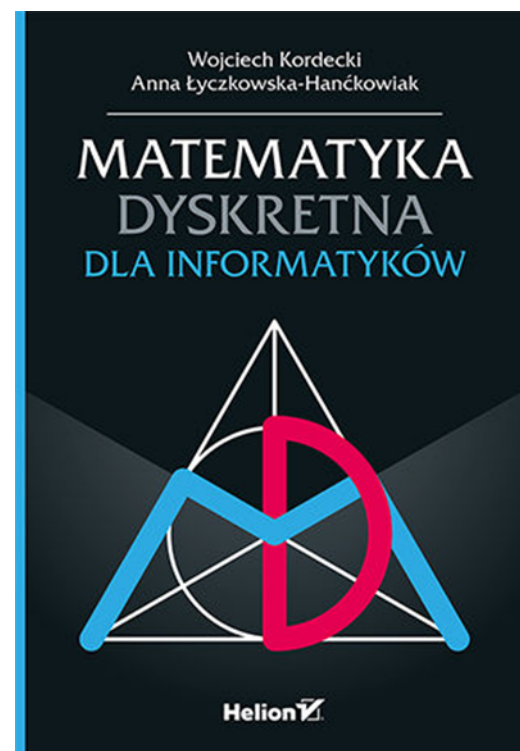
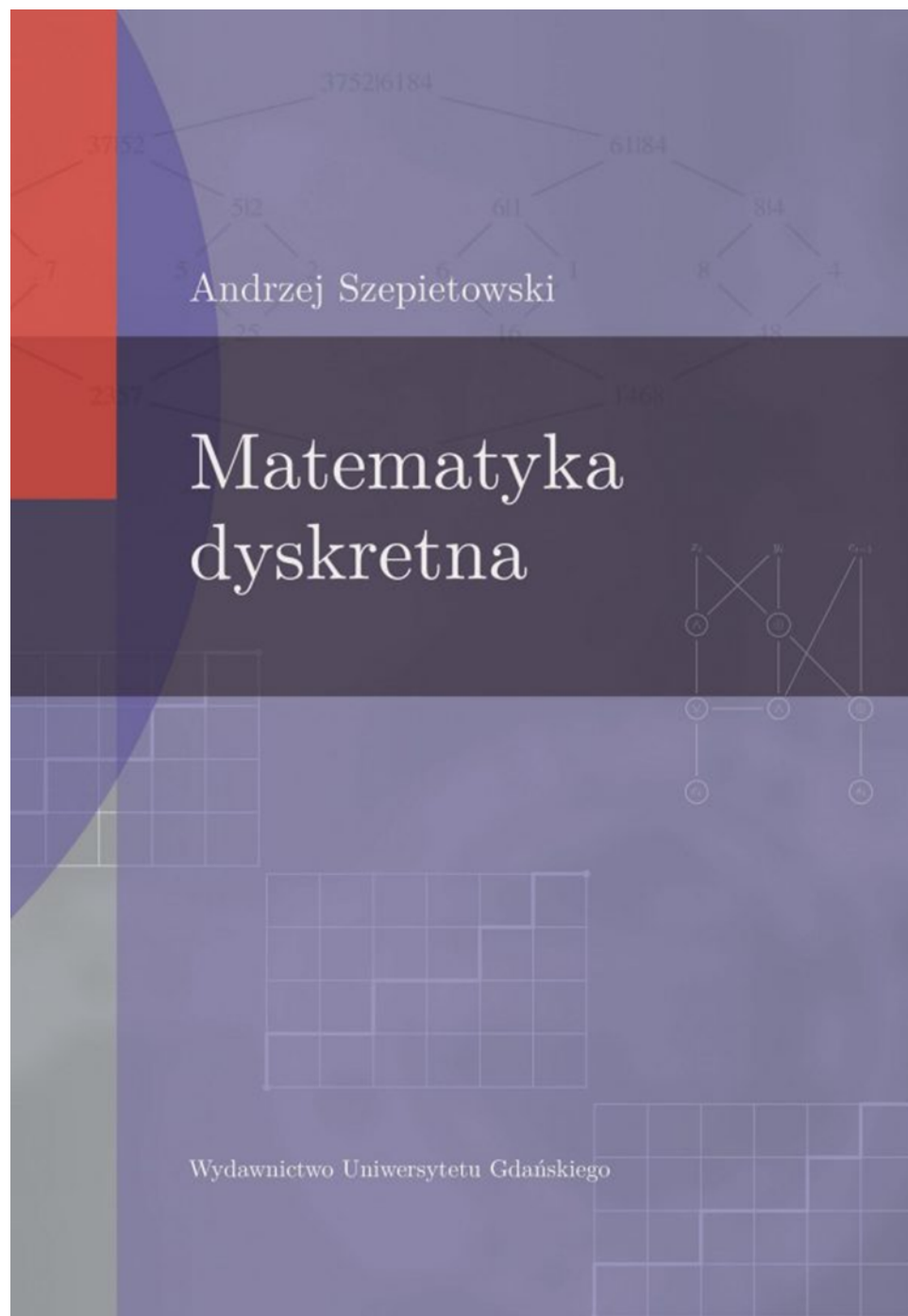




ELEMENTY ARYTMETYKI

— SYSTEM BINARNY —

2-pis

$$\underbrace{\left(\begin{array}{c} 327 \\ 210 \end{array} \right)}_{10}$$

=

2-pis

$$\underbrace{\left(\begin{array}{c} 327 \\ 210 \end{array} \right)}_8$$

$$3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0$$

wartość

≠

$$3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0$$

wartość

ZAMIANA SYSTEMÓW ZAPISU LICZBY

Niech $b = d_r d_{r-1} \dots d_1 d_0$ będzie zapisem liczby w systemie dwójkowym. Zamiana zapisu liczby b na zapis w systemie dziesiętnym odbywa się poprzez wykonanie dodawania oraz potęgowania w następującym wyrażeniu:

$$d_r \cdot 2^r + d_{r-1} \cdot 2^{r-1} \dots d_1 \cdot 2^1 + d_0 \cdot 2^0,$$

przy czym operacje dodawania i potęgowania wykonywane są w systemie o podstawie 10.¹

Zapisz liczbę $(10010)_2$ w systemie dziesiętnym.

$(0)_{10}^+$

◀ PRZYKŁAD

$$(10010)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (16)_{10} + (0)_{10} + (2)_{10} + (0)_{10} = (18)_{10}$$

Handwritten notes: "wartości" under the summands, and a bracket grouping the first four terms.

¹Dla ułatwienia będziemy przyjmować, że zapis $(x_r \dots x_0)_k$ oznacza zapis w systemie o podstawie k , przy czym w dalszych rozważaniach, dla uproszczenia notacji, będziemy zakładać, że brak indeksu przy zapisie liczby oznacza zapis w systemie dziesiętnym.

Z drugiej strony, niech b będzie liczbą zapisaną w systemie dziesiętnym. Zamiana zapisu liczby b na zapis w systemie dwójkowym odbywa się poprzez rozłożenie b na sumę kolejnych potęg dwójki:

$$b = d_r \cdot 2^r + d_{r-1} \cdot 2^{r-1} \dots d_1 \cdot 2^1 + d_0 \cdot 2^0,$$

gdzie $d_i \in \{0, 1\}$. Wówczas $b = (d_r d_{r-1} \dots d_1 d_0)_2$.

Zapisz liczbę $(81)_{10}$ w systemie dwójkowym.

◀ PRZYKŁAD

$$(81)_{10} = (64)_{10} + (16)_{10} + (1)_{10} = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (1010001)_2$$

6 4 0

$$(81)_{10} = 64 + 16 + 1 = 64 + 16 + 1$$

$$(84)_{10} = ()_5 = 3 \cdot 25 + 9 = 3 \cdot 25 + 1 \cdot 5 + 4 \cdot 1 = (314)_5$$

Owe rozłożenie na kolejne potęgi dwójki można wykonać algorytmicznie. A dokładnie, dokonujemy kolejnych dzielení liczby b w sposób całkowity przez 2 (w systemie dziesiętnym) i zapamiętujemy reszty z tegoż dzielenia. Reszty te, zapisane w odwrotnej kolejności, utworzą nam zapis binarny liczby b .

Zapisz liczbę $(81)_{10}$ w systemie dwójkowym.

◀ PRZYKŁAD

Podójście algorytmiczne daje nam następującą tabelę ilorazów i reszt:

$$(34)_5 = ()_2$$

$$\begin{array}{r} /2 \\ 34 \end{array}$$

liczba	iloraz	reszta
81	40	1.
40	20	0.
20	10	0.
10	5	0.
5	2	1.
2	1	0.
1	0	1.

$$(84)_{10} = ()_5$$

$$\begin{array}{r} /5 \\ 84 \end{array} \begin{array}{r} 16 \\ 4 \end{array} \begin{array}{r} 3 \\ 1 \\ 3 \end{array}$$

Kolejne reszty, czytane „od dołu”, są następujące: 1, 0, 1, 0, 0, 0, 1, a zatem

$$(81)_{10} = (1010001)_2$$

$$= (\bar{3}14)_5$$

Algorytm zwiększania liczby o jeden w systemie dwójkowym

$$\begin{array}{r} (314)_5 \\ + 1 \\ \hline (320)_5 \end{array}$$

1. Wskaż ostatni bit rozważanej liczby.

2. Powtarzaj, co następuje:

2.a. Jeżeli wskazywany bit to „0”, to zamień go na „1”; KONIEC.

2.b. W przeciwnym przypadku zamień go na „0” i wskaż kolejny bit na lewo; jeżeli nie ma następnego bitu w lewo, to wstaw „1”; KONIEC.

10: $327 + 1 = 328$

$$\begin{array}{r} 327 \\ + 1 \\ \hline 328 \end{array}$$

zapis $\boxed{?}$ zapis „wskazni + 1”

$$\begin{array}{r} 327 \\ \hline 328 \end{array}$$

Prześledź działanie algorytmu dodawania jedynki dla liczb 10010 oraz 101011.

◀ PRZYKŁAD

a) $10010 + 1 = 10011$, ponieważ
 $1001\underline{0} \rightarrow (= 0) \rightarrow 10011$ (KONIEC).

b) $101011 + 1 = 101100$, ponieważ
 $10101\underline{1} \rightarrow (= 1) \rightarrow 1010\underline{10} \rightarrow (= 1) \rightarrow 101\underline{000} \rightarrow (= 0) \rightarrow 101100$ (KONIEC).

$$3299 + 1$$

$$\begin{array}{r} 3299 \\ + 1 \\ \hline 3300 \end{array}$$

Algorytm porównywania liczb w systemie dwójkowym

1. Jeżeli liczby są różnej długości, to większą jest liczba o dłuższym zapisie.
2. Jeżeli liczby są tej samej długości, to porównujemy bit po bicie od lewej strony do prawej:
 - 2.a. Jeżeli bity są takie same, to przechodzimy do następnego bitu w prawo;
 - 2.b. Jeżeli bity są różne, to większą jest liczba o większym bicie na rozważanej pozycji; KONIEC.
3. Jeżeli wszystkie bity są takie same, to porównywane liczby są równe i KONIEC.

Prześledź działanie algorytmu porównywania liczb dla podanych niżej par liczb.

◀ PRZYKŁAD

a) 101101 oraz 11110

Jako że pierwsza liczba jest liczbą 6-bitową, a druga — 5-bitową, otrzymujemy, że $101101 > 11110$.

b) 1011101 oraz 1011001

$(\underline{1}011101 ? \underline{1}011001) \rightarrow (=) \rightarrow (10\underline{1}1101 ? 10\underline{1}1001) \rightarrow (=) \rightarrow$
 $(101\underline{1}101 ? 101\underline{1}001) \rightarrow (=) \rightarrow (1011\underline{1}01 ? 1011\underline{0}01) \rightarrow (=) \rightarrow$
 $(10111\underline{0}1 ? 10111\underline{0}01) \rightarrow (>) \rightarrow$, a zatem 1011101 $>$ 1011001.

329 ? 1440
329 > 318
29 > 18

Algorytm dodawania liczb w systemie dwójkowym

- Aby dodać do siebie dwie liczby zapisane w systemie dwójkowym, dodajemy bit po bicie od prawej do lewej, dodając jednocześnie w każdym z kroków bity przeniesienia z poprzedniej kolumny.

$$\begin{array}{r}
 10: \quad 1 \quad 1 \\
 \quad 3 \quad 7 \quad 5 \\
 + \quad 8 \quad 9 \\
 \hline
 \quad 4 \quad 6 \quad 4
 \end{array}$$

◀ PRZYKŁAD

$$\begin{array}{r}
 1 \\
 \hline
 1 \\
 \hline
 11 \\
 \hline
 10 \\
 \hline
 11 \\
 \hline
 111 \\
 \hline
 101 \\
 \hline
 111 \\
 \hline
 111 \\
 \hline
 111 \\
 \hline
 111 \\
 \hline
 + \quad 101 \\
 \hline
 100110
 \end{array}$$

Wykonaj poniższe dodawania w systemie dwójkowym.

a) $10101 + 111$

$10101 + 111 = 11100$, ponieważ

$$\begin{array}{r}
 111 \\
 \hline
 10101 \\
 + \quad 111 \\
 \hline
 11100
 \end{array}$$

$$(3)_{10} = (11)_2$$

b) $111 + 101 + 111 + 111 + 111$

Rozważmy najpierw sumę $1 + 1 + 1 + 1 + 1$ ostatnich bitów. Jej wartość zapisana w systemie dziesiętnym to 5_{10} , a w binarnym – $(110)_2$. Zatem w tym przypadku bitami przeniesienia są bity 11, natomiast ostatni bit w zapisie szukanej liczby to 0.

Rozważmy teraz sumę $1 + 0 + 1 + 1 + 1 + 0$ przedostatnich bitów powiększoną jeszcze o ostatni bit przeniesienia, czyli o 1. Jej wartość zapisana w systemie dziesiętnym to 5_{10} , a w binarnym – $(101)_2$. A zatem w tym przypadku bitami przeniesienia są bity 10. Itd.

Otrzymujemy ostatecznie, że $111 + 101 + 111 + 111 + 111 + 101 = 100110$.

UKŁAD LOGICZNY DODAJĄCY DWE LICZBY 2-bitowe

$$10 + 11 = \dots$$

	1	0
	1	0
+	1	1
1	0	1

⊕

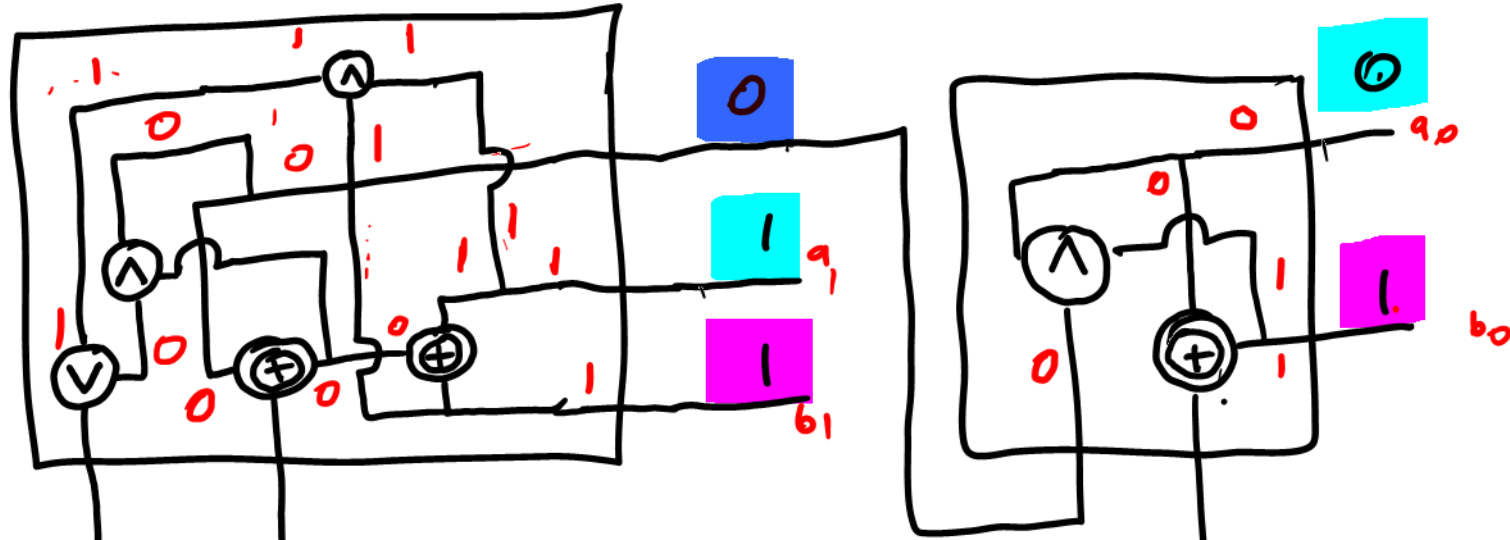
⊕	0	1
0	0	1
1	1	1

∧

∧	0	1
0	0	0
1	0	1

⊕

⊕	0	1
0	0	1
1	1	0



1 0

	a ₁	a ₀
	1	0
+	b ₁	b ₀
	1	1
1	0	1

a ₁	a ₀
1	0
b ₁	b ₀
1	1

$$\begin{array}{r} 1 \cdot 1 \cdot 0 \\ 0 \cdot 1 \cdot 0 \end{array} \quad \boxed{\begin{array}{r} 10 \\ 1 \end{array}}_{10} \quad (10)_2$$

Algorytm odejmowania liczb w systemie dwójkowym

- Aby odjąć od siebie dwie liczby zapisane w systemie dwójkowym, odejmujemy bit po bicie od prawej do lewej, a w przypadku, gdy trzeba odjąć bit większy od mniejszego, „pożyczamy” jedynkę z następnej niezerowej (na lewo) pozycji (o ile jest to jeszcze możliwe)

$$\begin{array}{r} 01(2)_2 \\ 100 \\ - 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 10110 \\ 11000 \\ - 1 \\ \hline 10111 \end{array}$$

Wykonaj poniższe odejmowania w systemie dwójkowym.

a) $10101 - 111$

$10101 - 111 = 1110$, ponieważ

$$\begin{array}{r} 012 \\ 1002 \\ - 111 \\ \hline 1110 \end{array}$$

b) $111000 - 11111$

, oraz $111000 - 11111 = 11001$, ponieważ

◀ PRZYKŁAD

$$\begin{array}{r} 02 \\ 102 \\ 110112 \\ - 111000 \\ \hline 11111 \\ - 11111 \\ \hline 11001 \end{array}$$

$$\begin{array}{r} 10: \quad 139 \\ - \quad 18 \\ \hline 121 \end{array} \quad :-)$$

$$\begin{array}{r} 2^2 \quad 6 \quad (11)_{10} \\ 2 \quad 1 \quad 1 \quad (15)_{10} \\ 2 \quad 1 \quad 2 \quad 5 \\ \quad \quad 3 \quad 8 \\ \hline 20 \quad 8 \quad 7 \end{array}$$

$$\begin{array}{r} 199(14)_{10} \\ 2004 \\ - \quad 7 \\ \hline \end{array}$$

Algorytm mnożenia liczb w systemie dwójkowym

- Aby pomnożyć dwie liczby (zapisane dwójkowo), mnożymy pierwszą liczbę przez poszczególne bity drugiej, a otrzymane wyniki, każdy kolejno przesunięty o jedną kolumnę w lewo, na koniec sumujemy.

$10101 \cdot 101 = 1101001$, ponieważ

10101
 101
 10101
 00000
 10101
 1101001

◀ PRZYKŁAD

$10:$

$$\begin{array}{r} 23 \\ 12 \\ \hline 46 \\ 920 \\ \hline 966 \end{array}$$

$$111 * 11111 = 11111 * 111$$

5 cyframi 3 cyframi

$$\begin{array}{r}
 027, \\
 \hline
 327 : 12 \\
 24 \\
 \hline
 87 \\
 84 \\
 \hline
 3 \dots
 \end{array}$$

$1101001 : 101 = 10101$, ponieważ

$$\begin{array}{r}
 10101 \\
 \hline
 1101001 : 101 \\
 101 \\
 \hline
 == 110 \\
 101 \\
 \hline
 == 101 \\
 101 \\
 \hline
 == 0
 \end{array}$$

◀ PRZYKŁAD

SYSTEMY O PODSTAWIE BĘDĄCEJ POTĘGĄ DWÓJKI

W systemie szesnastkowym używa się następujących „cyfr”:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Przyjmijmy notację, że liczbę zapisaną w systemie szesnastkowym będzie poprzedzać znak dolara \$.

Zamień zapis liczby \$A1 z szesnastkowego na dziesiętny.

◀ PRZYKŁAD

$$\text{\$A1} = 10 \cdot 16^1 + 1 \cdot 16^0 = 160 + 1 = 161.$$

Zamień zapis liczby 320 z dziesiętnego na szesnastkowy.

◀ PRZYKŁAD

	liczba	iloraz	reszta	
320 = \$140, ponieważ	320	20	0	.
	20	1	4	
	1	0	1	

Zamień zapis 10111100 z binarnego na ósemkowy i szesnastkowy.

◀ PRZYKŁAD

$10111100 = (274)_8$, ponieważ $\frac{010}{2} \mid \frac{111}{7} \mid \frac{100}{4}$.

Analogicznie, $10111100 = \$BC$, ponieważ $\frac{1011}{B} \mid \frac{1100}{C}$.

Zamień zapis \$A1 z szesnastkowego na binarny oraz ósemkowy.

◀ PRZYKŁAD

$\$A1 = 10100001$, ponieważ $\frac{A}{1010} \mid \frac{1}{0001}$.

Natomiast aby otrzymać zapis ósemkowy, przekształcamy otrzymany wyżej zapis binarny 10100001 w sposób opisany w poprzednim przykładzie, otrzymując $\$A1 = 10\ 100\ 001 = (241)_8$.

Liczbę $(175)_8$ przedstaw w postaci dwójkowej i dziesiętnej.

◀ PRZYKŁAD

$(175)_8 = 1111101$, ponieważ $\frac{1}{1} \mid \frac{7}{111} \mid \frac{5}{101}$.

Następnie $(175)_8 = 1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = (64)_{10} + (56)_{10} + (5)_{10} = (125)_{10}$.

UŁAMKI W SYSTEMIE DWÓJKOWYM

Zapis $(0.d_1d_2 \dots d_r)_k$ w systemie o podstawie k oznacza liczbę

$$d_1 \cdot k^{-1} + d_2 \cdot k^{-2} \dots d_r \cdot k^{-r}.$$

$$\underbrace{(0.11)_2}_{\text{zapis}} = \underbrace{1 \cdot 2^{-1} + 1 \cdot 2^{-2}}_{\text{wartość}} = \underbrace{(0.75)_{10}}_{\text{zapis}}$$

◀ PRZYKŁAD

$$(0.75)_{10} = 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

UŁAMKI W SYSTEMIE DWÓJKOWYM

Zapis $(0.d_1d_2 \dots d_r)_k$ w systemie o podstawie k oznacza liczbę

$$d_1 \cdot k^{-1} + d_2 \cdot k^{-2} \dots d_r \cdot k^{-r}.$$

$$(0.11)_2 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = (0.75)_{10}$$

◀ PRZYKŁAD

Aby zamienić zapis ułamka (<1) z systemu dziesiętnego na system o podstawie k , należy rozważaną część ułamkową kolejno mnożyć (w systemie dziesiętnym) przez k , wypisując kolejno otrzymywane części całkowite do momentu, aż część ułamkowa będzie równa 0.

Zamień zapis $(0.8125)_{10}$ z dziesiętnego na binarny.

◀ PRZYKŁAD

część całkowita	część ułamkowa
0.	0.8125
1	0.625
1	0.25
0	0.5
1	0.0

Otrzymujemy ostatecznie, że $0.8125 = (0.1101)_2$.

$$(0,3)_{10}$$

$$= ()_2$$

$$= (0.0(1001))_2$$

*2	
0.	0.3
0	.6
1	.2
0	.4
0	.8
1	.6
1	.2
0	.4
0	.8
1	.6
1	.2
...	

SYSTEMY O INNYCH PODSTAWACH

Wykonaj następujące działania:

◀ PRZYKŁAD

a) $(56)_7 + (43)_7 = (132)_7$, ponieważ

$$\begin{array}{r} 11 \\ 56 \\ + 43 \\ \hline 132 \end{array}$$

b) $(41)_5 - (24)_5 = (12)_5$, ponieważ

$$\begin{array}{r} 36 \\ 41 \\ - 24 \\ \hline 12 \end{array}$$

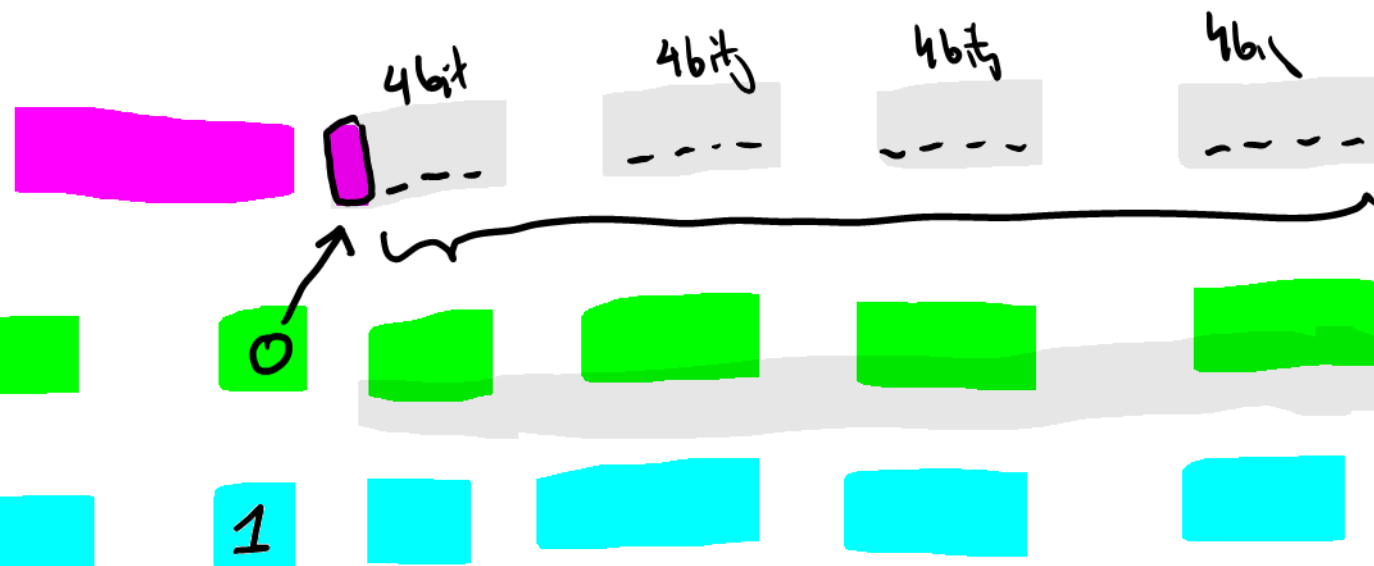
c) $(13)_6 \cdot (4)_6 = (100)_6$, ponieważ

$$\begin{array}{r} 2 \\ 13 \\ \cdot 4 \\ \hline 100 \end{array}$$

REPREZENTACJA LICZB W KOMPETERZE

Zmienne typu `short int` przechowywane są zwykle w dwóch bajtach, czyli 16 bitach.¹ Pierwszy bit określa znak liczby – jeżeli wynosi on 0, to liczba jest dodatnia, w przeciwnym razie liczba jest ujemna.

- Jeżeli liczba jest dodatnia, to pozostałe piętnaście bitów stanowi zapis binarny tej liczby.
- Liczby ujemne przechowywane są w tak zwanym systemie uzupełnieniowym U2, tzn. liczba ujemna o wartości bezwzględnej x przedstawiana jest jako liczba $2^{16} - x$ w postaci binarnej.



¹Analogicznie przechowuje się np. zmienne typu `int`, tylko że w czterech bajtach, czyli 32 bitach.

REPREZENTACJA LICZB W KOMPILERZE

Zmienne typu `short int` przechowywane są zwykle w dwóch bajtach, czyli 16 bitach.¹ Pierwszy bit określa znak liczby – jeżeli wynosi on 0, to liczba jest dodatnia, w przeciwnym razie liczba jest ujemna.

- Jeżeli liczba jest dodatnia, to pozostałe piętnaście bitów stanowi zapis binarny tej liczby.
- Liczby ujemne przechowywane są w tak zwanym systemie uzupełnieniowym U2, tzn. liczba ujemna o wartości bezwzględnej x przedstawiana jest jako liczba $2^{16} - x$ w postaci binarnej.

Rozważmy liczbę 82. Jest ona liczbą dodatnią.

◀ PRZYKŁAD

Jej zapis w postaci binarnej to 1010010. Zatem jest ona przechowywana w postaci:

$$\begin{array}{c|c} \text{znak} & 15 \text{ bitów} \\ \hline 0 & 000\ 0000\ 0101\ 0010 \end{array}, \text{ czyli ostatecznie } 82 = (0000\ 0000\ 0101\ 0010)_{\text{int}}.$$

Rozważmy teraz liczbę -82 . Jest ona liczbą ujemną. Zapis jej wartości bezwzględnej, czyli 82, w postaci binarnej to 1010010. Zatem jest ona przechowywana w postaci:

$$\begin{array}{r} 1 \quad \underbrace{0000\ 0000\ 0000\ 0000}_{16 \text{ zer}} \\ - \quad \quad \quad 1010010 \\ \hline 1111\ 1111\ 1010\ 1110 \end{array}, \text{ czyli ostatecznie } -82 = (1111\ 1111\ 1010\ 1110)_{\text{int}}.$$

2

¹Analogicznie przechowuje się np. zmienne typu `int`, tylko że w czterech bajtach, czyli 32 bitach.

Zapis liczby -82 w postaci `int` można również uzyskać następująco:

- Zapis jej wartości bezwzględnej na 16 bitach „zaprzeczamy” i dodajemy „1”;

$$\begin{array}{r} 0000\ 0000\ 0101\ 0010 \\ \hline 1111\ 1111\ 1010\ 1101 \\ + \qquad \qquad \qquad 1 \\ \hline 1111\ 1111\ 1010\ 1110 \end{array}$$

- Bądź też odejmujemy „1” od jej wartości bezwzględnej na 16 bitach i „zaprzeczamy”.

$$\begin{array}{r} 0000\ 0000\ 0101\ 0010 \\ - \qquad \qquad \qquad 1 \\ \hline 0000\ 0000\ 0101\ 0001 \\ \hline 1111\ 1111\ 1010\ 1110 \end{array}$$

Rozważmy liczbę $(0000000001010010)_{\text{int}}$. Jej „rozkodowywanie” przebiega analogicznie. Jako że pierwszy bit jest równy 0, zatem jest to liczba dodatnia. A zatem, jako że jej zapis binarny to 1010010, zakodowaną liczbą jest 82.

Rozważmy teraz liczbę $(1111111110101110)_{\text{int}}$. Jako że jej pierwszy bit jest równy 1, zatem jest to liczba ujemna. Wyznaczamy ją następująco:

■ Sposób 1:

$$\begin{array}{r} 1\ 0000\ 0000\ 0000\ 0000 \\ -\quad 1111\ 1111\ 1010\ 1110 \\ \hline 0000\ 0000\ 0101\ 0010 \end{array}, \text{ co daje nam } 82, \text{ ale że pierwszy bit był równy } 1,$$

zatem kodowaną liczbą jest -82.

Rozważmy liczbę $(0000000001010010)_{\text{int}}$. Jej „rozkodowywanie” przebiega analogicznie. Jako że pierwszy bit jest równy 0, zatem jest to liczba dodatnia. A zatem, jako że jej zapis binarny to 1010010, zakodowaną liczbą jest 82.

Rozważmy teraz liczbę $(1111111110101110)_{\text{int}}$. Jako że jej pierwszy bit jest równy 1, zatem jest to liczba ujemna. Wyznaczamy ją następująco:

■ Sposób 1:

$$\begin{array}{r} 1\ 0000\ 0000\ 0000\ 0000 \\ - \quad 1111\ 1111\ 1010\ 1110 \\ \hline 0000\ 0000\ 0101\ 0010 \end{array}, \text{ co daje nam } 82, \text{ ale że pierwszy bit był równy } 1, \\ \text{zatem kodowaną liczbą jest } -82.$$

■ Sposób 2 (zapis „zaprzeczamy” i dodajemy „1”):

$$\begin{array}{r} 1111\ 1111\ 1010\ 1110 \\ \hline 0000\ 0000\ 0101\ 0001 \\ + \quad \quad \quad \quad \quad 1 \\ \hline 0000\ 0000\ 0101\ 0010 \end{array}, \text{ co daje nam } 82, \text{ ale że pierwszy bit był równy } 1, \\ \text{zatem kodowaną liczbą jest } -82.$$

■ Sposób 3 (odejmujemy „1” od zapisu i „zaprzeczamy”):

$$\begin{array}{r} 1111\ 1111\ 1010\ 1110 \\ - \quad \quad \quad \quad \quad 1 \\ \hline 1111\ 1111\ 1010\ 1101 \\ \hline 0000\ 0000\ 0101\ 0010 \end{array}, \text{ co daje nam } 82, \text{ ale że pierwszy bit był równy } 1, \\ \text{zatem kodowaną liczbą jest } -82.$$

KODY GRAYA

Kod Graya, zwany również kodem *refleksyjnym*, jest to (uporządkowany) kod binarny (bezwagowy i niepozycyjny), w którym dwa kolejne słowa kodowe różnią się od siebie tylko stanem jednego bitu, a ponadto także ostatni i pierwszy wyraz tego kodu spełniają tę własność (a zatem kod Graya jest także kodem *cyklicznym*).

Algorytm rozszerzania kodu Graya

Założmy, że ciąg C jest kodem Graya, w którym słowa kodowe są długości n . Aby otrzymać kod Graya o słowach długości $n + 1$, rozszerzamy kod C o jeden bit w następujący sposób:

1. Dopisz do ciągu C jego odbicie lustrzane C^R (czyli te same słowa kodowe, ale w odwrotnej kolejności).
2. W tak otrzymanym ciągu, do wyrazów z ciągu C dopisz na początku bit o wartości zero, natomiast do tych z ciągu C^R — bit o wartości 1.

W oparciu o algorytm rozszerzania kodu Graya, wygeneruj (jakiś) kod Graya o słowach kodowych długości trzy.

◀ PRZYKŁAD

Zaczynamy od przykładowego (jednego z dwóch) kodu Graya $C_1 = (0, 1)$ dla 1-bitowych słów kodowych. Przebieg algorytmu rozszerzania kodu C_1 wygląda następująco.

C_1	C_1 oraz jego odbicie lustrzane C_1^R	dopisanie 0 oraz 1
0	0	00
1	1	01
	1	11
	0	10

A zatem mamy (przykładowy) kod Graya dla 2-bitowych słów kodowych: $C_2 = (00, 01, 11, 10)$. Następnie rozszerzamy kod C_2 .

A zatem mamy (przykładowy) kod Graya dla 2-bitowych słów kodowych: $C_2 = (00, 01, 11, 10)$. Następnie rozszerzamy kod C_2 .

C_2	C_2 oraz jego odbicie lustrzane C_2^R	dopisanie 0 oraz 1
00	00	000
01	01	001
11	11	011
10	10	010
	10	110
	11	111
	01	101
	00	100

Otrzymany (przykładowy) kod Graya dla słów 3-bitowych: $C_3 = (000, 001, 011, 010, 110, 111, 101, 100)$.

Należy podkreślić, że w/w metoda tworzy tylko jeden z możliwych kodów Graya. Równie dobrze można dopisywać 0/1 na innej (ustalonej) pozycji. Ponadto zastosowanie tej samej permutacji na każdym ze słów kodowych dla kodu wygenerowanego w w/w sposób także da w wyniku kod Graya.

(Przykładowy) kod Graya n -bitowych słów kodowych można też wygenerować bezpośrednio z naturalnego kodu binarnego, tj. z kolejnych n -bitowych zapisów liczb $0, 1, 2, \dots, 2^n - 1$.

Algorytm wyznaczania i -tego wyrazu n -bitowego kodu Graya

1. Zapisz pomniejszony o jeden numer wyrazu kodu Graya w naturalnym kodzie dwójkowym na zadanej liczbie bitów (brakujące bity uzupełnij bitem 0).
2. Przesuń kopię ciągu z kroku 1 o jeden bit w prawo (najmniej znaczący bit odrzuć, a na początku dopisz bit o wartości 0, czyli podziel całkowicie przez 2).
3. Wykonaj XOR na odpowiednich bitach liczb z kroków 1 oraz 2; wynik jest wyrazem w kodzie Graya.

W oparciu o algorytm wyznaczania i -tego wyrazu n -bitowego kodu Graya wyznacz 10-ty wyraz 4-bitowego (przykładowego) kodu Graya.

◀ PRZYKŁAD

Interesuje nas 10-ty wyraz. Mamy $(9)_{10} = (1001)_2$, a przesunięcie tego ciągu binarnego w prawo o jeden bit daje w wyniku ciąg $(0100)_2$.

$$\begin{array}{r} 1001 \\ XOR 0100 \\ \hline 1101 \end{array}$$

A zatem 10-ty wyraz ciągu Graya to 1101.

Algorytm konwersji wyrazów kodu Graya na wyrazy w naturalnym kodzie binarnym

1. Przyjmij pierwszą (najbardziej znaczącą) cyfrę kodu naturalnego równą pierwszej cyfrze kodu Graya.
2. Każdą kolejną cyfrę oblicz jako XOR odpowiedniej cyfry kodu Graya i poprzednio wyznaczonej cyfry kodu naturalnego.

W oparciu o algorytm konwersji wyrazów kodu Graya na wyrazy w naturalnym kodzie binarnym dokonaj konwersji wyrazu 1110 (4-bitowego kodu Graya).

◀ PRZYKŁAD

Zgodnie z algorytmem w kroku pierwszym otrzymujemy:

$$\begin{array}{r} 1110 \\ \hline 1??? \end{array}$$

Natomiast etapy ustalania kolejnych bitów w kroku drugim przedstawiają się następująco:

Algorytm konwersji wyrazów kodu Graya na wyrazy w naturalnym kodzie binarnym

1. Przyjmij pierwszą (najbardziej znaczącą) cyfrę kodu naturalnego równą pierwszej cyfrze kodu Graya.
2. Każdą kolejną cyfrę oblicz jako XOR odpowiedniej cyfry kodu Graya i poprzednio wyznaczonej cyfry kodu naturalnego.

W oparciu o algorytm konwersji wyrazów kodu Graya na wyrazy w naturalnym kodzie binarnym dokonaj konwersji wyrazu 1110 (4-bitowego kodu Graya).

◀ PRZYKŁAD

Zgodnie z algorytmem w kroku pierwszym otrzymujemy:

$$\begin{array}{r} 1110 \\ \hline 1??? \end{array}$$

Natomiast etapy ustalania kolejnych bitów w kroku drugim przedstawiają się następująco:

$$\begin{array}{r} 1110 \\ \text{XOR } 1 \\ \hline 10?? \end{array} \quad \begin{array}{r} 1110 \\ \text{XOR } 0 \\ \hline 101? \end{array} \quad \begin{array}{r} 1110 \\ \text{XOR } 1 \\ \hline 1011 \end{array}$$

A zatem wyrazem naturalnego 4-bitowego kodu binarnego, który posłużył do utworzenia wyrazu 1110 kodu Graya, jest 1011.

Załóżmy, że mamy „czarne pudełko”, które przechowuje równanie poziomej prostej $y = c$ (której nie znamy), zwanej dalej *murem*. Jedyne możliwe sposoby korzystania z pudełka to zapytanie o to, czy dana wartość d jest większa od c (dla podanego na wejściu argumentu d).

Rozważmy teraz punkt $P = P(t)$, zwany dalej *żabą*, który porusza się, tj. *skacze*, w czasie o wektor $\mathbf{v} = [2, 1]$, startując z punktu $P(0) = (0, 5)$ (czyli $P(0) = (0, 5), P(1) = (2, 6), P(2) = (4, 7), \dots, P(i + 1) = P(i) + \mathbf{v}, \dots$). Zaimplementuj efektywny algorytm wyznaczający największą liczbę skoków, które jest w stanie wykonać żaba, aby nie uderzyć w mur. ● ● ●

Rozważmy zbiór $A = \{x_0, x_1, x_2, x_3, \dots, x_{15}\}$. Załóżmy, że w grze przeciwnik wybiera element x z A , a my musimy za pomocą jak najmniejszej liczby pytań odgadnąć ten element. Wówczas sposób postępowania może być następujący:

Dzielimy zbiór A na dwa rozłączne podzbiory $A_1 = \{x_0, x_1, \dots, x_7\}$ i $A_2 = \{x_8, x_9, \dots, x_{15}\}$ i pytamy przeciwnika, do którego podzbioru należy wybrany przez niego element — niech B będzie tym podzbiorem. Następnie w podobny sposób dzielimy zbiór B na „połowy” i powtarzamy pytanie, itd.

Dla przykładu, niech $A = \{0, 1, 2, \dots, 15\}$ i niech $x = 10$.

$\{0, 1, 2, 3, 4, 5, 6, 7\}$	$\{8, 9, 10, 11, 12, 13, 14, 15\}$
NIE	TAK
$\{8, 9, 10, 11\}$	$\{12, 13, 14, 15\}$
TAK	NIE
$\{8, 9\}$	$\{10, 11\}$
NIE	TAK
$\{10\}$	$\{11\}$
TAK	NIE

czyli ostatecznie $x = 10$. Zadaliśmy 4 pytania.

Powyższy sposób rozumowania można rozszerzyć na dowolny n -elementowy zbiór A , przy czym w najgorszym przypadku minimalna liczba pytań, jaką należy zadać, to $\lceil \log_2 n \rceil$. Zatem np. mając do dyspozycji k pytań można odgadnąć całkowitą liczbę z przedziału od 0 do $2^k - 1$ (czyli element ze zbioru o mocy 2^k).

W szczególności metodę przeszukiwań binarnych można zastosować do stwierdzenia, czy jakaś liczba naturalna n jest kwadratem innej liczby naturalnej, tzn. czy istnieje naturalna liczba k taka, że $k^2 = n$.

Algorytm(int n)

1. $k_d := 1$; $k_g := n$.
2. Powtarzaj aż do skutku:
 - 2.a. Jeżeli $k_g - k_d \leq 1$, to KONIEC: n nie ma pierwiastka.
 - 2.b. $j := \lfloor \frac{k_g + k_d}{2} \rfloor$;
 - 2.c. Jeżeli $j^2 = n$, to KONIEC: n jest kwadratem j ;
 - 2.d. Jeżeli $j^2 > n$, to $k_g := j$, w przeciwnym wypadku $k_d := j$.

Dla podanych niżej liczb wyznacz pierwiastki stopnia drugiego.

◀ PRZYKŁAD

a) 49

k_d	k_g	$?(k_g - k_d \leq 1)$	j	$?(j^2 = n)$	$?(>, <)$
1	49	$?(49 - 1 \leq 1)$	25	$?(25^2 = 49)$	$>$
1	25	$?(25 - 1 \leq 1)$	13	$?(13^2 = 49)$	$>$
1	13	$?(13 - 1 \leq 1)$	7	$?(7^2 = 49)$	KONIEC

czyli ostatecznie istnieje $k = 7$ takie, że $k^2 = 49$.

Dla podanych niżej liczb wyznacz pierwiastki stopnia drugiego.

◀ PRZYKŁAD

a) 49

k_d	k_g	$?(k_g - k_d \leq 1)$	j	$?(j^2 = n)$	$?(>, <)$
1	49	$?(49 - 1 \leq 1)$	25	$?(25^2 = 49)$	$>$
1	25	$?(25 - 1 \leq 1)$	13	$?(13^2 = 49)$	$>$
1	13	$?(13 - 1 \leq 1)$	7	$?(7^2 = 49)$	KONIEC

czyli ostatecznie istnieje $k = 7$ takie, że $k^2 = 49$.

b) 59

k_d	k_g	$?(k_g - k_d \leq 1)$	j	$?(j^2 = n)$	$?(>, <)$
1	59	$?(59 - 1 \leq 1)$	30	$?(30^2 = 59)$	$>$
1	30	$?(30 - 1 \leq 1)$	15	$?(15^2 = 59)$	$>$
1	15	$?(15 - 1 \leq 1)$	8	$?(8^2 = 59)$	$>$
1	8	$?(8 - 1 \leq 1)$	4	$?(4^2 = 59)$	$<$
4	8	$?(8 - 4 \leq 1)$	6	$?(6^2 = 59)$	$<$
6	8	$?(8 - 6 \leq 1)$	7	$?(7^2 = 59)$	$<$
7	8	$?(8 - 7 \leq 1)$	KONIEC		

czyli ostatecznie nie istnieje k takie, że $k^2 = 59$. Jednakże z warunków zatrzymania algorytmu wynika, że otrzymaliśmy przybliżenie: $\sqrt{59} \in (7, 8)$.