Úvod.

Čísla a číselné soustavy.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK.

Obsah přednášky

1. Úvodní informace o předmětu
2. Číselné soustavy a jejich charakteristika
3. Přehled číselných soustav
4. Převody mezi číselnými soustavami
5. Znázorňování čísel se znaménky
6. Operace s čísly
7. Reprezentace celých čísel
8. Reprezentace reálných čísel
9. Zákonitosti při práci s reálnými čísly

1. Plán přednášek:

1. Úvod (1 hodina).
2. Čísla a číselné soustavy (1 hodina).
3. Booleovská algebra (1 hodina).
4. Algoritmy a jejich vlastnosti, programovací jazyky (1 hodina).
5. Datové struktury, podmínky, cykly, metody (2 hodiny).
6. Základní algoritmické techniky (2 hodiny).
7. Třídící algoritmy (2 hodiny).
8. Objektově orientované programování (2 hodiny).
9. Algoritmy numerické matematiky (podle času).

2. Literatura:

- Přednášky:

[1] VIRIUS M.: Základy algoritmizace, 2004,Vydavatelství ČVUT
[2] WROBLEWSKI P.: Algoritmy,datové struktury a programovací techniky, 2004,Computer Press
[3] JOKL E.,ŠIBRAVA Z.,VOSPĚL Z.: Programování1,1990,Vydavatelství ČVUT
[4] KOLÁŘ J.: Teoretická informatika, 2004, Vydavatelství ČVUT

- Cvičení:

[1] Eckel B. : Myslíme v jazyce Java, Grada Publishing, 2000
[2] Herout P.: Java, grafické uživatelské prostředí a čeština, Kopp, 2001
[3] Herout P.: Učebnice jazyka Java, Java, Kopp, 2001
[4] Chapman S. J.: Začínáme programovat v jazyce Java, Computer Press, 2001
[5] Pecinovský R.: Myslíme objektově v jazyku Java, Grada publishing, 2006
[6] Virius M.: Java pro zelenáče, Neocortex, 2001

Elektronická verze cvičení:

`http://www.natur.cuni.cz/~bayertom/Prog1/java_rukopis.pdf`

3. Číselné soustavy:

Dělení do dvou skupin:

- polyadické soustavy,
- nepolyadické soustavy.

Polyadické soustavy:

Označovány jako soustavy poziční, hodnota číslice je dána její pozicí v čísle. V běžném životě i v informatice často používány.

Nepolyadické soustavy:

Hodnota číslice není odvozována od její pozice v čísle, ale jiným, často speciálním postupem. V informatice nejsou využívány, složitá konstrukce čísel a realizace aritmetických operací.

- Příklad:

Římské číslice: 2007=MMVIII. 1955=MCMLV

4. Polyadické soustavy:

Základ soustavy z:

Každá polyadická soustava má celočíselný základ z, $z > 1$.

Nejčastěji používané polyadické soustavy:

Dvojková (binární) –>$z = 2$, osmičková (oktalová) ->$z = 8$, desítková (dekadická) ->$z = 10$, šestnáctková (hexadecimální) ->$z = 16$.

Dělení polyadických soustav:

Nejčastější dělení do dvou skupin:

- *Standardní soustavy*
  Jeden celočíselný základ z, rovnoměrné dělení. Viz výše uvedené soustavy.
- *Nestandardní soustavy*
  Více číselných základů, nerovnoměrné dělení, např. soustava pro určování času: 24h, 60min, 60s. Příliš často se nepoužívají.

5. Vyjadřování čísel v polyadické soustavě:

Libovolné číslo a lze vyjádřit prostřednictvím polynomu. Proto soustavy nazýváme *polyadické*. Koeficienty polynomu $b_i \leq z - 1$.

Číslo a lze zapsat jako součet celočíselné části $\bar{a}$ a desetinné části $\tilde{a}$ ve tvaru:

$$a = \bar{a} + \tilde{a} \tag{1}$$

Celočíselná část čísla $\bar{a}$:

$$\bar{a} = \pm \sum_{i=0}^{n} b_i z^i = \pm(b_n z^n + b_{n-1} z^{n-1} + b_{n-2} z^{n-2} + \ldots + b_0 z^0). \tag{2}$$

Desetinná část čísla $\tilde{a}$:

$$\tilde{a} = \pm \sum_{i=-1}^{-m} b_i z^i = \pm(b_{-1} z^{-1} + b_{-2} z^{-2} + b_{-3} z^{-3} + \ldots + b_{-m} z^{-m}). \tag{3}$$

- Příklad:
  1997=1*10^3+9*10^2+9*10^1+7*10^0 $\Rightarrow z = 10, b_3 = 1, b_2 = 9, b_1 = 9, b_0 = 7.$
  3.14159=3*10^0+1*10^-1+4*10^-2+1*10^-3+5*10^-4+9*10^-5
  $\Rightarrow z = 10, b_0 = 3, b_{-1} = 1, b_{-2} = 4, b_{-3} = 1, b_{-4} = 5, b_{-5} = 9.$

6. Dvojková (binární) soustava.

Základ $z = 2$, vyjádření čísla prostřednictvím mocnin 2.
Koeficienty b_0, b_1 nabývají hodnot 0,1. Nejčastěji používaná polyadická soustavou v informatice, počítače nativně pracují s dvojkovou soustavou. Hodnoty odpovídají dvěma stavům: prochází nebo neprochází elektrický proud.

$$\overline{a} = \sum_{i=0}^{n} b_i 2^i = (b_n 2^n + b_{n-1} 2^{n-1} + b_{n-2} 2^{n-2} + \ldots + b_0 2^0). \qquad (4)$$

- Příklad:
  Vyjádření $a = (41.125)_{10}$ v binární soustavě.

i	5	4	3	2	1	0	-1	-2	-3
z^i	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
	32	16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
b	1	0	1	0	0	1	0	0	1
$b \cdot z^i$	32	0	8	0	0	1	0	0	0.125
$(a)_{10}$	32+8+1+0.125=41.125								

7. Desítková (decimální) soustava

Základem je $z = 10$, vyjádření čísla prostřednictvím mocnin 10.
Koeficienty $b_0, ..., b_9$, nabývají hodnot $\langle 0, ..., 9 \rangle$. V této soustavě počítá člověk, počítače však neumí provádět aritmetické operace v desítkové soustavě přímo.

$$\bar{a} = \pm \sum_{i=0}^{n} b_i 10^i = \pm(b_n 10^n + b_{n-1} 10^{n-1} + b_{n-2} 10^{n-2} + ... + b_0 10^0). \qquad (5)$$

- Příklad:
  Vyjádření $a = 506007.008$ v desítkové soustavě.

i	5	4	3	2	1	0	-1	-2	-3
z^i	10^5 100 000	10^4 10 000	10^3 1000	10^2 100	10^1 10	10^0 1	10^{-1} $\frac{1}{10}$	10^{-2} $\frac{1}{100}$	10^{-3} $\frac{1}{1000}$
b	5	0	6	0	0	7	0	0	8
$b \cdot z^i$	500 000	0	6000	0	0	7	0	0	0.008
$(a)_{10}$	500 000+6000+7+0.008=506 007.008								

8. Šestnáctková (hexadecimální) soustava.

Základem je číslo $z = 16$, vyjádření čísla prostřednictvím mocnin 16. Koeficienty $b_0, ..., b_{15}$ nabývají hodnot $\langle 0, ..., 9, a, ..., f \rangle$. Tato soustava je používána v nižších programovacích jazycích pro zápis instrukcí.

$$\bar{a} = \pm \sum_{i=0}^{n} b_i 16^i = \pm(b_n 16^n + b_{n-1} 16^{n-1} + b_{n-2} 16^{n-2} + ... + b_0 16^0). \qquad (6)$$

- Příklad:
  Vyjádření $a = 10542017.0007$ v šestnáctková soustavě.

i	5	4	3	2	1	0	-1	-2	-3
z^i	16^5 1 048 576	16^4 65 536	16^3 4096	16^2 256	16^1 16	16^0 1	16^{-1} $\frac{1}{16}$	16^{-2} $\frac{1}{256}$	16^{-3} $\frac{1}{4096}$
b	a	0	f	0	0	1	0	0	3
$b \cdot z^i$	10 480 576	0	61 440	0	0	1	0	0	0.0007
$(a)_{10}$	10 480 576+61 440+1+0.0007$\doteq$10 542 017.0007								

9. Převod do desítkové soustavy:

Princip převodu do desítkové soustavy velmi jednoduchý, představuje dosazení do (2) nebo (3).
Postup se snadno algoritmizuje.

- **Příklad:**
  Převod z dvojkové do desítkové soustavy:

  $$(0101,01)_2 \Rightarrow a = 0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0+0\cdot 2^{-1}+1\cdot 2^{-2} = (5.25)_{10}.$$

  Převod z šestnáctkové do desítkové soustavy:

  $$(12af)_{16} \Rightarrow a = 1\cdot 16^3 + 2\cdot 16^2 + 10\cdot 16^1 + 15\cdot 16^0 = (4783)_{10}\,.$$

10. Převod z desítkové soustavy:

Převod provádíme postupným odečítáním co nejvyšších mocnin základu soustavy z, do které číslo a převádíme $\Rightarrow$ pracujeme se zbytkem čísla a po dělení z^k.
Dělení provádíme opakovaně, dokud zbytek po dělení není roven nule nebo nedosáhneme požadované přesnosti.

- **Příklad:**
  Převod z desítkové soustavy do dvojkové:
  $(105.50)_{10} = 1 \cdot 2^6 - 1 \cdot 2^5 - 1 \cdot 2^3 - 1 \cdot 2^0 - 1 \cdot 2^{-1} = 0.$
  $(105.5)_{10} = (1101001,1)_2.$

  Převod z desítkové do šestnáctkové soustavy:
  $(6378)_{10} = 1 \cdot 16^3 - 8 \cdot 16^2 - 14 \cdot 16^1 - 10 \cdot 16^0 = 18ea.$
  $(6378)_{10} = (18ea)_{16}.$

11. Převody mezi dvojkovou a šestnáctkovou soustavou

Využívají fakt, že $16 = 2^4$. Vztah (1) lze vyjádřit prostřednictvím uspořádaných čtveřic binárních čísel

$$a = b_n 2^{4n} + b_{n-1} 2^{4(n-1)} + b_{n-2} 2^{4(n-2)} + ... + b_0 2^0. \qquad (7)$$

Převod z šestnáctkové soustavy do dvojkové:
Číslice čísla a převádíme samostatně do dvojkové soustavy tak, aby binární čísla tvořily čtveřice. Případné neúplné čtveřice (s výjimkou první cifry) doplníme zleva nulami.

Převod z dvojkové soustavy do šestnáctkové
Rozdělení binárního čísla na čtveřice a jejich postupný převod do šestnáctkové soustavy.

- **Příklad:**

  Převod z šestnáctkové do dvojkové soustavy. Pak $(12af)_{16} = (1001010101111)_2$.

1	2	a	f
1	10	1010	1111
1	0010	1010	1111

12. Převod čísel $|a| < 1$

Vztah

$$a = \pm \sum_{i=-1}^{-m} b_i z^i = \pm(b_{-1}z^{-1} + b_{-2}z^{-2} + b_{-3}z^{-3} + \ldots + b_{-m}z^{-m}). \quad (8)$$

můžeme po vynásobením z pro přepsat do tvaru

$$z \cdot a = \pm z \sum_{i=-1}^{-m} b_i z^i = \pm(b_{-1}z^{0} + b_{-2}z^{-1} + b_{-3}z^{-2} + \ldots + b_{-m}z^{-m+1}). \quad (9)$$

Princip převodu:

Desetinnou část čísla opakovaně násobíme základem z. V každém krokem spočteme další koeficient b_i.
Postupujeme tak dlouho, dokud nedosáhneme požadované přesnosti nebo $z \cdot b^j \neq 1$.

13. Převod čísel $|a| < 1$ a přesnost

Popsaný princip převodu má negativní důsledek: Číslo, které je ve zdrojové soustavě vyjádřené jako přesné, může být v cílové soustavě znázorněno jako číslo neúplné.

Důsledek:
Tato vlastnost se projeví při aritmetických operacích s desetinnými čísly. Výsledek jakékoliv operace je pak zatížen chybou.

- **Příklad:**
  Převod z desítkové do binární soustavy. $(0.625)_{10}$.

2^i	b_i	$0.625 \cdot 2$
2^{-1}	1	$0.25 \cdot 2$
2^{-2}	0	$0.5 \cdot 2$
2^{-3}	1	1

14. Zápis čísel pomocí řádové mřížky.

Použita pro reprezentaci čísel v počítači. Řádová mřížka je tvořena dvěma částmi:

- celočíselnou
- desetinnou

Obě části jsou mřížky jsou odděleny řádovou čárkou.

Levá krajní pozice řádové mřížky je vyhrazena znaménkovému bitu, který indikuje, zda je číslo kladné či záporné. Délku řádové mřížky označíme l, nejvyšší možný celočíselný řád jako n, nejnižší možný desetinný řád jako m.

Celková délka mřížky doplněná o znaménkový bit

$$l = m + n + 1. \tag{10}$$

15. Přeplnění (přetečení)

Pokud je délka mřížky menší než počet cifer čísla a, číslo nebude zobrazitelné $\Rightarrow$ v takovém případě není možné do mřížky uložit všechny číslice čísla a.

Dojde k *přeplnění (přetečení)* a následné ztrátě přesnosti v reprezentaci čísla.

- *Důsledek přeplnění*
  Číslo bude uloženo s chybou, dojde k jeho zaokrouhlení či záměně za číslo s opačným znaménkem.
- *Výskyt přeplnění*
  K tomuto jevu dochází při aritmetických operací, kdy počet cifer výsledku může být výrazně vyšší než počet číslic jednotlivých činitelů nebo v případě znázornění neúplných čísel.

Opakem přetečení je *podtečení*. Číslo a se blíží nule, je zobrazeno jako 0. Praktické ukázky přetečení a podtečení při aritmetických operacích budou ukázány dále.

16. Přímý kód

Tři varianty znázorňování čísel se znaménky:

1. přímý kód
2. inverzní kód
3. doplňkový kód

Přímý kód a jeho charakteristika.

Doplnění znaménkového bitu na první pozici. Pokud je číslo kladné, nabývá znaménkový bit hodnoty 0, je -li číslo záporné, nabývá hodnoty 1. Maximální zobrazitelné číslo v intervalu $\langle -2^r - 1, 2^r - 1 \rangle$.

Nevýhodou je dvojí reprezentace nuly, a to jako: +0 nebo -0. Kód nezachovává relace mezi kladným a záporným číslem. Záporné číslo není menší než kladné. Kladné číslo stejné jako záporné až na znaménkový bit.

- **Příklad**
  Znázornění záporných čísel s použitím přímého kódu.

$$
\begin{aligned}
P^8(105)_{10} &= (01101001)_2 \\
P^8(-105)_{10} &= (11101001)_2 \\
P^8(0)_{10} &= (00000000)_2 \\
P^8(\quad 0)_{10} &= (10000000)_2
\end{aligned}
$$

17. Inverzní kód

Levý bit rezervován pro znaménko. Čísla kladná zobrazujeme stejným způsobem jako v přímém kódu, čísla záporná *negací* každého z bitů. Záporná čísla převádíme na čísla doplňková, vzniká tzv. *jedničkový doplněk*. Maximální zobrazitelné číslo leží v intervalu $\langle -2^r - 1, 2^r - 1 \rangle$.

Nevýhodou je možnost dvojí reprezentace nuly, a to jako: +0 nebo -0.
Kód na rozdíl od předchozího zachovává relace mezi čísly kladnými a zápornými čísly (záporné číslo je menší než kladné).

- **Příklad:**
  Znázornění záporných čísel s použitím inverzního kódu.

$$
\begin{aligned}
I^8(105)_{10} &= (01101001)_2 \\
I^8(-105)_{10} &= (10010110)_2 \\
I^8(0)_{10} &= (00000000)_2 \\
I^8(-0)_{10} &= (11111111)_2
\end{aligned}
$$

18. Doplňkový kód:

Nejpoužívanější varianta kódu, odstraňuje problematiku dvojí reprezentace nuly. Levý bit rezervován pro znaménko.

Kladné číslo se zobrazí stejně jako v přímém kódu, pro záporné použijeme tzv. *dvojkový doplněk*. Dvojkový doplněk vznikne připočtením čísla 1 k jedničkovému doplňku. Maximální zobrazitelné číslo leží v intervalu $\langle -2^r - 1, 2^r - 1 \rangle$.

- **Příklad:**
  Znázornění záporných čísel s použitím doplňkového kódu.

$$
\begin{aligned}
ID^8(105)_{10} &= (01101001)_2 \\
D^8(-105)_{10} &= (10010111)_2 \\
D^8(0)_{10} &= (00000000)_2 \\
D^8(-0)_{10} &= (00000000)_2
\end{aligned}
$$

19. Sčítání čísel

Základní aritmetická operace, prostřednictvím této operace lze vyjádřit všechny ostatní operace.

Princip sčítání:
Prováděno stejným postupem jako v desítkové soustavě. Jednotlivé sčítance zarovnáme pod sebe tak, abychom sčítali číslice stejných řádů. Sčítání probíhá od čísel nižšího řádu k číslům vyššího řádu, tj. ve směru z leva do prava.

Přenos do vyššího řádu p:
Pokud je součet sčítanců stejných řádů pro číselnou soustavu se základem z větší než $\max(b_i)$, na pozici součtu v tomto řádu zapíšeme hodnotu číslice nižšího řádu z tohoto součtu, ke sčítancům následujícího, tj.vyššího řádu, přidáme hodnotu 1, kterou přeneseme do vyššího řádu.

20. Příklad sčítání

Sečtěte následující sčítance v desítkové, dvojkové a šestnáctkové soustavě:

- $(2387)_{10} + (95)_{10}$
- $(1011)_2 + (11)_2$
- $(51ad4)_{16} + (ff)_{16}$

Výsledky uvedeny v tabulkách.

p	1	1		
$(2$	3	8	$7)_{10}$	
		$(9$	$5)_{10}$	
$(2$	4	8	$2)_{10}$	

p	1	1		
$(1$	0	1	$1)_2$	
		$(1$	$1)_2$	
$(1$	1	1	$0)_2$	

p		1	1		
$(5$	1	a	d	$4)_{16}$	
			$(f$	$f)_{16}$	
$(5$	1	b	d	$3)_{16}$	

21. Přeplnění (přetečení) při sčítání

Při sčítání může docházet k přeplnění (přetečení) ovlivňující veškeré následné operace s výsledkem.

Součet nemusí být stejného řádu jako oba sčítance. Pokud dojde k situaci, kdy výsledek není možné zobrazit do řádové mřížky, dojde k *přeplnění*.

K přeplnění dochází v těchto případech:

- sčítáme čísla se stejnými znaménky a dojde k přenosu do znaménkové bitu. Tímto krokem změníme znaménko součtu,
- sčítáme-li čísla a dojde k různému přenosu "z" a "do" znaménkového bitu.

22. Příklad přeplnění (1/2)

Pro $l = 8$ a z=2 určete $c = a + b$, kde $a = (-7)_{10}$, $b = (-3)_{10}$.

V tomto případě je přenos do znaménkového bitu (1) i ze znaménkového bitu (1) roven jedné, výsledek je *správný*.

$(c)_D = (11110110)_2 = -(c) = (-10)_{10}$.

p		1	1	1	1	1			1	
a	00000111	$(a)_D$	1	1	1	1	1	0	0	1
b	00000011	$(b)_D$	1	1	1	1	1	1	0	1
		$(c)_D$	1	1	1	1	0	1	1	0
		c	0	0	0	0	1	0	1	0

23. Příklad přeplnění (2/2)

Pro $l = 8$ a $z = 2$ určete $c = a + b$, kde $a = (81)_{10}$, $b = (66)_{10}$.

V tomto případě je přenos do znaménkového bitu roven 1a přenos ze znaménkového bitu žádný 0, výsledek je *nesprávný*.

$(c)_D = (11110110)_2 \neq a + b$,

0	1							
a	0	1	0	1	0	0	0	1
b	0	1	0	0	0	0	1	0
c	1	1	1	1	0	1	1	0

24. Celá čísla a jejich reprezentace v počítači

Tyto datové typy, které nazýváme *celočíselnými datovými typy*.

Liší se počtem cifer / a tím pádem i rozsahem hodnot, které do nich lze uložit.

- *Dělení podle přesnosti:*
  Typy s nižší přesností bývají označovány jako `short`, typy s vyšší přesností jako `long`.
- *Dělení podle znaménka:*
  Celočíselné datové typy se rozdělují na typy se znaménka a bez znaménka. Typy bez znaménka bývají označovány jako `unsigned`, typy se znaménky jako `signed`.

V každém programovacím jazyku nemusí být zastoupeny všechny kombinace celočíselných datových typů.

25. Přehled celočíselných datových typů:

Přehled celočíselných datových typů v programovacích jazycích C++, Java a Python:

Velikost	Rozsah	Znaménko	C++	Java	Python
1B	$-128, 127$	ano	signed char	byte	-
1B	$0, 255$	ne	unsigned char	-	-
2B	$-32768, 32767$	ano	int	short	integer
2B	$0, 65536$	ne	unsigned	-	-
4B	$-2^{31}, 2^{31} - 1$	ano	long	int	long integer
4B	$0, 2^{32} - 1$	ne	unsigned long	-	-
8B	$-2^{63}, 2^{63} - 1$	ano	-	long	-

26. Reálná čísla a jejich reprezentace v počítači:

Reálná čísla se používají pro reprezentaci desetinných čísel. Většina výpočtů prováděných v programech probíhá právě s reálnými čísly.

Reálná čísla lze v počítači reprezentovat dvěma způsoby:

- reprezentace s pevnou řádovou čárkou,
- reprezentace s pohyblivou řádovou čárkou.

Zobrazení s pevnou řádovou čárkou

Řádová čárka oddělující celou desetinnou část čísla od celé části je umístěna na pevné pozici. První bit znaménkový, pro celočíselnou část je vyhrazeno n bitů, pro desetinnou část m bitů (viz řádová mřížka).

Rozsah zobrazitelných čísel je dán intervalem $\langle -2^{n-1}, 2^{n-1} - 2^{-m} \rangle$. Přesnost znázornění čísel v pevné řádové čárce závisí na počtu bitů použitých pro jejich reprezentaci.

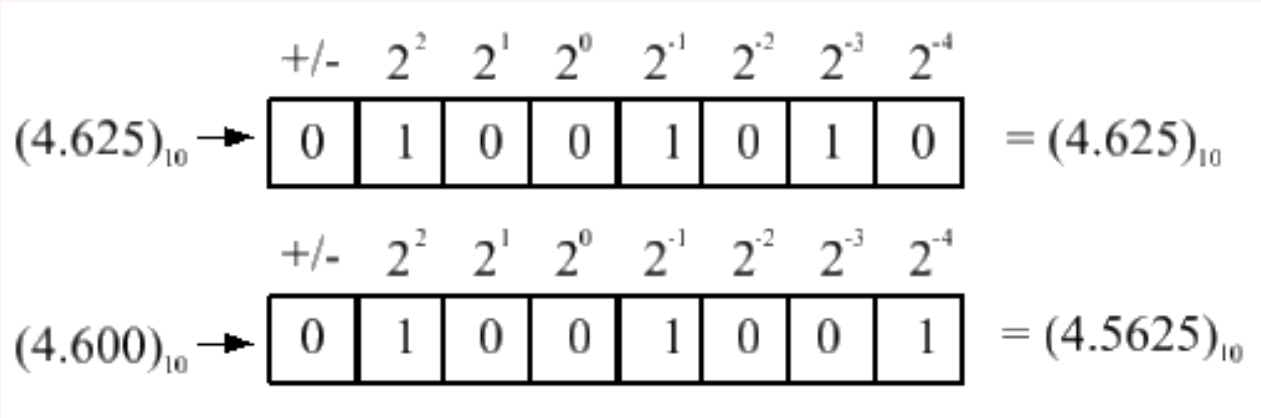
27. Nevýhoda zobrazení v pevné řádové čárce

Všechna čísla jsou zobrazována se stejnou *absolutní* přesností.
Důsledkem je fakt, že jsou "malá" čísla zobrazována s *nízkou* relativní přesností.

Při vědeckých výpočtech však požadujeme, aby byla všechna čísla uložena se *stejnou* relativní přesností.

- Příklad:
  Znázornění čísel čísel 4.625 a 4.6 v pevné řádové čárce v 8 bitové reprezentaci za použití kódu $3-4$.

  První číslo lze reprezentovat s tímto počtem bitů přesně, druhé již nikoliv, počet bitů pro znázornění desetinné části není dostatečný, dochází k *přeplnění*.



28. Zobrazení čísel s pohyblivou řádovou čárkou

Semilogaritmický tvar čísla:

Při zobrazení čísel s pohyblivou (resp. s plovoucí řádovou čárkou) používáme semilogaritmický tvar čísla. Lze ho vyjádřit ve tvaru:

$$a = q \cdot z^e. \tag{11}$$

Hodnota q představuje *mantisu* čísla x, hodnota z je základ číselné soustavy, e představuje *exponent*. Mantisa je zobrazována vždy přímým kódem. Mantisa splňuje normalizační podmínku ve tvaru

$$\frac{1}{z} \leq q \leq 1. \tag{12}$$

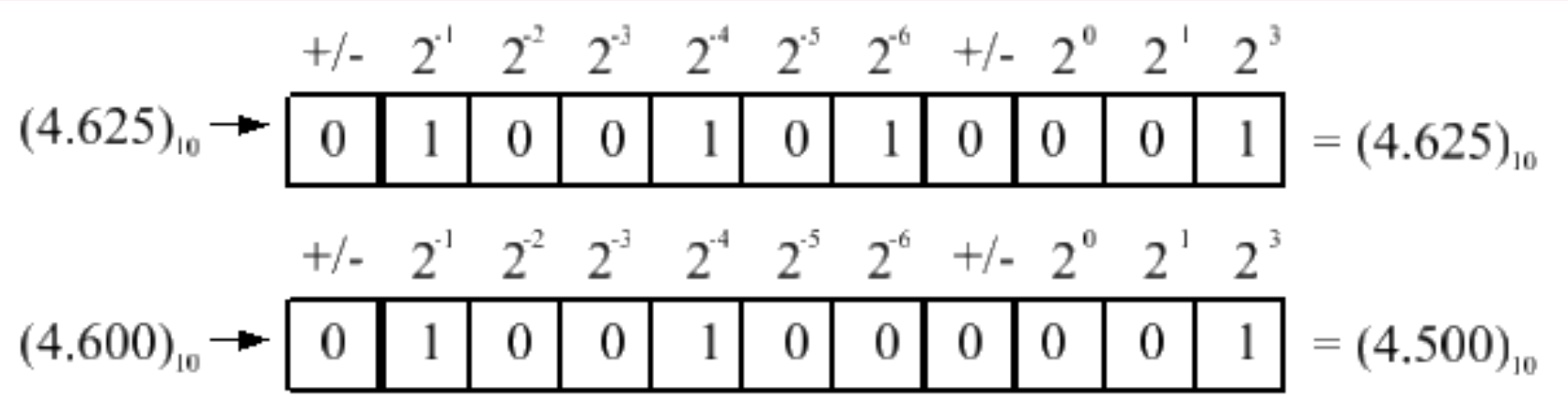
Levá část nerovnosti zajišťuje, aby nedošlo ke ztrátě přesnosti, pravá část, aby nedošlo k přeplnění. Číslo, které splňující normalizační podmínku nazýváme *normalizované*.

Mantisu zobrazujeme v celočíselné části řádové mřížky, exponent v desetinné části. Před každou z částí je umístěn znaménkový bit. Výhodou tohoto postupu je *efektivnější* ukládání dat a stejná relativní přesnost všech činitelů, které vstupují do výpočtu.

29. Příklad znázornění čísla v plovoucí řádové čárce

Znázornění čísla 4.625 a 4.6 v plovoucí řádové čárce.

Použijeme normalizační podmínku ve tvaru: $4.625 = +0,578125 \cdot 2^{+3}$, $4.600 = +0,575 \cdot 2^{+3}$.



30. Přetečení, podtečení (další podrobnosti)

Dochází k nim při práci s "příliš velkými" či "příliš malými" čísly.

- *Podtečení:*
  Pokud je číslo $a < a_{min}$, dochází k podtečení.
- *Přetečení:*
  Pokud je $a > a_{max}$, dochází k přetečení.

Důsledkem obou operací je ztráta přesnosti, dojde k zaokrouhlení čísla.

Důsledek přetečení:
Dojde -li k přetečení, pak se při reprezentaci čísla v plovoucí řádové čárce výsledek nevejde do počtu bitů určených pro mantisu. Bity v mantise se posunou o jeden bit vpravo, hodnota exponentu se zvýší o jedničku.

Důsledek podtečení:
Pokud dojde k podtečení, je výsledkem operace hodnota $+0$ nebo -0.

31. Problémy při práci s reálnými čísly (1/2):

Rovnost dvou čísel

Porovnáváme -li dvě reálné hodnoty a, b, pak podmínka

$$a = b \tag{13}$$

v obecném případě *nebude* pravdivá. V průběhu výpočtů dochází k zaokrouhlování a následné ztrátě přesnosti.

Nebudeme testovat rovnost dvou čísel a, b, ale absolutní hodnotu jejich rozdílu $|a - b|$ s hodnotou $\varepsilon \gg 0$:

$$|a - b| < \varepsilon. \tag{14}$$

Komutativní zákon.

Vzhledem k tomu, že při práci s reálnými čísly dochází k jejich zaokrouhlení, nemusí platit komutativní zákon pro sčítání a násobení.

$$\begin{aligned} a + b &\neq b + a \\ a - b &\neq a + (-b) \\ -(a + b) &\neq -a - b \\ ab &\neq ba \\ (-a)b &\neq -(ab) \\ 1a &\neq a \end{aligned}$$

32. Problémy při práci s reálnými čísly (2/2):

Asociativní zákon.

Z výše uvedených důvodů nemusí platit ani asociativní zákony pro sčítání a násobení:

$$\begin{aligned} a+(b+c) &\neq (a+b)+c \\ a(bc) &\neq (ab)c \end{aligned}$$

Distributivní zákon.

Z výše uvedených důvodů nemusí platit ani distributivní zákon:

$$(a+b)c \neq ac+bc.$$

Rychlost aritmetických operací.

Rychlost aritmetických operací s celými čísly je řádově vyšší než rychlost provádění aritmetických operací s reálnými čísly.
Tuto vlastnost je nutné vzít v potaz při návrhu algoritmu a důkladně zvážit, které proměnné budou deklarovány jako celočíselné, a které jako reálné.

33. Reálná čísla a jejich reprezentace v počítači

Přehled reálných datových typů v programovacích jazycích C++, Java a Python.

Velikost	Rozsah	Znaménko	C++	Java	Python
4B	$1.4 \cdot 10^{-45}, 3.4 \cdot 10^{38}$	ano	float	float	float
8B	$4.9 \cdot 10^{-234}, 1.7 \cdot 10^{308}$	ano	double	double	-
10B	$1,190 \cdot 10^{4932}$	ano	long double	-	-
		ano	-	-	complex

34. Hodnoty INF a NaN

Při práci s čísly se setkáme se dvěma speciálními hodnotami INF a NaN.

Hodnota INF:
Hodnota INF představuje nekonečno, získáme ho jako výsledek některých aritmetických operací, např. dělení nulou.

Hodnota NaN:
Představuje akronym "not a number". Vzniká jako výsledek operací, který není definován, např. odmocnina ze záporného čísla.