Co to jest informacja, oraz w jaki sposób jest ona reprezentowana w komputerze.

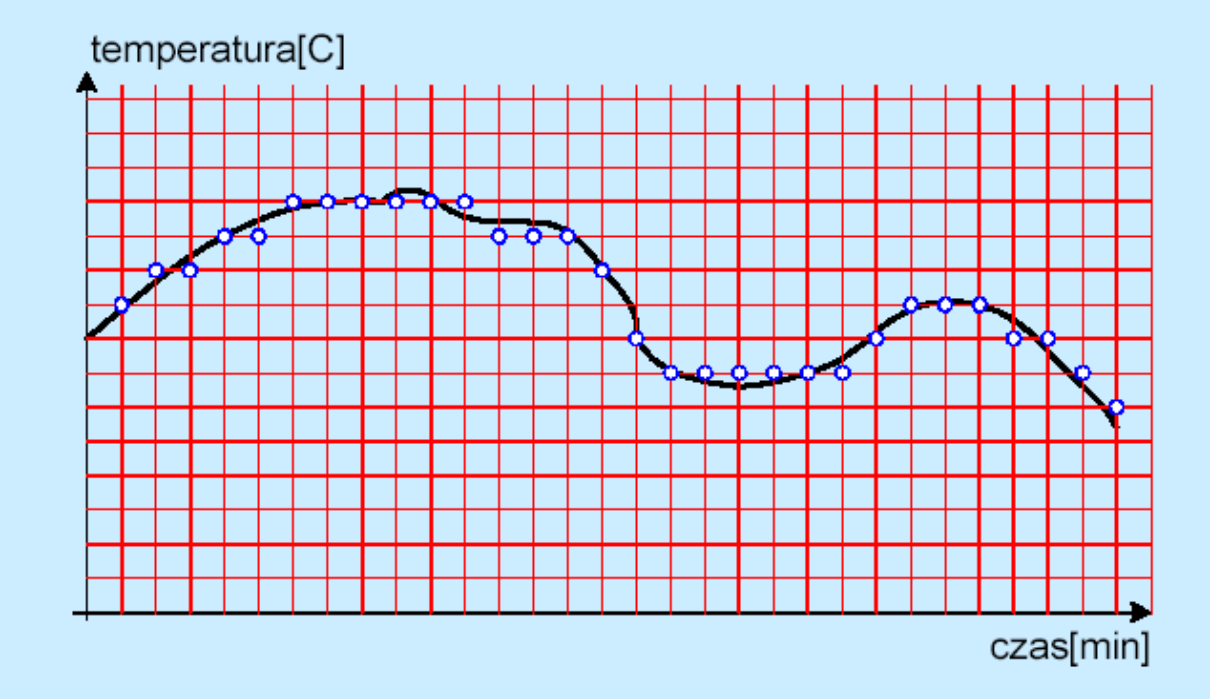
Pojęcie informacji

Informacja to dane niosące ze sobą pewne znaczenie. Komputer jest urządzeniem do przetwarzania danych - wykonuje pewne operacje na danych, które reprezentują informacje.

Informacja to obiekt abstrakcyjny, który w postaci zakodowanej (dane) może być przechowywany (na nośniku danych), przesyłany (np. głosem, falą elektromagnetyczną, prądem elektrycznym), przetwarzany (w trakcie wykonywania algorytmu) i użyty do sterowania (np. komputerem steruje program, będący zakodowaną informacją). Informacja odzwierciedla jakiś fakt, zdarzenie, stan obiektu z fizycznego świata. W oparciu o nią możemy podejmować określone działania.
Dane to symbole, które mogą reprezentować informację, np. ALA, 22, XXII, 10110. Aby zrozumieć te symbole, muszą być one osadzone w pewnym kontekście, a my musimy posiadać odpowiednią wiedzę i umiejętności. Z reguły nie posiądziemy żadnej informacji z rozkładu jazdy zapisanego w języku arabskim lub z tekstu zaszyfrowanego do którego nie posiadamy klucza.

Fizyczna reprezentacja informacji w komputerze

Nośnikiem danych (a zatem i informacji) są sygnały – wielkości fizyczne zmienne w czasie. Najczęściej wykorzystywane są sygnały elektryczne: napięcie, prąd, ładunek elektryczny. Inne wielkości fizyczne, zanim będą przetwarzane przez komputery, są przetwarzane na sygnały elektryczne przez odpowiednie przetworniki (sensory).
Wielkości te mogą być ciągłe np. temperatura pacjenta, szybkość pojazdu, lub dyskretne np. stan magazynu, liczba dzieci w rodzinie itp.



Aby sygnały analogowe mogły być przetworzone przez system komputerowy muszą być przetworzone w kodowane sygnały cyfrowe. Rolę tę spełniają przetworniki analogowo-cyfrowe (A/C) umieszczone na styku części analogowej i cyfrowej systemu.

Przetwarzanie ciągłego sygnału analogowego na sygnał cyfrowy polega na dyskretyzacji sygnału w czasie czyli jego próbkowaniu, dyskretyzacji wartości sygnału czyli kwantowaniu oraz na kodowaniu uzyskanego sygnału dyskretnego. **Próbkowanie** czyli dyskretyzacja argumentów funkcji x(t) następuje przez kolejne pobieranie próbek wartości sygnału w pewnych odstępach czasu, w taki sposób, aby ciąg próbek umożliwiał jak najwierniejsze odtworzenie całego przebiegu funkcji. **Kwantowanie** czyli dyskretyzacja wartości przebiegu analogowego polega na przyporządkowaniu każdej próbce skończonej liczby poziomów amplitudy, odpowiadającym dyskretnym wartościom od zera do pełnego zakresu. Najczęściej stosowane jest kwantowanie równomierne.

Prawo próbkowania Shannona - Kotielnikowa, mówi, że przebieg ściśle dolnopasmowy jest całkowicie określony przez próbki pobierane z częstotliwością co najmniej dwukrotnie większą od maksymalnej częstotliwości występującej w widmie próbkowanego sygnału.

Ponieważ komputer jest urządzeniem elektrycznym, to wszelkie informacje, jakie do niego docierają muszą być reprezentowane również przez sygnały elektryczne. Dzisiejsze komputery budowane są w taki sposób, że rozróżniają jedynie dwa stany elektryczne: 1 – gdy sygnał elektryczny dochodzi do komputera (pierwszy, wyróżniony poziom sygnału), oraz 0 – gdy sygnału brak (drugi, wyróżniony poziom sygnału).

Logiczna reprezentacja informacji w komputerze

Podstawowa ilość informacja nosi nazwę **bitu informacji**. Za pomocą jednego bitu można zapisać dwa stany informacji: lampa świeci lub nie. Z pomocą większej liczby bitów możemy zapisać większe porcje informacji (reprezentować większą liczbę stanów).

bity	ilość stanów, jakie da się zapisać
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536

Za pomocą **N bitów** możemy reprezentować $\mathbf{2^N}$ **stanów**.

Ponieważ za pomocą bitu można przedstawić bardzo małą ilość informacji, dlatego wprowadzono jednostki o większej pojemności czyli: **bajty, kilobajty, megabajty, gigabajty, terabajty** itd.

1 Bajt = 8 bitów jeden znak (nie zawsze)
1 KB = 2^{10} B = 1024 Bajty typowa strona tekstu to kilka kB
1 MB = 2^{20} B = 1024 KB książka bez grafiki lub minuta muzyki
1 GB = 2^{30} B = 1024 MB film cyfrowy, sporo grafiki, ludzki genom
1 TB = 2^{40} B = 1024 GB (terabajt) duża biblioteka
1 PB = 2^{50} B = 1024 TB (petabajt) ludzka pamięć
Słowo = 8, 16, 32, 64, 128, 256 bitów, na których wykonywana jest operacja
Blok = wielkość wektora informacji cyfrowej, którą można przesłać w ramach operacji zapisu lub odczytu informacji zewnętrznej

Jakiego rodzaju dane mogą być przetwarzane przez komputery?

Od początku istnienia komputerów podstawowym rodzajem danych były dane liczbowe. Liczby są pewnymi pojęciami abstrakcyjnymi, które umożliwiają wyrażenie wyniku liczenia przedmiotów oraz wyniku mierzenia wielkości, czyli wyniku porównania jednej wielkości z inną tego samego rodzaju, obraną za jednostkę miary. Ogół zasad umożliwiających przedstawienie liczb za pomocą umownych znaków przyjęto nazywać systemem liczbowym, a znaki, za pomocą których zapisuje się obecnie liczby - cyframi.

Oprócz liczb współczesne komputery przetwarzają teksty, dane multimedialne. Sposób reprezentacji tych danych w komputerach jest niezwykle istotny z punktu widzenia łatwości przetwarzania, wielkości zajmowanych zasobów (pamięci), szybkości i niezawodności przesyłania.

Systemy liczbowe

W życiu codziennym wykorzystujemy *system dziesiętny, pozycyjny*. Oznacza to, że do zapisu dowolnej liczby używamy 10 cyfr (od 0 do 9), a znaczenie każdej cyfry zależy od pozycji jaką zajmuje w liczbie. Przypomnijmy: liczba to symbol oznaczający jakąś wartość, zaś cyfry to znaki do zapisu liczby. Przykładowo liczbę 121 można zapisać w następującej postaci:

$$\mathbf{1 \times 10^2 + 2 \times 10^1 + 1 \times 10^0}$$

Liczba 10 w tym zapisie nazywa się *podstawą* systemu liczenia. W systemie dziesiętnym wykorzystuje się 10 cyfr : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**.

Znamy również liczby w zapisie rzymskim, w którym wykorzystujemy symbole: I, V, X, L, C, D, M. System ten nie jest systemem pozycyjnym – znaczenie cyfry nie zależy od pozycji zajmowanej w liczbie np. IV, VII. Spróbujmy w systemie rzymskim dodać lub pomnożyć dwie liczby.

Mówimy o tuzinie jaj (12 sztuk), kopie jaj (60 sztuk- 5 tuzinów), a gros to tuzin tuzinów (144 sztuki). Korzystamy również z systemu sześćdziesiętnego: godzina ma 60 minut, minuta – 60 sekund, a kąt pełny – 360 stopni. Zwróćmy uwagę na arytmetykę w tym systemie: 40 minut i 40 minut to 1 godzina i 20 minut.

System liczbowy, jakim posługują się komputery wykorzystuje tylko dwie cyfry: **0** i **1** i nazywany jest systemem dwójkowym. Jego popularność wynika z łatwości reprezentacji dwóch stanów za pomocą dwóch wyróżnionych poziomów napięć, wartości lub kierunku prądu w obwodzie, obecności lub braku strumienia światła itp.

Zajmiemy się teraz zamianą liczb dwójkowych (binarnych) na liczby dziesiętne i odwrotnie.

Zadanie: Zamień liczbę binarną 1110 $_{(2)}$ na liczbę dziesiętną:

3	**2**	**1**	**0**
2^3	**2^2**	**2^1**	**2^0**
8	**4**	**2**	**1**
1 x 8	**1 x 4**	**1 x 2**	**0 x 1**

Odp.: 1110 $_{(2)}$ = 8 + 4 + 2 +0 = 14 $_{(10)}$

Zadanie: Zamień liczbę dziesiętną 14$_{(10)}$ na liczbę binarną:

Dzielenie	Iloraz	Reszta z dzielenia	Cyfra
14 : 2	7	**0**	C_0
7 : 2	3	**1**	C_1
3 : 2	1	**1**	C_2
1 : 2	0	**1**	C_3

Dzielenie wykonujemy aż iloraz osiągnie 0. Otrzymana liczba wynosi : (1110) $_2$

Oprócz systemu dwójkowego, zwanego też **binarnym**, programiści często używają systemu szesnastkowego (heksadecymalnego), w którym występuje aż 16 cyfr:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Liczby w systemie		
dziesiątkowym	dwójkowym	szesnastkowym
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Tabela : porównanie liczb w różnych systemach

Zamiana liczb z systemu binarnego na szesnastkowy jest bardzo prosta: cyfry liczby binarnej grupujemy po 4 od prawej strony (ewentualnie uzupełniając z lewej strony zerami) i każdej czwórce przypisujemy odpowiedni symbol według tabeli powyżej. Zapis szesnastkowy np. 32-bitowego adresu pamięci jest zdecydowanie krótszy niż zapis binarny. Pamiętajmy jednak, że komputery rozumieją tylko zapis binarny.

Za pomocą dwóch cyfr możemy zapisywać wartości całkowite, ułamkowe, stało- i zmienno-przecinkowe.

Liczby całkowite reprezentowane w komputerach obejmują tylko pewien podzbiór zbioru liczb całkowitych z matematyki. Zwykle jest to zbiór wszystkich liczb całkowitych mniejszych co do wartości bezwzględnej niż pewna określona wartość maksymalna. Wynika to z możliwości reprezentowania tylko 2^N stanów (wartości) za pomocą N bitów (cyfr binarnych). Przekraczając zakres reprezentowanych liczb napotykamy przypadek tzw. nadmiaru co może sprawić, że po zwiększeniu największej liczby o 1 otrzymujemy w wyniku najmniejszą liczbę (zapętlenie zakresu). Korzystając z komputera, programując go, należy pamiętać o ograniczeniach wynikających z przyjętej reprezentacji danych.

Sposoby kodowania :

Kodowanie liczb – pozycyjne systemy liczbowe o dodatniej podstawie.

W systemie tym (n+m) - pozycyjną nieujemną liczbę :

$$A = a_{n-1}p^{n-1} + \ldots + a_1 p^1 + a_0 p^0 + a_{-1}p^{-1} + \ldots + a_{-m}p^{-m} = \sum_{i=-m}^{n-1} a_i p^i$$

zapisuje się w postaci :

$$(A)_p = (a_{n-1} \ldots a_1 a_0 , a_{-1} \ldots a_{-m})_p = (C_A, U_A)_p = (C_A)_p , (U_A)_p$$

przy czym:

p - podstawa systemu liczbowego, p $\in$ {2, 3, ...};

a_i - cyfra i-tej pozycji, $a_i \in$ {0, 1, ..., p-1};

n - ilość cyfr części całkowitej liczby A;

m - ilość cyfr części ułamkowej liczby A

Przecinek oddziela część całkowitą C_A liczby A od jej części ułamkowej U_A.

Podstawa pozycyjnego systemu liczbowego odpowiada ilości cyfr wykorzystywanych do przedstawienia liczb w tym systemie. Jeżeli m = 0, to liczba A jest całkowita, jeżeli n = 0 , to ułamkowa. Jeśli natomiast m i n są niezerowymi liczbami całkowitymi, to A jest liczbą mieszana.

Cyfra a_{n-1} - jest **cyfrą najbardziej znaczącą** (ang. Most Significant Digit - MSD),
a_{-m} - **cyfrą najmniej znaczącą** (ang. Least Significant Digit - LSD)

W systemach cyfrowych najbardziej powszechne uznanie znalazły następujące pozycyjne systemy liczbowe :

- Kod binarny - system dwójkowy

  (kod binarny prosty – służy do kodowania liczb dodatnich)

  $a_i \in \{0, 1\}$ podstawa = 2

  $$L = \sum_{i=0}^{n-1} 2^i a_i \in \langle 2^n - 1, 0 \rangle$$

- Kody ósemkowe (system oktalny)

  $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ podstawa = 8

  $$L = \sum_{i=0}^{n-1} 8^i a_i \in \langle 8^n - 1, 0 \rangle$$

- Kody dziesiętne – system dziesiętny (dziesiątkowy, decymalny)

  $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ podstawa = 10

- Kody szesnastkowe – (system heksadecymalny, heksagonalny)

  $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ podstawa = 16
  przy czym A = 10, B = 11, ... , F = 15

Przedstawianie liczb ze znakiem :

Operacje arytmetyczne wykonuje się na liczbach zarówno dodatnich, jak i ujemnych. W systemach cyfrowych wykorzystuje się takie przedstawienia liczb, że ich znaki określane są przez jedną z cyfr przedstawienia, w określonym miejscu rozwinięcia, zwykle najbardziej znacząca. Jest to tzw. cyfra znaku, która na ogół przyjmuje wartości:
0 - dla liczb dodatnich,
1 - dla liczb ujemnych.

Pozostałe cyfry liczby określa się wtedy nazwą cyfr liczbowych.
Przy zapisie cyfrę znaku oddziela się od cyfr liczbowych umownym symbolem kropki.

Liczbę ujemną można przedstawić następującymi sposobami:

- **znak-moduł** (zapis modułowy, zapis prosty) :
liczby mające takie same wartości bezwzględne są przedstawione identycznie z wyjątkiem cyfry znaku. W zapisie tym rozróżnia się dwa rodzaje zera:
  - zero dodatnie: +0 = 0.0...0,0...0,
  - zero ujemne: -0 = 1.0...0,0...0.

Nie stosuje się w przypadku działań arytmetycznych.

$$a_n = \begin{cases} 0 - l.dodatnia \\ 1 - l.ujemna \end{cases}$$

$$L = \begin{cases} \sum_{i=0}^{n-1} 2^i a_i ; a_n = 0 \\ -\sum_{i=0}^{n-1} 2^i a_i ; a_n = 1 \end{cases}$$

- **kod uzupełnieniowy do 2** :

$$L = -2^n a_n + \sum_{i=0}^{n-1} 2^i a_i \in \langle -2^n, +2^n - 1 \rangle$$

$$L = a_n, a_{n-1}, \ldots, a_1, a_0$$

$$-L = (\bar{a}_n, \bar{a}_{n-1}, \ldots, \bar{a}_1, \bar{a}_0) + 1$$

Korzystając z N bitów można zapisać liczby w przedziale od -2^{N-1} do $2^{N-1}-1$. Dla liczb 8-bitowych będą to wartości od -128 do 127.

Specyficznym systemem liczbowym jest **system dziesiętny kodowany dwójkowo**, który również znalazł szerokie zastosowanie w systemach cyfrowych. System ten jest systemem liczbowym, w którym cyfry dziesiętne są przedstawione w kodzie dwójkowym (kodowane dwójkowo).

Dla jednoznacznego przedstawienia 10 cyfr (0, 1, ..., 9) systemu dziesiętnego za pomocą zestawu symboli zerojedynkowych trzeba zastosować kod dwójkowy przynajmniej 4-pozycyjny (4-bitowy). Kody służące do kodowania dwójkowego cyfr systemu dziesiętnego noszą nazwę kodów dwójkowo-dziesiętnych.

W tabeli podano powszechnie stosowany 4-bitowy kod dwójkowo-dziesiętny.

Kody BCD	
Cyfra dziesiętna	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Arytmetyka z wykorzystaniem kodów BCD wymaga specjalnych rozkazów i czasami wykonywania korekcji, gdyż np. dziesiętnie 5 + 5 = 10
w kodzie BCD da wynik 0101 + 0101 = 1010, który nie reprezentuje, żadnej cyfry.
Dokonuje się wówczas korekcji przez dodanie wartości 6 czyli 0110 i uzyskuje wynik 0001 0000 co odpowiada w zapisie BCD dziesiętnej liczbie 10.

Reprezentacja zmiennopozycyjna liczb

Obliczenia naukowe, a nierzadko również inżynierskie, są często prowadzone na bardzo dużych lub bardzo małych liczbach. Na przykład liczba Avogadro, będąca liczbą cząsteczek w jednym molu, wynosi $6{,}02252*10^{23}$, a stała Plancka, będąca jednostką energii, jest równa $6{,}62607*10^{-34}$. Gdybyśmy chcieli użyć dziesiętnego systemu stałopozycyjnego to liczby te miałyby postać:

602 252 000 000 000 000 000 000 , 000 000 000 000 000 000 000 000 000 000 000 000 000

000 000 000 000 000 000 000 000 , 000 000 000 000 000 000 000 000 000 000 000 662 607

Aby uniknąć takich reprezentacji, stosuje się znormalizowaną reprezentację zmiennopozycyjną. Liczba w tej reprezentacji ma postać:

$$a = m*10^c$$

gdzie ***m*** jest liczbą ułamkową, zwaną **mantysą**, spełniającą nierówność $0,1 \leq |m| < 1$, a ***c*** jest liczbą całkowitą i nazywa się **cechą** liczby ***a***.
Znormalizowana reprezentacja zmiennopozycyjna liczb, które są zapisywane w komputerze ma postać, w której podstawą jest liczba 2, a zatem:

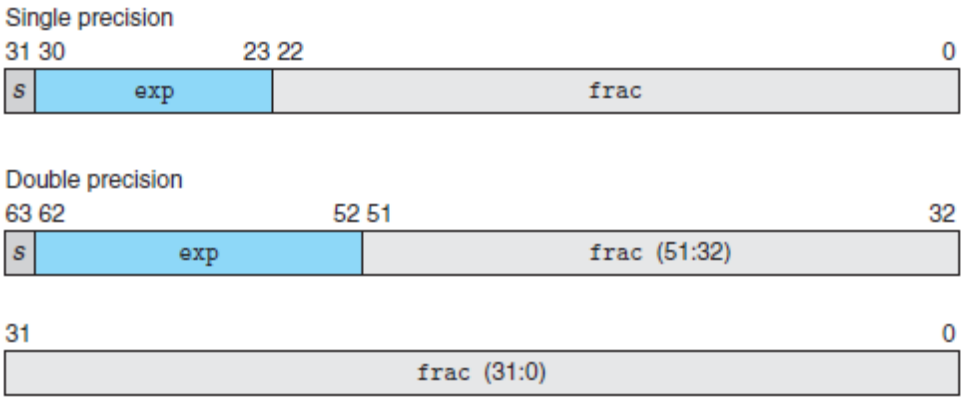
$$a = m*2^c$$

gdzie mantysa spełnia nierówność $0,5 \leq |m| < 1$, by pierwsza cyfra po kropce w jej rozwinięciu binarnym była równa 1. Mantysa i cecha liczby są zapisywane w komputerze w binarnej reprezentacji stałopozycyjnej i przeznacza się na nie stałą liczbę bitów. Na przykład liczba π w zapisie dziesiętnym może być przedstawiona jako $3,1415=0,785375*2^2$. Następnie przedstawiamy mantysę i cechę w postaci binarnej na dwóch bajtach jako
0110010 00000010, gdzie pierwszy bajt jest reprezentacją mantysy, a drugi cechy.

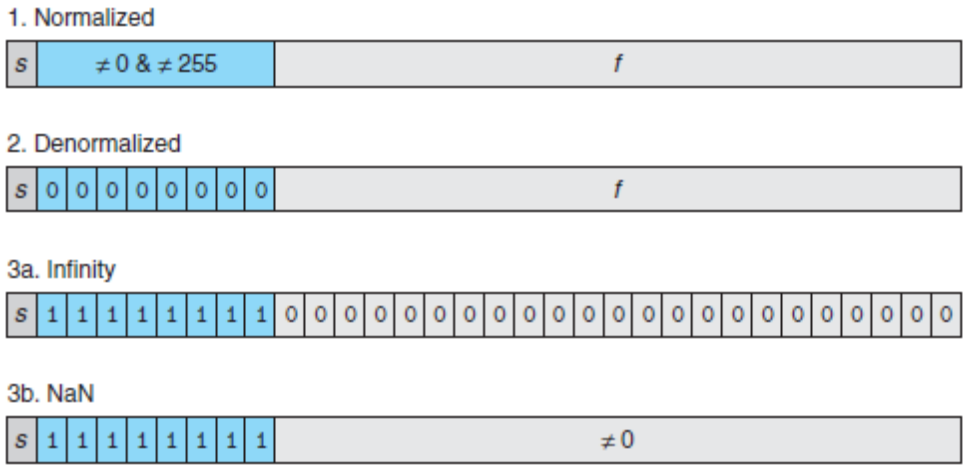
Format IEEE 754

Aby ujednolicić wyniki obliczeń numerycznych wykonywanych na różnych platformach sprzętowych, wprowadzono ściśle określony standard zapisu zmiennoprzecinkowego IEEE 754. Pełna nazwa standardu to: IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985). Obecnie praktycznie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych oparte są na tym standardzie.
Standard IEEE 754 definiuje dwie podstawowe klasy liczb zmiennoprzecinkowych:
32 bitowe - pojedynczej precyzji (ang. single-precision),
64 bitowe - podwójnej precyzji (ang. double-precision).

Format	Bit znaku	Bity cechy	Bity mantysy
32 bity - pojedyncza precyzja	1 bit	8 bitów	23 bity
64 bity - podwójna precyzja	1 bit	11 bitów	52 bity

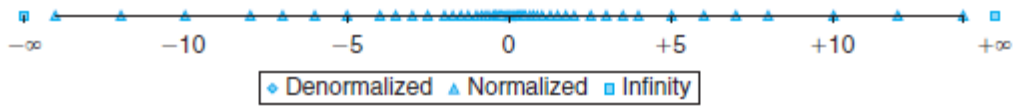


Format liczb zmiennoprzecinkowych pojedynczej i podwójnej precyzji

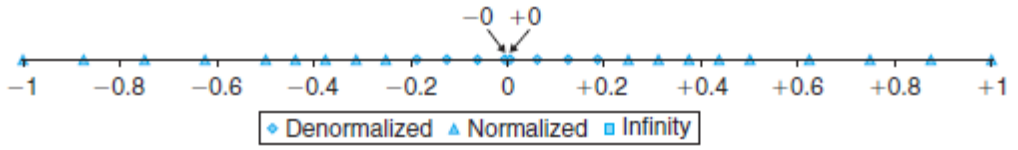


Kategorie liczb zmiennoprzecinkowych pojedynczej precyzji (float):

1. Znormalizowane
2. Zdenormalizowane
3. Wartości specjalne
   - nieskończoność
   - nie liczba (NaN) np. wynik dzielenia przez 0, pierwiastek z -1, wynik ∞ - ∞



(a) Complete range



(b) Values between −1.0 and +1.0

Wartości reprezentowane za pomocą 6-bitowego formatu zmiennoprzecinkowego, gdzie cecha ma 3 bity, mantysa 2, a przesunięcie wynosi 3 ($2^{3-1} - 1$).

Standardy reprezentowania znaków alfanumerycznych

Kody alfanumeryczne – stosowane do zapisu znaków pisarskich (litery, cyfry, znaki przestankowe i inne)

Proces przetwarzania wymaga uprzedniego wprowadzenia informacji i zapamiętania ich. Informacje są wprowadzane do komputera najczęściej w postaci ciągów znaków którymi są na ogół liczby lub teksty. Zbiór akceptowanych przez komputer znaków można traktować jako szerzej pojęty alfabet. Wprowadzenie do komputera znaków w nie zmienionej postaci graficznej byłoby wygodne dla człowieka, ale wymagało by stosowania *bardzo skomplikowanych* urządzeń odczytujących, przetwarzających i zapisujących. Dlatego każdy znak jest w komputerze kodowany, czyli otrzymuje *jednoznaczną reprezentację liczbową*.

W komputerach zgodnych ze standardem IBM używa się kodu **ASCII**, w którym określono kody 256 znaków

- ASCII – American Standard Code for Information Interchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- **Rozszerzony standard ASCII: 8 bitów.**
- Różne rozszerzenia:
  DOS: Code Page, czyli strona kodowa 852, zwana Latin 2,
  Windows 3/95: CP-1250, Central-European encoding
  Oficjalny standard: ISO-8859-2.
- UNICODE – 16 bitów (2 bajty) na znak

Unicode jest nazwą dla sposobu kodowania znaków w różnych językach narodowych (czeski, polski, rosyjski, turecki, chiński ..) do binarnej formy komputerowej.

Poprzednio wszystkie języki były zapisywane do 1 bajta, tzn. że w ten sposób było możliwe kodować ogółem 256 różnych znaków. Pierwsze 128 znaków jest przepisanych przez standard i tworzy tzw. tabelę ASCII w której jest zdefiniowane przyporządkowanie wszystkich podstawowych małych (a-z) i dużych (A-Z) liter alfabetu łacińskiego bez znaków diakrytycznych, wszystkich cyfr (0,1-9), znaków specjalnych takich jak przecinek, średnik, dwukropek i dużo innych.

Pozostałe 128 możliwości jest przeznaczonych do kodowania znaków specjalnych dla poszczególnych języków narodowych. Dla języków środkowoeuropejskich są to np. znaki á, č, ü, itd., dla języka rosyjskiego są to np. znaki ф, и, б, ъ, itd. Niestety, takich znaków specjalnych dla wszystkich języków na świecie jest zbyt dużo i nie jest możliwe ich zakodowanie do pozostałych 128 możliwości. Z tego powodu powstały strony kodowe dla poszczególnych grup języków. Np. kodowanie Win-1250 dla wszystkich języków stosowanych w Europie środkowej, kodowanie Win-1251 dla wszystkich znaków Cyrylicy, itd. W ten sposób zostało rozwiązane zapisywanie tekstów dla poszczególnych języków, jednak nie było możliwe napisanie tekstu zawierającego znaki nawzajem "niekompatybilnych" języków.

Problem kodowania wszystkich znaków we wszystkich językach został rozwiązany przez wprowadzenie kodowania Unicode (patrz serwer konsorcjum Unicode www.unicode.org). To kodowanie umożliwia poprawnie zakodować wszystkie znaki na świecie. Problem został rozwiązany w ten sposób, że znaki nie są zapisywane do 1 bajta (tylko 256 możliwości) lecz są zapisywane do 2 bajtów (czyli 65536 możliwości). Ten sposób kodowania oznacza się identyfikatorem **UTF-16**.

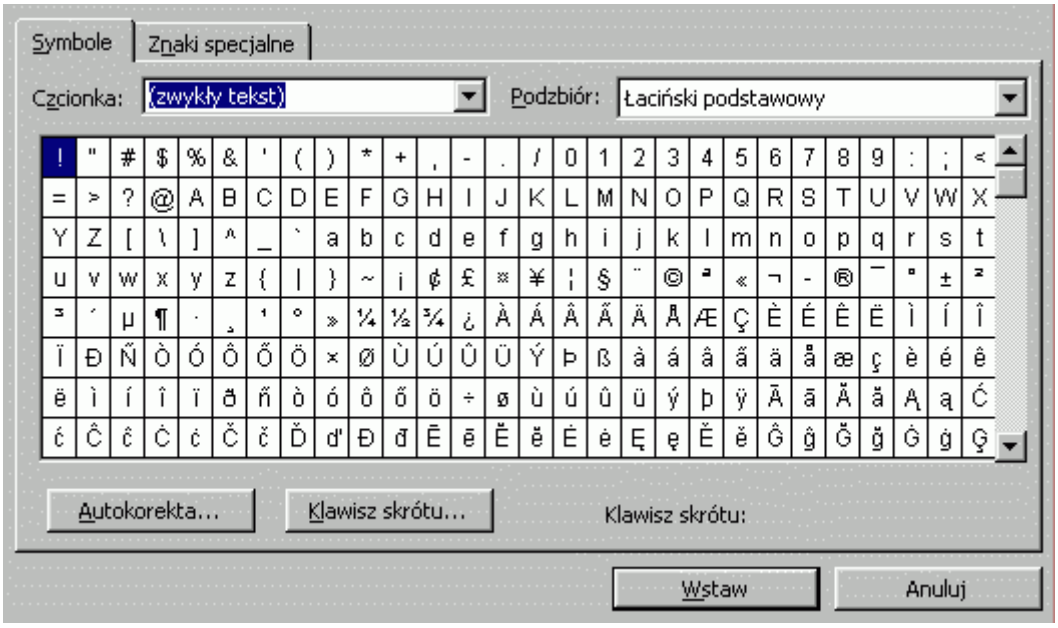
Zaletą UTF-16 jest łatwe zarządzanie wszystkimi znakami, jego mankamentem jest dwukrotna wielkość i niekompatybilność z tabelą ASCII. Taki mankament przejawia się zwłaszcza przy zapisywaniu do plików tekstowych. Z tego powodu został jeszcze zestandardyzowany następny sposób kodowania Unicode, który posiada zmienną wielkość zapisu dla poszczególnych znaków. Znaki tabeli ASCII są zapisywane w 1 bajcie, pozostałe znaki są zapisywane kolejno w 2 lub więcej bajtach (po pierwszym bajcie można rozpoznać, że wystąpi kolejny bajt). Ten sposób kodowania oznacza się identyfikatorem **UTF-8**. Jest stosowane przeważnie dla plików tekstowych (XML, HTML). Do bezpośredniej pracy w pamięci komputera tekst taki jest transformowany do UTF-16, ponieważ praca z tekstem w tym kodowaniu jest szybsza.

Aby wprowadzić znak ASCII, którego kod znamy, naciskamy klawisz Alt i z klawiatury numerycznej wprowadzamy kod znaku.

Aby wprowadzić znak Unikode, którego kod znamy, naciskamy cztery cyfry kodu znaku (można pominąć 0), a następnie Alt+x

Tablicę znaków można otworzyć w systemie Windows w następujący sposób: Kliknij przycisk Start. W polu Wyszukaj wpisz tekst Tablica znaków, a następnie na liście wyników kliknij dwukrotnie pozycję Tablica znaków.

Można również wcisnąć klawisz **Win + R** a następnie wpisać **charmap**

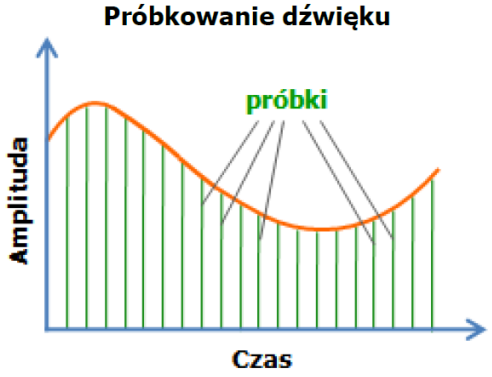


Zapis obrazu i dźwięku

Opisany kod ASCII służy do cyfrowego zapisu znaków. W komputerach mamy jednak często, oprócz znaków, do czynienia z obrazem i dźwiękiem. Aby zapisać w postaci cyfrowej obraz, komputer musi go podzielić na drobne punkty (tak, jakbyśmy nałożyli na niego kalkę z papierem milimetrowym). Każdemu z tych punktów są przypisane liczby określające jego współrzędne: X i Y oraz numer koloru. W ten sposób otrzymywany jest zapis cyfrowy obrazu. Otrzymane punkty nazywamy *pikselami*, a maksymalną ilość punktów na osi X i Y rozdzielczością poziomą i pionową lub w skrócie rozdzielczością (np. 800 x 600).

Kod RGB	Kolor		Kod RGB	Kolor
FF 00 00			FF CC 00	
00 FF 00			99 33 66	
00 00 FF			89 D1 EB	

Podobnie rzecz się ma z dźwiękiem – przetwarzany na postać cyfrową dźwięk jest *próbkowany*, tzn. dzielony z określoną częstotliwością (np. 44,1 kHz, czyli 44 100 razy na sekundę) na króciutkie fragmenty, zwane *próbkami*, których natężenie wyrażone w postaci napięcia sygnału analogowego (przed zamianą na postać cyfrową) jest przeliczane na postać binarną.



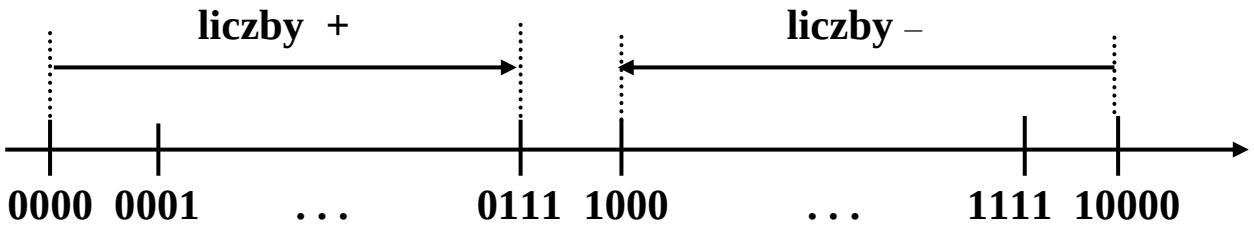
Zadanie:
Wyznaczyć ile muzyki można zapisać na płycie CD o pojemności 650 MB jeśli rejestrowane są dwa kanały (zapis stereofoniczny), a każda próbka reprezentowana jest przez 8 bitów (256 poziomów dźwięku).

DEC ➔ BIN : dzielenie przez 2

44	0
22	0
11	1
5	1
2	0
1	1
0	

$44_{10} = 101100_2$

- **uzupełnienie do 2 (Uzp2)**



0 0 0 1	+ 1	1 0 0 1	– 7
0 0 1 0	+ 2	1 0 1 0	– 6
0 0 1 1	+ 3	1 0 1 1	– 5
0 1 0 0	+ 4	1 1 0 0	– 4
0 1 0 1	+ 5	1 1 0 1	– 3
0 1 1 0	+ 6	1 1 1 0	– 2
0 1 1 1	+ 7	1 1 1 1	– 1

0 0 0 0 ➔ 0 1 0 0 0 – 8

- **liczba 4–bitowa Uzp2 ➔ – 8 ÷ + 7**
- **obliczanie liczby przeciwnej dla Uzp2**

```
    0101        + 5
~   1010
+      1
-------------
    1011        – 5
```

```
    1011        – 5
~   0100
+      1
-------------
    0101        + 5
```

- **dodawanie i odejmowanie niezależne od znaków ➔ wynik poprawny, gdy nie ma nadmiaru**

```
  0111          7
+ 1011        + (– 5 )
-------       ---------
1 0010          2
```

przeniesienie (pomijamy)

- **zamiast odejmowania dodawanie liczby przeciwnej**

4 – 6 = 4 + (– 6)

```
  0100          4
+ 1010        + (– 6 )
-------       ---------
0 1110          – 2
```

- **wykrywanie nadmiaru**
  - **dla znak – moduł nadmiar,**

    gdy przeniesienie na bit znaku = 1

$$\begin{array}{rr} 0 & 110 \\ +\ 0 & 011 \\ \hline 1 & 001 \end{array} \qquad \begin{array}{r} 6 \\ +\ 3 \\ \hline 9 \end{array}$$

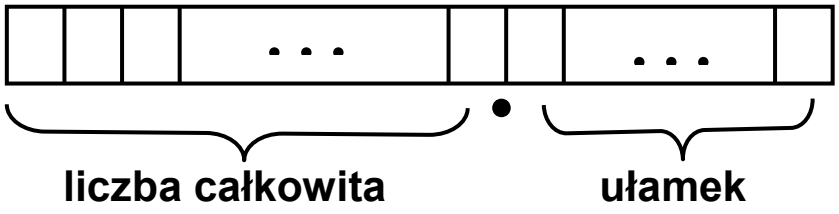
  - **dla Uzp2**

0110	6	1010	–6	0110	6	1010	–6
+0100	4	+1100	–4	+1100	–4	+0100	+4
0 | 1010		1 | 0110		1 | 0010		0 | 1110	
OV		OV		OK		OK	
$C_z = 0$		$C_z = 1$		$C_z = 1$		$C_z = 0$	
$C_s = 1$		$C_s = 0$		$C_s = 1$		$C_s = 0$	

$$C_z \oplus C_s = 1 \rightarrow \text{nadmiar}$$

Liczby niecałkowite

- **zapis stałopozycyjny**



$$\ldots a_2 p^2 + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + a_{-2} p^{-2} \ldots$$

- **konwersja binarno – dziesiętna**

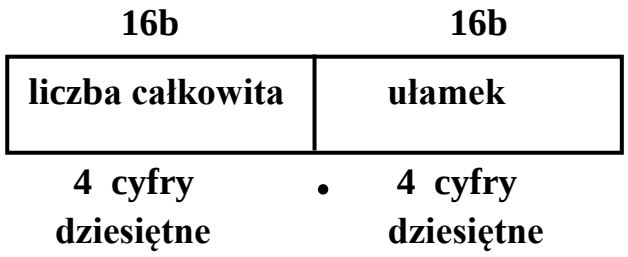
$$101.01 = 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} * 1*2^{-2} =$$

$$4 + 0 + 1 + 0 + \tfrac{1}{4} = 5\tfrac{1}{4} = 5.25$$

- **konwersja dziesiętno – binarna**

0	43 * 2
0	86
1	72
1	44
0	88
1	76
1	52
1	04
0	08

0.43 = 0.0110111 ...



- **dodawanie i odejmowanie jak dla liczb całkowitych bez znaku**
- **mnożenie i dzielenie ➔ algorytmy iteracyjne prostsze niż dla liczb zmiennopozycyjnych**

- **zapis zmiennopozycyjny**

$$\pm m * p^{\pm c}$$

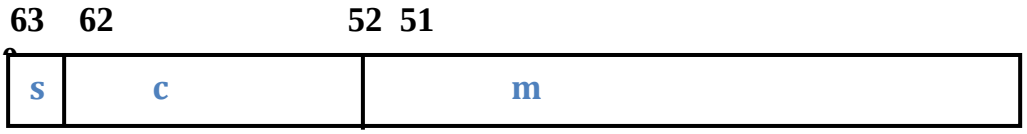
- **p = 10 $1.57*10^{12}$ 1.57E12 157xxx....xxx**

13 cyfr

- **normalizacja**

125.78E5
12.578E6 → **0.12578E8**
12578E3

- **p = 2 , standard IEEE**



cecha przesunięta

01...00 ➔ – 3
01...01 ➔ – 2
01...10 ➔ – 1
01...11 ➔ 0
10...00 ➔ 1
10...01 ➔ 2
10...10 ➔ 3
10...11 ➔ 4

znak liczby
0 ➔ +
1 ➔ –

- **mantysa znormalizowana (m) ma zawsze postać**

1.xxxxxxxx....xxxxx (1.0...0 ÷ 1.1..1)

początkowa jedynka nie jest pamiętana

$$1.0_{10} \leq m < 2.0_{10}$$

- **przykłady**

0_{10} ➔ 0 00000000000 00..00

1.0_{10} ➔ 0 01111111111 00...00

-1.0_{10} ➔ 1 01111111111 00...00

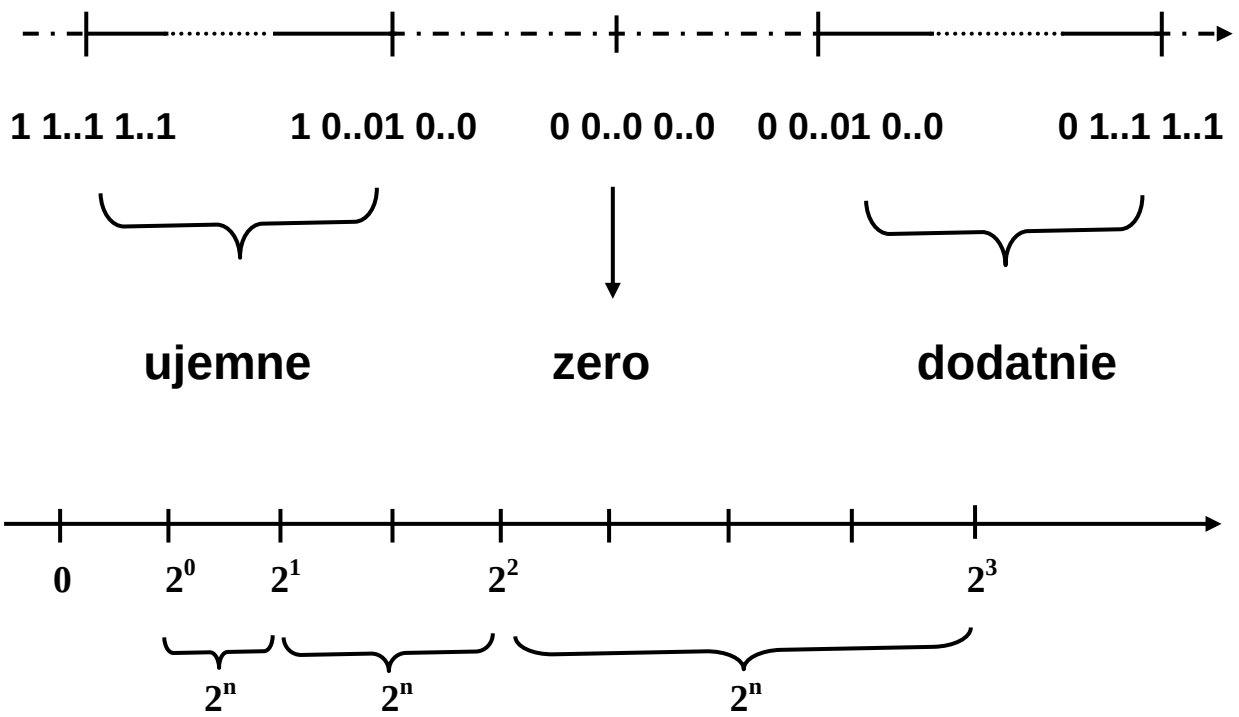
10_{10} ➔ 0 10000000010 0100..00

-10_{10} ➔ 1 10000000010 0100..00

15.5 ➔ 0 10000000010 111100..00

–15.5 ➔ 1 10000000010 111100..00

- **reprezentowane wartości**



n ➔ liczba cyfr mantysy

- **dla n = 52 odległość pomiędzy reprezentowanymi liczbami wynosi:**

c = 0 ➔ 2.22044604925e-16

c = 1 ➔ 4.440892098501e-16

c = 52 ➔ 1

c = 150 ➔ 3.169126500571e+29

c = 1023 ➔ 1.995840309535e+292

- **typy liczb zmiennopozycyjnych**

float :	**32b**	**1 / 8 / 23**	**±3.4 E ±38**	**7 cyfr**
double :	**64b**	**1 / 11 / 52**	**±1.7 E ±308**	**15 cyfr**
long double:	**80b**	**1 / 15 / 64**	**±1.2 E ±4932**	**17 cyfr**

Arytmetyka zmiennopozycyjna

- **<u>mnożenie</u> ➔ mnożenie mantys, sumowanie cech, normalizacja**
- **<u>dzielenie</u> ➔ dzielenie mantys, odejmowanie cech, normalizacja**
- **<u>dodawanie</u> ➔ wyrównanie cech, dodawanie, normalizacja**
- **<u>odejmowanie</u> ➔wyrównanie cech, odejmowanie, normalizacja**

- **dla 3 cyfr dokładnych**

  $0.375 * 10^5 + 0.25 * 10^{-2} =$

  $0.375 * 10^5 + 0.000000025 * 10^5 =$

  $0.375000025 * 10^5 \cong 0.375 * 10^5$

- **sposób sumowania trzech liczb zmiennopozycyjnych aby błąd był najmniejszy**
- **obliczenia zmiennopozycyjne na odpowiedzialność programisty**