 71, 73
 - niezaicjalizowany, 77
 - NULL, 77
 - podwójny, 72
 - potrójny, 72
 - tworzenie, 71
 - typ
 - int, 76
 - void, 76
 - wiszący, 77

Z

- zmienna, 23, 24
 - deklarowanie, 92
 - lokalna, 77
 - statyczna, 77
 - typ, *Patrz:* typ
 - zasięg, 30
 - znakowa, 47
- znak
 - !, 42
 - !=, 42
 - %, 41
 - %c, 20
 - %d, 20, 58
 - %f, 20
 - %p, 20
 - %s, 20
 - &, 71
 - &&, 38, 42, 47
 - *, 41
 - ***, 72
 - */, 17

znak	<=, 42
., 72	=, 41
/, 41	==, 42
/*, 17	>, 42
//, 17	->, 72
, 38, 42, 48	>=, 42
+, 41	kropka, 72
++, 41	\n, 15
<, 42	średnik, 15

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>

Programowanie w języku C wcale nie jest tak trudne, jak mogłoby się wydawać. W rzeczywistości może je opanować nawet uczeń podstawówki, o ile tylko będzie pamiętać o zasadach logiki oraz o przeznaczeniu poszczególnych elementów języka. Dlaczego więc tak trudno samemu nauczyć się programować? Przyczyna często tkwi w opisie języka programowania — podręczniki do nauki czyta się tak, jakby ich autorzy zapomnieli, dla kogo piszą. Jeśli nie masz ochoty przedzierać się przez specjalistyczny żargon, ale chcesz programować, ta książka pomoże Ci osiągnąć cel.

Znajdziesz tu wszystko, czego potrzebuje początkujący programista — od wskazówek w kwestii instalacji odpowiednich programów, przez omówienie niezbędnych komend, stałych i zmiennych, po tablice i funkcje. Dowiesz się, jak działają pętle i operatory, do czego służą struktury i unie, jak działają kompilator, preprocesor i linker. Wreszcie zrozumiesz, o co chodzi z operacjami wyjścia i wejścia, a także nauczysz się zapewniać bezawaryjne działanie Twojego programu. I nagle okaże się, że odkrywasz całkiem nowy, fascynujący świat, a porozumienie z Twoim komputerem jest w gruncie rzeczy dziecinnie proste! Sprawdź to!

NAPISZ SOBIE SWÓJ WŁASNY PROGRAM!



- Pierwszy program
- Idea bibliotek i funkcji
- Zmienne, stałe i funkcje
- Tablice i pętle
- Instrukcje warunkowe i operatory
- Podstawy logiki
- Operacje wejścia i wyjścia, podstawowa obsługa błędów
- Enum i typedef
- Struktury i unie
- Malloc i free oraz stos i sterta
- Operacje na plikach i parametry wejściowe programu
- Preprocesor, kompilator i linker
- Pliki nagłówkowe oraz static i extern

PROGRAMUJ W JĘZYKU C!

Helion

36825 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. Kościuski 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-1491-7



9 788328 314917

Informatyka w najlepszym wydaniu

cena: 24,90 zł