Technologie Informacyjne i Komunikacyjne 2023/2024

dr Magdalena Posiadała-Zezula

Wykład 4:
Reprezentacje liczb i znaków

Photo Credit: Piotr Mijakowski

Reprezentacje liczb i znaków

- Liczby:
  - Reprezentacja „naturalna” – nieujemne liczby całkowite – naturalny system dwójkowy.
  - **Reprezentacje „umowne” – liczby ujemne, liczby niecałkowite**
- Znaki:
  - **Tylko reprezentacje „umowne” – zbiory znaków (ang. character set).**
  - **ASCII (m. in. 1000001_{bin} = A, 1000010_{bin} = B itd.)**
  - **Strony kodowe, standardy ISO-8859, Unicode.**

Reprezentacja liczb rzeczywistych - przypomnienie!

$125_{(10)}=?_{(2)}$

Liczby całkowite :
Operacja modulo % – reszta z dzielenia:

125%2=62 reszta 1
62% 2= 31 reszta 0
31% 2=15 reszta 1
15%2=7 reszta 1
7%2=3 reszta 1
3%2=1 reszta 1
1%2= 0 reszta 1

Część całkowita: spisujemy **od DOŁU !!!**
$125_{(10)}=1111101_{(2)}$!!!

$125.40625_{(10)}=?_{(2)}$

Część całkowita- operacja modulo %
Część ułamkowa- operacja mnożenia * 2

0.40625*2 = **0**.8125
0.8125*2 = **1**.625
0.625 *2 = **1**.25
0.25*2 = **0**.5
0.5*2 = **1**.0

Część ułamkowa: spisujemy **od GÓRY!!!**

$125.40625_{(10)}=1111101.01101$!!!

Reprezentacja zmiennoprzecinkowa

- **Liczby zmiennoprzecinkowe (ang. floating point numbers)** są komputerową reprezentacją liczb rzeczywistych zapisanych w formie wykładniczej **(naukowej).**

- Np. stała grawitacji:

$$G = 6.67 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2} = 66.7 \cdot 10^{-10} \frac{m^3}{kg \cdot s^2} = 667 \cdot 10^{-9} \frac{m^3}{kg \cdot s^2}$$

- Położenie przecinka nie jest ustalone i może się dowolnie zmieniać.

Reprezentacja zmiennoprzecinkowa

- **Liczby zmiennoprzecinkowe (ang. floating point numbers)**

$$G = 6.67 \cdot 10^{-11} \frac{m^2}{kg \cdot s^2}$$

- Zapis składa się z 3 liczb:
  - **mantysy (m)**, tu równej 6.67,
  - **podstawy (p)**, tu równej 10, w syst. binarnym równej 2.
  - **cechy (c)**, tu równej -11- dla postaci znormalizowanej.

Reprezentacja zmiennoprzecinkowa

- **Liczby zmiennoprzecinkowe (ang. floating point numbers)**

$$G = 6.67 \cdot 10^{-11} \frac{m^2}{kg \cdot s^2}$$

- Liczbę zmiennoprzecinkową można zapisywać w różny sposób, przyjęto tzw. **postać znormalizowaną:**
- Znormalizowana liczba zmiennoprzecinkowa to taka, w której mantysa spełnia nierówność: $p > |m| \geq 1$.

Wartość liczby zmiennoprzecinkowej

$$\text{Liczba}_{(\text{FP})} = m \cdot p^{c}$$

gdzie : m – mantysa, p- podstawa, c- cecha liczby.

➢W systemie dwójkowym wszystkie trzy elementy m, p i c będą zapisane dwójkowo za pomocą odpowiednio dobranego systemu kodowania liczb. Podstawa p zawsze będzie równa 2, zatem wzór obliczeniowy przyjmie postać:

$$\text{Liczba}_{(\text{FP})} = m \cdot 2^{c}$$

Wartość liczby zmiennoprzecinkowej

- Do określenia formatu zmiennoprzecinkowego pozostaje nam podanie sposobu kodowania mantysy i cechy, ponieważ podstawa p=2 będzie zawsze znana.

$$\text{Liczba}_{(\text{FP})} = m \cdot 2^{c}$$

Wartości specjalne liczb zmiennoprzecinkowych

1. **NaN** – nie-liczba (ang. Not-a-Number), to symbol, który nie reprezentuje wartości liczbowej, powstały zazwyczaj w wyniku niedozwolonej operacji (np. pierwiastkowanie liczby ujemnej)
2. **Zero – rozróżnia się +0,0 i -0,0.**
3. **Nieskończoność (∞) –** jest wynikiem operacji w przypadku wystąpienia nadmiaru (przepełnienia), przy dzieleniu przez 0, itp.; może być dodatnia lub ujemna.

Standard IEEE 754

- W celu ujednolicenia zasad operacji na liczbach zmiennoprzecinkowych na różnych platformach sprzętowych, opracowano **standard IEEE 745**.
- Definiuje on dwie klasy liczb:
  - **pojedynczej precyzji -** 32-bitowe (ang. single precision)
  - **podwójnej precyzji -** 64-bitowe (ang. double precision)

Standard IEEE 754
pojedyncza precyzja

32 bity- pojedyncza precyzja	1 bit b_{31}	(8 bitów) b_{30} ... b23 (nadmiar=127)	(23 bity) b_{22} ... b0 (U1)
	bit znaku	bit kodu cechy	bit kodu mantysy

★Pierwszy bit w zapisie liczby zwany jest **bitem znaku**. Stan 0 oznacza liczbę dodatnią, stan 1 liczbę ujemną. Aby zatem zmienić znak liczby zmiennoprzecinkowej na przeciwny, wystarczy dokonać negacji tego bitu.

★W pojedynczej precyzji cecha posiada 8 bitów, a nadmiar wynosi 127. W polu cechy można zapisać wartości od -127 (wszystkie bity $b_{30}...b_{23}$ wyzerowane) do 128 (wszystkie bity $b_{30}...b_{23}$ ustawione na 1).

★W pojedynczej precyzji mantysa posiada 23 bity. Mantysy są zapisywane w stałoprzecinkowym kodzie U1. Moga byc też zapisywane w U2,ZM. Ponieważ mantysa jest prawie zawsze znormalizowana, to jej wartość zawiera się miedzy 1 a 2. Dlatego jej pierwszy bit całkowity zawsze wynosi 1 i nie musi być zapamiętany- pomysł użyty już w 1936 roku przez niemieckiego konstruktora kmputerów Konrada Zuse'a.

Standard IEEE 754
podwójna precyzja

64 bity - podwójna precyzja	1 bit b_{63}	(11 bitów) b_{62} ... b52 (nadmiar=1023)	(52 bity) ,b_{51} ... b0 (U1)
	bit znaku	**bit kodu cechy**	**bit kodu mantysy**

★Pierwszy bit w zapisie liczby zwany jest **bitem znaku**. Stan 0 oznacza liczbę dodatnią, stan 1 liczbę ujemną. Aby zatem zmienić znak liczby zmiennoprzecinkowej na przeciwny, wystarczy dokonać negacji tego bitu.

★W podwójnej precyzji cecha zbudowana jest z 11 bitów, nadmiar wynosi 1023. Najmniejszą wartością będzie -1023 (bity b_{62}...b_{52} ustawione na 0) , a największą 1024 (bity b_{62}...b_{52} ustawione na 1).

★W podwójnej precyzji mantysa posiada 52 bity. Mantysy są zapisywane w stałoprzecinkowym kodzie U1. Moga byc też zapisywane w U2, ZM.

Wartość liczby IEEE 754

➢ Wartość liczby **IEEE 754** obliczamy stosując poniższe kroki :

1. Z kodu wydzielamy poszczególne pola znaku z, cechy c i mantysy m.
2. Odczytana mantysa zawiera tylko bity ułamkowe. Dodajemy zatem na początku 01 i przecinek. W wyniku otrzymujemy dodatnią liczbę stałoprzecinkową w kodzie U1.
3. Obliczamy wartość cechy i mantysy.
4. Wyliczamy wartość liczby wg wzoru:

$$L_{10} = (-1)^{z} 2(c - 127)(1.m)$$

Wartość liczby IEEE 754- -przykład (1)

Obliczyć wartość dziesiętną liczby zmiennoprzecinkowej :
$11000001110110000000000000000000_{(IEEE\ 754)}$.
Kod binarny dzielimy na poszczególne pola zawierające kolejno znak, cechę oraz bity ułamkowe mantysy:

1 10000011 10110000000000000000000

z cecha bity ułamkowe mantysy

z = 1 - liczba jest ujemna

$c = 10000011_{(BIAS=127)} = 131 - 127 = 4$

$m = 01,10110000000000000000000_{(U1)} =$

$1^{11}/_{16}$

Przykład zaczerpnięty z http://edu.i-lo.tarnow.pl/inf/alg/006_bin/0022.php

Wartość liczby IEEE 754- -przykład (2)

$$L_{(IEEE\ 754)} = (-1)^z m 2^c = (-1)^1 \times 1^{11}/_{16} \times 2^4$$

$$= -\,^{27}/_{16} \times 2^4 = -27_{(10)}$$

$$11000001101100000000000000000000_{(IEEE\ 754)} = -27_{(10)}.$$

Przykład zaczerpnięty z http://edu.i-lo.tarnow.pl/inf/alg/006_bin/0022.php

Notacja szesnastkowa (1)

➢ **Szesnastkowy system liczbowy, system heksadecymalny, hex (ang. *hexadecimal)*** – pozycyjny system liczbowy, w którym podstawą jest liczba 16.

➢Do zapisu liczb w tym systemie potrzebne jest szesnaście znaków (cyfr szesnastkowych).

➢Poza cyframi dziesiętnymi od **0 do 9** używa się pierwszych sześciu liter alfabetu łacińskiego: **A, B, C, D, E, F** (wielkich lub małych).

➢Cyfry 0-9 mają te same wartości co w systemie dziesiętnym.

➢Litery odpowiadają następującym wartościom:

➢ **A = 10, B = 11, C = 12, D = 13, E = 14 , F = 15**.

Notacja szesnastkowa (2)

Przykłady liczb przedstawionych w kodzie heksadecymalnym:

1) $1C = 1*16^1 + 12*16^0 = 16+12=28$
2) $10F = 1*16^2 + 0*16^1 + 15*16^0 = 271$

Konwersja dwójkowo – szesnastkowa (1)

➢ Liczbę dwójkową dzielimy na grupy 4-ro bitowe idąc od strony prawej ku lewej.
➢Jeśli w ostatniej grupie jest mniej bitów, to brakujące wypełniamy zerami.

11101010001010101111010101 01

1110 1010 0010 1010 1111 0101 0101
E A 2 A F 5 5

$11101010001010101111010101_{(2)} = EA2AF55_{(16)}$.

Przykład zaczerpnięty z http://edu.i-lo.tarnow.pl/inf/alg/006_bin/0022.php

Konwersja dwójkowo – szesnastkowa (2)

➢Konwersja z liczby szesnastkowej na liczbę binarną:

1. Każdą cyfrę szesnastkową zastępujemy grupą 4 bitów według zasady konwersji**: Każda cyfra w notacji szesnastkowej odpowiada czterem bitom w zapisie binarnym tej samej liczby**.

0	0000	**1**	0001	**2**	0010	**3**	0011
4	0100	**5**	0101	**6**	0110	**7**	0111
8	1000	**9**	1001	**A**	1010	**B**	1011
C	1100	**D**	1101	**E**	1110	**F**	1111

2. Grupy łączymy w całość otrzymując odpowiednik dwójkowy wyjściowej liczby szesnastkowej.

Zadania

Zad 1. Wyrazić liczbę w systemie szesnastkowym: 1234?

Zad2. Wyrazić liczbę w systemie dwójkowym: **3FAC72608E**?

Zad3. Wyrazić liczbę w systemie szesnastkowym $10110101111010101111_{(2)}$?

Notacja szesnastkowa – języki programowania

➢W różnych językach programowania zapis liczb szesnastkowych wygląda różnie:

➢**C, C++, Java, Python** - stosuje się **prefiks 0x** (zero oraz x) np. 0x102f, a w ciągach tekstowych \x (ukośnik oraz x) np. \x102f. Np w języku Python: **hex(123) # zwróci '0x7b' 7b=7*16^1+11*16^0= 112+11=123.**

➢**Pascal** - stosuje się prefiks $, np. $102f

Czym są zbiory znaków?

- Czym są zbiory znaków?
- **Zbiór znaków (ang. character set)**
  - System, zgodnie z którym symbole graficzne z pewnego zbioru (znaki) są reprezentowane w określony sposób przez ciągi bitów (słowa). Definiuje także przyporządkowanie symbolom określonych wartości liczbowych (kody znaków).

- **Kodowanie (ang. encoding)**

  Zastępowanie symboli z określonego zbioru (np. graficznych) symbolami z zbioru (np. liczby) zgodnie z ustalonymi zasadami. **Nie ma na celu ukrywania informacji.**

- Istnieją „niecyfrowe” systemy o takim charakterze:
  1. Alfabet Morse’a (ang. Morse code).
  2. Pismo punktowe Braille’a.

Kod ASCII

ASCII (American Standard Code for Information Interchange)

➢ Najstarszy standardowy system reprezentowania znaków z pomocą ciągów bitów (słów).
➢Zbiór znaków określający reprezentacje binarne oraz (w związku z tym) liczbowe dla 128 znaków (kody od 0 do 127) – 7 bitów.
➢Obejmuje litery angielskiego alfabetu (wielkie i małe), cyfry, znaki przestankowe i symbole matematyczne, symbole specjalne (np. $) i tak zwane znaki sterujące (ang. control characters).
➢Na przykład litera "a" jest kodowana liczbą 97, a znak spacji jest kodowany liczbą 32.

Znaki sterujące (kody ASCII od 0 do 31)

➢ Reprezentują operacje, np. przejście do następnego wiersza (ang. line feed) lub przesunięcie głowicy drukującej na początek wiersza (ang. carriage return) albo przesunięcie jej o jedną pozycję wstecz (ang. backspace). Wykorzystywane m. in. do formatowania tekstu.

Kod ASCII- tabela

➢**0 – 31 – kody sterujące**.

➢**Małe litery i duże różnią się o stałą wartość liczby 32.**
Np. A=65
a=65+32=97

Char	Ctrl	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
NUL	^@	0	00	space	32	20	@	64	40	`	96	60
SOH	^A	1	01	!	33	21	A	65	41	a	97	61
STX	^B	2	02	"	34	22	B	66	42	b	98	62
ETX	^C	3	03	#	35	23	C	67	43	c	99	63
EOT	^D	4	04	$	36	24	D	68	44	d	100	64
ENQ	^E	5	05	%	37	25	E	69	45	e	101	65
ACK	^F	6	06	&	38	26	F	70	46	f	102	66
BEL	^G	7	07	'	39	27	G	71	47	g	103	67
BS	^H	8	08	(	40	28	H	72	48	h	104	68
HT	^I	9	09	)	41	29	I	73	49	i	105	69
LF	^J	10	0A	*	42	2A	J	74	4A	j	106	6A
VT	^K	11	0B	+	43	2B	K	75	4B	k	107	6B
FF	^L	12	0C	,	44	2C	L	76	4C	l	108	6C
CR	^M	13	0D	-	45	2D	M	77	4D	m	109	6D
SO	^N	14	0E	.	46	2E	N	78	4E	n	110	6E
SI	^O	15	0F	/	47	2F	O	79	4F	o	111	6F
DLE	^P	16	10	0	48	30	P	80	50	p	112	70
DC1	^Q	17	11	1	49	31	Q	81	51	q	113	71
DC2	^R	18	12	2	50	32	R	82	52	r	114	72
DC3	^S	19	13	3	51	33	S	83	53	s	115	73
DC4	^T	20	14	4	52	34	T	84	54	t	116	74
NAK	^U	21	15	5	53	35	U	85	55	u	117	75
SYN	^V	22	16	6	54	36	V	86	56	v	118	76
ETB	^W	23	17	7	55	37	W	87	57	w	119	77
CAN	^X	24	18	8	56	38	X	88	58	x	120	78
EM	^Y	25	19	9	57	39	Y	89	59	y	121	79
SUB	^Z	26	1A	:	58	3A	Z	90	5A	z	122	7A
ESC	^[	27	1B	;	59	3B	[	91	5B	{	123	7B
FS	^\	28	1C	<	60	3C	\	92	5C	|	124	7C
GS	^]	29	1D	=	61	3D	]	93	5D	}	125	7D
RS	^^	30	1E	>	62	3E	^	94	5E	~	126	7E
US	^_	31	1F	?	63	3F	_	95	5F	delete	127	7F

Magdalena.Posiadala@fuw.edu.p

Kod ASCII- przykład

Ala ma kota.

01000001011011000110000100100000011011010110000100100000
01101011011011110111010001100001 00101110
decymalny zapis
651089732109732107111169746

Kod ASCII -ograniczenia

- Zbiór znaków ASCII nie wystarcza do wszystkich zastosowań:
  - Litery akcentowane.
  - Litery z innych alfabetów (np. cyrylica, alfabet grecki).
  - Znaki nie będące literami (np. symbole matematyczne).
  - Języki, w których nie używa się liter.

Dodatkowo zakładano, że każdy znak będzie zajmował taką samą przestrzeń na wydruku lub na ekranie terminala (tzn. do drukowania każdego znaku był przeznaczony prostokąt o określonej wysokości i szerokości jednakowej dla wszystkich znaków).

Kod ASCII- rozszerzenia

➢ Ponieważ kod **ASCII jest 7-bitowy**, a większość komputerów operuje na 8-bitowych bajtach, **dodatkowy bit** można wykorzystać na powiększenie zbioru kodowanych znaków do **256 symboli**.

➢ Powstało wiele różnych rozszerzeń ASCII wykorzystujących ósmy bit:

1. rozszerzenia firm IBM lub Microsoft nazywanych stronami kodowymi.
2. **norma ISO 8859**.

Strony kodowe IBM

- **Strona kodowa (ang. code page)**
  - Zbiór znaków, w którym znaki o kodach 0 . . . 127 są zgodne z ASCII
  - Pozostałe kody oznaczają znaki spoza zbioru ASCII

- **Strony kodowe IBM**
  - Zaprojektowane dla zgodności ze sprzętem („znakowe” tryby działania kart graficznych).
  - Przykłady:
    - 850 – Multilingual (Latin-1): **języki zachodnioeuropejskie.**
    - 852 – Multilingual (Latin-2): **języki środkowo- i wschodnioeuropejskie.**
    - 855 – Cyrylica

Strony kodowe Microsoftu (Windows)

- **Strony kodowe ANSI**
  - W Windows występuje odmiana kodów ASCII zwana ANSI (Amerykański Narodowy Instytut Standardów)

.

  - Przykłady:
    - CP1250 – Latin-2: języki środkowo- i wschodnioeuropejskie.
    - CP1251 – Cyrylica.
    - CP1252 – Latin-1: języki zachodnioeuropejskie.

Standardy ISO-8859

➢ **ISO/IEC 8859 to zestaw standardów służących do kodowania znaków za pomocą 8 bitów.**

➢Standardy te zostały utworzone przez **ECMA (European Computer Manufacturers Association)** w połowie lat osiemdziesiątych, po czym zostały uznane za ISO.

➢Wszystkie zestawy **ISO 8859 mają znaki 0-127 (hex 00-7F)** takie same jak ASCII, zaś **pozycjom 128-159 (hex 80-9F**) przypisane są dodatkowe kody sterujące.

➢Przykłady :

- ➢ISO 8859-1 (Latin-1) - alfabet łaciński dla Europy zachodniej
- ➢ISO 8859-2 (Latin-2) - łaciński dla Europy środkowej i wschodniej, również odpowiednia Polska Norma
- ➢ISO 8859-3 (Latin-3) - łaciński dla Europy południowej
- ➢ISO 8859-4 (Latin-4) - łaciński dla Europy północnej
- ➢ISO 8859-5 (Cyrillic) - dla cyrylicy

Standardy ISO-8859

- **ISO-8859-1**
  - Znaki wykorzystywane w językach zachodnioeuropejskich.
  - Brak znaków „akcentowanych” z języka polskiego.
- **ISO-8859-2**
  - Znaki wykorzystywane w językach środkowoeuropejskich.
  - Kody 128 . . . 255 przypisane innym znakom, niż w ISO-8859-1.
- **ISO-8859-15**
  - Rewizja ISO-8859-1 wprowadzająca znak waluty euro.

Standard Unicode

➢**Standard Unicode**

➢zestaw znaków mający w zamierzeniu obejmować wszystkie pisma używane na świecie.

➢Definiują go dwa standardy – Unicode oraz ISO 10646.

➢Znaki obu standardów są identyczne.

➢Standard Unicode obejmuje przydział przestrzeni numeracyjnej poszczególnym grupom znaków, **nie obejmuje zaś sposobów bajtowego kodowania znaków**. **Koniec zasady „1 znak – 1 bajt**.

➢Jest kilka metod kodowania, oznaczanych skrótowcami **UCS (Universal Character Set) i UTF (Unicode Transformation Format)**. Do najważniejszych należą:

➢ UTF-32/UCS-4

➢UTF-16

➢UTF-8.

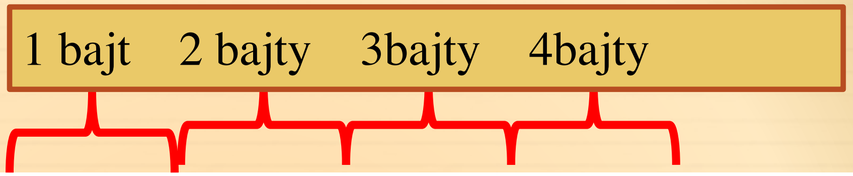
UTF-8 (8-bit Unicode Transformation Format)

➢UTF-8 – system kodowania Unicode, wykorzystujący od 8 do 32 bitów do zakodowania pojedynczego znaku, w pełni kompatybilny z ASCII.

➢Znaki z przedziału ASCII (0 do 127) kodowane są jako jeden bajt, czyli m.in. litery alfabetu łacińskiego.
➢Polskie znaki diakrytyczne kodowane już są jako dwa bajty.
➢Najbardziej popularny format Unicode
➢Budowa kodowania UTF-8 pozwala przypadkowo zapisać ten sam znak na kilka różnych sposobów, ale standard mówi, że poprawnym zapisem jest tylko najkrótszy z możliwych.


Thank you!
Photo Credit: Piotr Mijakowski
ν_e
ν_μ
ν_τ

Mapowanie znaków Unicode na ciągi bajtów UTF-8



- 0x00 do 0x7F – bity 0xxxxxxx, gdzie kolejne „x" to bity – licząc od najwyższego
- 0x80 do 0x7FF – bity 110xxxxx 10xxxxxx
- 0x800 do 0xFFFF – bity 1110xxxx 10xxxxxx 10xxxxxx
- 0x10000 do 0x1FFFFF – bity 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
- 0x200000 do 0x3FFFFFF – bity 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
- 0x4000000 do 0x7FFFFFFF – bity 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

➢ Liczby po lewej to w systemie heksadecymalnym podane liczby zakresu kodowania. Np 0x00=0, zaś 0x7F= 7*16^1+15^16^0=112+15=127

➢Liczby na prawo to sposob mapowania znaków w UTF-8.
➢Np liczby w zakresie od 0-127 maja zapis **0**xxxxxxx gdzie każdy x to kolejne bity liczby w systemie dwojkowym. Np: 9=**0**0001001

Kodowanie UTF-8

Bity	Największa wartość	Bajt 1	Bajt 2	Bajt 3	Bajt 4
7	U+007F	0xxxxxxx			
11	U+07FF	110xxxxx	10xxxxxx		
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Kodowanie UTF-8 przykłady

Znak	Kod		Bin	Hex	Dec
@	U+0040	100 0000	0100 0000	40	64
Ą	U+0104	001 0000 0100	1100 0100 1000 0100	c4 84	50308
龜	U+2EF1	0010 1110 1111 0001	1110 0010 1011 1011 1011 0001	e2 bb b1	??
𐌄	U+10304	0 0001 0000 0011 0000 0100	1111 0000 1001 0000 1000 1100 1000 0100	f0 90 8c 84	??