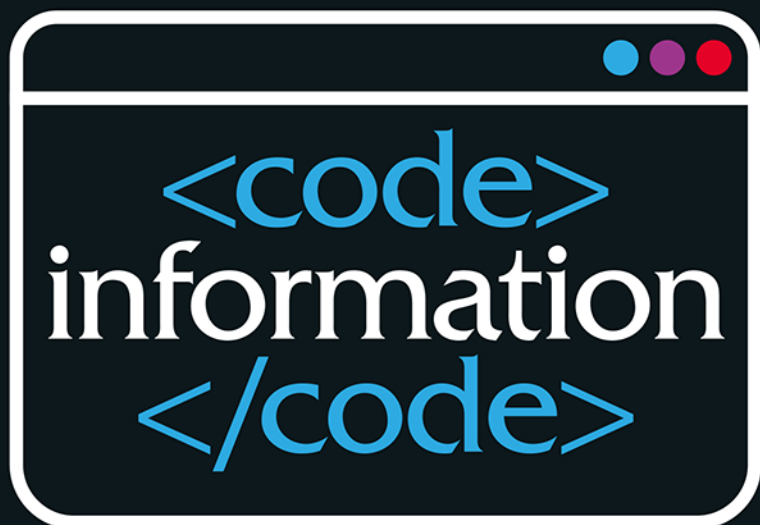


Wojciech Kordecki

---

# INFORMACJA I KODOWANIE

KRÓTKIE WPROWADZENIE  
Z PRZYKŁADAMI ZASTOSOWAŃ



Informacja i kodowanie. Krótkie  
wprowadzenie z przykładami zastosowań  
PDF

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 wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Skład komputerowy w systemie  $\text{\LaTeX}$  wykonał autor.

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/infkod>

Możesz tam pisać swoje uwagi, spostrzeżenia, recenzje.

ISBN: 978-83-289-0088-2

Copyright © Helion S.A. 2024

Printed in Poland.

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

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

# Spis treści

<b>Wstęp</b>	<b>1</b>
<b>1. Wprowadzenie</b>	<b>4</b>
1.1. Znaki i symbole . . . . .	4
1.2. Znaki, litery i liczby . . . . .	9
1.3. Informacja . . . . .	17
1.4. Entropia i redundancja . . . . .	22
<b>2. Źródła i ich entropia</b>	<b>28</b>
2.1. Bezpamięciowe źródła informacji . . . . .	28
2.2. Źródła z pamięcią . . . . .	37
<b>3. Kodowanie</b>	<b>45</b>
3.1. Kody prefiksowe . . . . .	45
3.2. Kod Huffmana . . . . .	52
<b>4. Redundancja i kody</b>	<b>59</b>
4.1. Problemy i przykłady . . . . .	59
4.2. Odległość Hamminga . . . . .	66
4.3. Kod Hamminga . . . . .	69
4.4. Kod Hamminga ogólnie . . . . .	74
<b>5. Kody liniowe</b>	<b>77</b>
5.1. Kody liniowe – teoria . . . . .	77
5.2. Kody liniowe – przykłady . . . . .	82
<b>6. Kody cykliczne i wielomiany</b>	<b>84</b>
6.1. Definicje i przykłady . . . . .	84
6.2. Wielomiany . . . . .	86

6.3.	Generowanie kodów przez wielomiany . . . . .	89
6.4.	Kody dualne . . . . .	94
<b>7.</b>	<b>Kody CRC</b>	<b>97</b>
7.1.	Konstrukcja kodu CRC . . . . .	97
7.2.	CRC – standardy . . . . .	104
<b>8.</b>	<b>Macierze a kodowanie</b>	<b>110</b>
8.1.	Macierze . . . . .	110
8.2.	Generowanie kodów przez macierze . . . . .	113
<b>9.</b>	<b>Przetwarzanie sygnałów</b>	<b>120</b>
9.1.	Sygnały analogowe i cyfrowe . . . . .	120
9.2.	Wymagania digitalizacji . . . . .	130
<b>10.</b>	<b>Kompresja danych</b>	<b>137</b>
10.1.	Kompresja bezstratna . . . . .	138
10.2.	Przykłady kompresji bezstratnej . . . . .	146
10.3.	Kompresja stratna . . . . .	153
<b>11.</b>	<b>Kompresja audio-wideo</b>	<b>155</b>
11.1.	Kompresja grafiki . . . . .	155
11.2.	Kodowanie multimedialnych . . . . .	163
<b>A.</b>	<b>Różne różności</b>	<b>167</b>
A.1.	Logarytmy . . . . .	167
A.2.	Drzewa binarne . . . . .	170
A.3.	Wielomiany . . . . .	171
A.4.	Kod ASCII . . . . .	174
	<b>Przypisy biograficzne</b>	<b>177</b>
	<b>Literatura</b>	<b>182</b>
	<b>Kody QR</b>	<b>184</b>
	Literatura . . . . .	184
	Inne kody QR . . . . .	186
	<b>Skorowidz</b>	<b>187</b>



# Opowiadanie 8

## Macierze a kodowanie

### 8.1. Macierze

#### O pożytkach z tablic

Macierz to tablica, w naszym przypadku tablica liczb, o wymiarze  $m \times n$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

mająca  $m$  wierszy i  $n$  kolumn.

Będziemy korzystać z następujących działań na macierzach:

- dodawania macierzy,
- mnożenia macierzy przez liczbę,
- mnożenia macierzy przez macierz.

Dodawanie:

$$\begin{aligned} & \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}. \end{aligned}$$

Mnożenie macierzy przez liczbę:

$$\alpha \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \dots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \dots & \alpha a_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha a_{m1} & \alpha a_{m2} & \dots & \alpha a_{mn} \end{bmatrix}.$$

Mnożenie macierzy:

$$\begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{i,k-1} & a_{ik} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} \dots & b_{1j} & \dots \\ \dots & b_{2j} & \dots \\ \dots & \dots & \dots \\ \dots & b_{k-1,j} & \dots \\ \dots & b_{kj} & \dots \end{bmatrix} \\ = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & a_{i1}b_{1j} + \dots + a_{ik}b_{kj} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}.$$

Uwaga: Dodawać można tylko macierze o tych samych wymiarach. Mnożyć można tylko macierze takie, że lewy mnożnik ma tyle samo kolumn, co prawy mnożnik wierszy.

Macierzą transponowaną  $A^T$  macierzy  $A$  jest macierz, w której wiersze zamieniono na kolumny, czyli

$$A^T = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \dots & \dots & \dots & \dots \\ a_{1m} & a_{2m} & \dots & a_{nm} \end{bmatrix}.$$

Od tego miejsca, jeśli wyraźnie nie zaznaczymy, że jest inaczej, dodawanie i mnożenie zawsze będzie oznaczać działanie mod 2.

### Przykład 8.1.

Mnożenie i dodawanie:

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$



Niech  $w_1, w_2, \dots, w_k$  będą wybranymi wierszami macierzy  $A$ . Suma tych wierszy pomnożonych przez dowolne liczby  $\alpha_1, \alpha_2, \dots, \alpha_k$

$$\alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_k w_k$$

jest się kombinacją liniową tych wierszy. Jeżeli macierz jest binarna, to dodawanie jest mod 2, więc kombinacja liniowa jest sumą mod 2 wybranych wierszy.

Jeżeli żaden z wierszy nie jest kombinacją liniową pozostałych wierszy, to te wiersze są niezależne. Wiersze niezależne, takie że wszystkie inne są ich kombinacjami liniowymi, tworzą bazę.

### Przykład 8.2.

W macierzy binarnej

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

wiersz czwarty jest sumą wiersza drugiego i trzeciego. Niezależne są trzy pierwsze wiersze – są bazą. Wiersze pierwszy, drugi i czwarty też tworzą bazę.



Wszystkie definicje niezależności odnoszą się również do kolumn macierzy. Możemy więc też mówić o niezależnych kolumnach i o bazie kolumn, tak jak o bazie wierszy.

### Przykład 8.3.

W macierzy binarnej (tej samej co poprzednio)

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

kolumna czwarta jest sumą trzech pierwszych kolumn, a kolumna pierwsza jest sumą trzech ostatnich.



## 8.2. Generowanie kodów przez macierze

### Piszemy kody na tablicy

W tym punkcie ponownie odwołamy się do artykułu Aydin [3]. Zawarto w nim sporo bardzo użytecznych informacji o macierzach generujących kod.

Niech  $G$  będzie macierzą, której wiersze są słowami kodowymi pewnego kodu liniowego. Zbiór wszystkich niezależnych wierszy macierzy  $G$  nazywa się bazą kodu liniowego  $[n, k]$ , gdzie  $n$  jest liczbą kolumn, a  $k$  jest liczbą wierszy tej macierzy.

#### Przykład 8.4.

Kontrola parzystości w słowie kodowym czterobitowym daje nam kod blokowy  $[4, 3]$ . W słowie kodowym trzy pierwsze bity to bity informacyjne, a czwarty bit to bit parzystości.

$i$	słowo kodowe	$w_i$
0	0 0 0 0	$w_0$
1	0 0 1 1	$w_1 = b_1$
2	0 1 0 1	$w_2 = b_2$
3	0 1 1 0	$w_3 = b_1 + b_2$
4	1 0 0 1	$w_4 = b_3$
5	1 0 1 0	$w_5 = b_1 + b_3$
6	1 1 0 0	$w_6 = b_2 + b_3$
7	1 1 1 1	$w_7 = b_1 + b_2 + b_3$

Wiersze, którymi są słowa kodowe  $b_1, b_2, b_3$ , tworzą bazę. Słowa kodowe  $b_i$  są wierszami macierzy

$$G_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Za bazę można też przyjąć wiersze w odwrotnej kolejności:  $b_3, b_2, b_1$ , co daje macierz

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

**Przykład 8.5.**

Wróćmy znów do kodu Hamminga (7,4) w postaci podanej w tabeli 4.1 na str. 69. Bazą są słowa kodowe zawierające ciągi złożone z bitów informacyjnych:

$$\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}$$

Odpowiadającymi słowami kodowymi są wiersze macierzy

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Zapisaćmy ciąg bitów informacyjnych w postaci jednowierszowej macierzy  $[i_1 \ i_2 \ \dots \ i_k]$ . Macierz o  $k$  wierszach, której wiersze są słowami kodowymi i tworzą bazę kodu, nazywamy macierzą generującą kod. Jeśli  $G$  jest macierzą generującą kod  $[n, k]$ , czyli  $G$  ma  $k$  wierszy i  $n$  kolumn, to

$$[x_1 \ x_2 \ \dots \ x_k] G$$

jest słowem kodowym dla ciągu  $[i_1 \ i_2 \ \dots \ i_k]$  w postaci jednowierszowej macierzy o  $n$  elementach.

**Przykład 8.6.**

Wróćmy do kontroli parzystości. Słowo trzybitowe kodujemy z kontrolą parzystości, korzystając z macierzy generujących  $G_1$  i  $G_2$ .

$$[i_1 \ i_2 \ i_3] \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} = [i_3 \ i_2 \ i_1 \ i_1 + i_2 + i_3].$$

$$[i_1 \ i_2 \ i_3] \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [i_1 \ i_2 \ i_3 \ i_1 + i_2 + i_3].$$

Dla tych dwóch przypadków dostajemy słowa kodowe o parzystej liczbie jedynek.

W następnym przykładzie znów wracamy do kodu Hamminga. Jest to kod wystarczająco prosty, aby dobrze zilustrować metodę, ale jednak nie tak banalny, jakim jest kontrola parzystości.

**Przykład 8.7.**

Kod Hamminga (7,4) w postaci podanej na str. 69 otrzymany za pomocą macierzy generującej z przykładu 8.5.

$$\begin{aligned}
 [i_1 \ i_2 \ i_3 \ i_4] & \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
 & = [i_1 + i_3 + i_4 \ i_1 + i_2 + i_4 \ i_1 \ i_1 + i_2 + i_3 \ i_2 \ i_3 \ i_4].
 \end{aligned}$$



**Przykład 8.8.**

Kod Hamminga (7,4) w postaci (6.1) otrzymamy po odpowiednim przestawieniu kolumn macierzy z przykładu 8.7. Macierz  $G$  generująca kod będzie miała postać:

$$G = [I_k | A_{k \times m}], \quad (8.1)$$

gdzie  $I_k$  jest macierzą jednostkową (wymiaru  $k \times k$ , jedynki na głównej przekątnej, poza tym zera), natomiast macierz  $A_{k \times m}$  wskazuje, które bity informacyjne wchodzi do bitów kontrolnych jako sumy. W tym przykładzie  $k = 4$  i  $m = 3$ :

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \quad (8.2)$$

W wyniku otrzymujemy słowa kodowe, w których na pierwszych czterech pozycjach są bity informacyjne, a na ostatnich trzech znajdują się bity kontrolne.

$$\begin{aligned}
 [i_3 \ i_2 \ i_1 \ i_0] & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \\
 & = [i_3 \ i_2 \ i_1 \ i_0 \ i_3 + i_2 + i_1 \ i_2 + i_1 + i_0 \ i_3 + i_2 + i_0].
 \end{aligned}$$

Na przykład ciąg czterech bitów informacyjnych 1011 da słowo kodowe 1011000.

Macierz generująca kod postaci (8.1) ma walor ogólniejszy niż tylko taki, że ma bardziej uporządkowany wygląd. Wprowadzimy bowiem macierz  $H$  kontroli parzystości, mającą  $n$  wierszy i  $m$  kolumn. Ma ona własność taką, że jeśli  $c_1c_2 \dots c_n$  jest słowem kodowym o  $k = n - m$  bitach informacyjnych i  $m$  bitach kontrolnych, to

$$H [c_1c_2 \dots c_n]^T = [0 \ 0 \ \dots \ 0]^T. \quad (8.3)$$

Inaczej mówiąc, każdy wiersz macierzy  $H$  z każdym słowem kodowym tworzą parę ciągów (wektorów) ortogonalnych.

**Twierdzenie 11.**

*Jeżeli  $G$  jest macierzą generującą kod postaci (8.1), to macierz kontroli parzystości  $H$  wyraża się wzorem*

$$H = A^T | I_m, \quad (8.4)$$

gdzie  $A^T$  jest macierzą transponowaną macierzy  $A = A_{m \times k}$ , ma więc  $m$  wierszy i  $k$  kolumn.  $\square$

**Przykład 8.9.**

Dla macierzy  $A$  określonej wzorem (8.2) macierz kontroli parzystości wyrazi się wzorem

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Sprawdźmy to dla słowa kodowego 1011001:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Jeśli zostanie przekłamaný jeden bit i otrzymamy na przykład 1011001, to

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Daje to syndrom 001.

■

Ogólnie dla kodu Hamminga (7,4), jeśli wystąpił jeden błąd, a odebrany słowem (w postaci jednowierszowej macierzy) jest  $B = [b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ b_6 \ b_7]$ , to  $HB^T$  jest macierzą jednokolumnową. Cyfry w tej macierzy dają pozycję, w której wystąpił błąd w sposób określony w tabeli 8.1.

Tabela 8.1. Określenie pozycji błędu za pomocą syndromu

Syndrom	Pozycja
000	0000000
001	0000001
010	0000010
100	0000100
011	0001001
110	0010000
111	0100000
101	1000000

### Przykład 8.10.

Z tabeli 8.1 otrzymujemy, że syndrom wskazuje na błąd na pozycji bitu siódmego, co istotnie ma miejsce.

■

Powróćmy jeszcze na chwilę do kodu dualnego do kodu Hamminga (7,4), czyli do kodu (7,3) – str. 96. Kod (7,3) ma trzy bity informacyjne i cztery bity kontrolne.

**Przykład 8.11.**

Baza kodu, co widać w tabeli 6.4 (wiersze drugi, trzeci i piąty), to

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Słowa kodowe mają więc następującą postać:

$$\begin{aligned} [i_1 \ i_2 \ i_3] & \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \\ & = [i_1 \ i_2 \ i_3 \ i_2 + i_1 \ i_3 + i_2 \ i_3 + i_2 + i_1 \ i_3 + i_1]. \end{aligned}$$

Otrzymany wynik sprawdzimy dla słów 011 i 111 i porównajmy z tabelą 6.4:

$$\begin{aligned} [0 \ 1 \ 1] & \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = [0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1]. \\ [1 \ 1 \ 1] & \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = [1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0]. \end{aligned}$$

Macierz kontroli parzystości wyraża się wzorem (8.4), więc

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

**Przykład 8.12.**

Słowo kodowe z przykładu 8.11: 0111001 zostało odebrane jako 0001000 – błąd polega na przekłamaniu bitów na trzech pozycjach. Sprawdzenie:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Znaleziono błąd! Został wykryty, ale trzech błędów poprawić nie można. Jeśli jednak wiemy (a raczej zakładamy, bo tego na pewno nie wiemy), że nastąpił jeden błąd, to można go poprawić.

Jeśliby został przekłamany bit tylko na jednej pozycji, na przykład odebrano by 0111000, to

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Łatwo ten błąd poprawić po odnalezieniu w tabeli 6.4 słowa kodowego (jest ich tylko osiem) różniącego się od otrzymanego na jednej pozycji.

■

Z przykładów 8.7, 8.9 i 8.11 wynika, że macierz generująca kod w kodzie Hamminga (7,4) jest macierzą kontroli parzystości w kodzie Hamminga (7,3) dualnym do (7,4) i na odwrót. Nie jest to przypadek. Zachodzi bowiem twierdzenie:

**Twierdzenie 12.**

*Jeżeli  $C$  jest binarnym kodem liniowym o macierzy generującej kod  $G$  i macierzy kontroli parzystości  $H$ , to kod dualny  $C^d$  jest kodem liniowym o macierzy generującej kod  $H$  i macierzy kontroli parzystości  $G$ . □*

Wróćmy do ternarnego kodu Hamminga (4,2) przedstawionego w opowiadaniu 4 na str. 73.

Słowo kodowe ma postać  $(d_1, d_2, c_1, c_2)$ . Macierzą generującą kod może być macierz  $G$ :

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix}.$$

Wtedy

$$[d_1 \ d_2] G = [d_1 \ d_2 \ d_1 + d_2 \ d_1 - d_2],$$

co daje wynik zgodny z równaniami (4.8) i (4.9). Zauważmy też, że kod ten jest kodem liniowym, ale nie jest kodem cyklicznym.





# Skorowidz

## A

alfabet, 4  
Morse'a, 9  
aliasing, 131  
ARJ, 140

## B

baza, 112, 113  
bit, 5  
informacyjny, 68, 69, 113  
kontrolny, 68, 69  
najmłodszy i najstarszy, 12  
parzystości, 60, 68, 85, 113

## C

C++, 126, 170  
CGA, 49, 62, 146  
ciało Galois, 79  
ciąg Markowa, 40  
ciągi ortogonalne, 81, 116  
CRC, 98  
wartość początkowa, 106  
częstość, 29  
częstotliwość Nyquista, 130

## D

digit, 5  
digitalizacja, 121, 126, 127, 129  
dodawanie  
ciągów, 80  
macierzy, 110  
mod  $m$ , 78

## drzewo

binarne, 9, 52, 170  
Huffmana, 54, 150  
puste, 170

## dzielenie

mod  $m$ , 77  
wielomianów, 87

## E

efektywność kodu, 56  
entropia, 22  
kalkulator, 33  
własności, 34  
źródła, 32

## F

flip-flop, 38  
z losowością, 38  
format  
AVI, 166  
BMP, 156  
GIF, 156, 158  
hex Intela, 65  
JPEG, 156, 158  
MP3, 165  
PNG, 156  
RIFF, 166  
TIFF, 156  
WAV, 164  
WMV, 166  
fraktal, 157, 159

**I**

iloczyn

- mod 2, 78
- mod 3, 79
- skalarny, 80

ilość informacji, 18, 32

izomorfizm, 80

**K**

kod

- ASCII, 15, 59
- BCD, 16
- blokowy, 69, 81, 84
- CP 852, 16
- CP 1250, 16
- CRC, 97
  - 3-bitowy, 100
  - 4-bitowy, 101
  - 8-bitowy, 101
  - algorytm, 98
- CRC-16-CCITT, 105
- cykliczny, 84
- dualny, 94
- Graya, 16, 67
- Hamminga, 74, 75
  - (7,3), 95, 117
  - (7,4), 69, 115
  - cykliczny, 91
  - ternarny, 73
- hex, 12
  - Intela, 65
- Huffmana, 52, 138, 150
  - adaptacyjny, 138
- ISO 8859-2, 16
- jednoznacznie dekodowalny,
  - 45, 47
- liniowy, 81
- PESEL, 63, 81
- prefiksowy, 46, 47
- repetycyjny, 82

systematyczny, 27, 62

szesnastkowy, 12

telegraficzny, 13

uzupełnień do 2, 10, 65

kodowanie, 5

słownikowe, 140

adaptacyjne, 142

dynamiczne, 142

kombinacja liniowa, 81, 112

kompresja, 21

bezstratna, 52

JPEG, 158, 165

LZW, 142

RLE, 138, 152

z flagami, 139

słownikowa, 140

stratna, 153

konkatenacja, 50, 62, 142

kontrola

nieparzystości, 61

parzystości, 25, 61, 90, 114,  
116

konwerter

A/D, 129

D/A, 129

korzeń drzewa, 170

kwantyzacja, 126

**L**

LCD, 7

liczba

 $e$ , 167 $\pi$ , 37

liść drzewa, 170

logarytm, 11

naturalny, 169

programowanie, 169

zmiana podstaw, 167

**M**

macierz, 110  
     generująca kod, 114  
     kontroli parzystości, 116  
     transponowana, 111

Maxima, 126, 170, 173  
     divide, 173

mikrokontroler, 65

mnożenie  
     ciągu przez liczbę, 80  
     macierzy, 111  
     przez liczbę, 111  
     mod  $m$ , 78

MPEG, 165

**N**

nierówność  
     Krafta, 47  
     trójkąta, 67

niezależność  
     kolumn, 112  
     wierszy, 112  
     zdarzeń, 30, 31

**O**

odległość  
     Hamminga, 66  
     minimalna, 67

**P**

Pascal, 126, 170  
 PCM, 164  
 PESEL, 63  
 piksel, 155  
 półbajt, 12  
 prawdopodobieństwo, 18, 29, 30  
 prefiks, 46, 142  
 próbkowanie, 125, 131  
 Python, 126, 170

**R**

RAR, 140  
 redundacja, 59  
 RGB, 156  
 rozdzielczość próbkowania, 127,  
     164

**S**

SECEDED, 72  
 Słopiewnie, 141  
 słownik, 140, 142  
 słowo, 5, 140  
     kodowe, 5  
 sprawność kodu, 69, 76  
 standard  
     DVD, 166  
     MP4, 166

suma  
     mod 2, 78  
     mod 3, 79  
     ciągów, 80  
     kontrolna, 64, 65

sygnał  
     cyfrowy, 121

syndrom, 71

system  
      $F_2$ , 78  
      $F_3$ , 79  
      $F_3^*$ , 79  
     pozycyjny, 10

**średnia**

statystyczna, 20, 52  
 ważona, 52

**T**

tw. Kotelnikowa-Shannona, 131

**W**

wektory ortogonalne, 81, 116

wiadomości

elementarne, 28

niezależne, 30

widmo częstotliwości, 131

wielomian, 86

dodawanie, 86

dzielenie, 87, 172

reszta, 88

generujący kod, 89

mnożenie, 87

**X**

xor, 78

**Z**

zdarzenia, 29

niezależne, 30

zera wiodące, 10

ZIP, 140

źródło, 28

bezpamięciowe, 31


ergodyczne, 41

$k$ -tego rzędu, 40

rozszerzone, 35

# PROGRAM PARTNERSKI

— GRUPY 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>

GRUPA  
**Helion**

# Naukę programowania zaczynij od solidnych podstaw

Jak sądzisz, co stanowi bazę informatyki? Od czego powinien zacząć przyszły programista? Może od opanowania jednego z najpopularniejszych języków programowania? Oczywiście mógłby od tego rozpocząć, tyle że to trochę tak, jakby uczyć się korzystać z narzędzia bez świadomości, do czego ono właściwie służy. Języki programowania to praktyka. Tymczasem niezbędną wiedzą, którą także powinien opanować każdy przyszły informatyk, są podstawy dotyczące teorii informacji i kodowania.

Wraz z tą książką przyswoisz je bez konieczności odwoływania się do zaawansowanej matematyki i samej informatyki. Ten podręcznik obejmuje najważniejsze kwestie: od znaków, które przekazują informacje, źródeł informacji i sposobów mierzenia ilości przekazywanych danych po przetwarzanie sygnałów z analogowych na cyfrowe i odwrotnie. Po drodze zgłębisz takie zagadnienia jak podstawy kodowania (w tym kodowanie Huffmana), bezstratna kompresja i digitalizacja danych, grafiki i dźwięku, a także konstrukcja kodów liniowych i cyklicznych.

**Dr hab. inż. Wojciech Kordecki** — absolwent Wydziału Elektroniki Politechniki Wrocławskiej, uzyskał doktorat z matematyki na tej samej uczelni i habilitację z matematyki na Uniwersytecie im. Adama Mickiewicza w Poznaniu. W latach 1971–2008 pracował na Politechnice Wrocławskiej, obecnie jest profesorem uczelni w Collegium Witelona Uczelni Państwowej w Legnicy. Jego główna specjalizacja naukowa to matematyka dyskretna — grafy i matroidy losowe, ale również badania operacyjne, informatyka, teoria niezawodności i inżynieria biomedyczna.

	<b>KOD KORZYŚCI</b> <i>Sięgnij po więcej!</i> ▶ 
 <b>helion.pl</b>	ISBN 978-83-289-0088-2  9 788328 900882
 <b>HELION SA</b> ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	<b>Cena: 59,00 zł</b>