Série sešitů MTP

Matematické technologie pro modelování

Stanislav Míka, Ludvík Vlček

PLZEŇ 2008 – 2015

Obsah série sešitů:

Věříme, že existuje neprázdná množina lidí, kteří ocení informace, které se jim zde snažíme poskytnout. Víme, že do této množiny nepatří ti, jejich vzdělání je menší, než elementární. Na druhé straně do ní nepatří specialisté, pro něž je naše vyprávění nedostatečně podrobné a hluboké.

Takže,

není už na čase změnit vnímání matematiky ve vzdělávání?

Inspirační zdroje:

1. Sebraná moudra - Kniha I. Bible
   Kniha II. Korán
   Kniha III. Talmud
   Kniha Manifest komunistické strany
2. Zkušenosti dlouholeté práce ve vzdělávací a odborné oblasti
   - přednášky a semináře pro studenty matematických, fyzikálních, technických, ekonomických, geomatických, kartografických oborů na ZČU,
   - přednášky z Aplikované matematiky pro inženýry a výpočtáře podniku Škoda Plzeň,
   - přednášky pro středoškolské studenty,
   - učebnice ,skripta, texty ze základních matematických disciplín,
   - práce v oblasti IT
3. *Petr Vopěnka*, především jeho edice *Prameny evropské vzdělanosti*
4. *Chalmers University project*: Body & Soul: Applied Mathematics Education Reform Project
5. Edice MŠMT 1978 až 1989 a novější multimediální prameny
6. *Karel Rektoris* a kolektiv: Přehled užité matematiky
7. *Rudolf Výborný*: Diferenciální počet, Academia 1966
8. *Vojtěch Jarník*: Úvod do počtu integrálního, Úvod do počtu diferenciálního

ZÁKLADNÍ MATEMATICKÉ STRUKTURY

I. Sešit série MTM

Stanislav Míka, Ludvík Vlček

Plzeň 2015

Anotace

Tento sešit je nezbytným vstupem ke všem dalším sešitům. Kromě přehledu středoškolských elementarit jsme sem zařadili odstavce, které se většinou uvádějí ve specializovanějších výkladech, ale svou povahou odpovídají názvu sešitu.

Zvolili jsme výklad ve stylu vyprávění, něco jako „Hovory o matematice a jejích aplikacích“. Matematickou přesnost se snažíme respektovat tam, kde to je nezbytné, ale podřizujeme se spíše zásadě srozumitelnosti a čtivosti.

Snažíme se také ukázat, kam směřují uvedené poznatky v aplikačních technologiích. Nedodržujeme hierarchickou strukturu, protože nechceme být standardní učebnicí úvodu do matematiky, ani sbírkou „vzorečků“ či příkladů.

Adresátem našeho výkladu je ten, kdo si potřebuje nebo chce doplnit či připomenout poznatky ze školy a případně si odpovědět na otázky, proč je matematika tak užitečná a zajímavá.

Již název tohoto sešitu může pro neprofesionála vypadat „strašidelně“, ale pouze zde připomínáme to, co skoro každý maturant slyšel ve škole a pokud neslyšel, pak

většinou neposlouchal. Připomínáme zde základní vlastnosti čísel, skupin čísel, množin a výroků a to nejen z pohledu matematiky, ale i informatiky.

Na rozdíl od učebnic středoškolského typu zde výklad koncentrujeme v čase i prostoru. Na množinu se díváme jako na systém, který obsahuje jak objekty, tak vztahy mezi nimi a pravidla, jak se s nimi manipuluje, tj. pravidla operací. Nejdeme do teoretické hloubky , ale snažíme se postihnout nezbytné informace potřebné v aplikacích. Termínem *abstraktní struktura* rozumíme to, co se také nazývá *systém*.

Poslední kapitolu věnujeme námětům pro další vzdělávání s historickými souvislostmi. Zaznamenáváme zde také svoje názory, stanoviska k jiným názorům a didaktické postřehy.

Příklady mají především ilustrační charakter a nepředpokládáme, že adresát si bude „za dlouhých zimních večerů“ trénovat často nepotřebné postupy.

Tento úvodní sešit je věnován především jazyku čísel, číselným oborům , množin s troškou formální logiky. Ve věku všeobecného užívání počítačů považujeme za rozumné zde uvést strukturu počítačových čísel a principy zaokrouhlování. Stejně tak některé vstupní informace do informatiky považujeme zde za užitečné.

Obsah I. sešitu

Kapitola 1
Čísla, body; počty, manipulace

Obsah kapitoly

1.1 Čísla reálná, iracionální, racionální, celá, přirozená

Dovolujeme si předpokládat, že se obracíme na čtenáře, jehož znalosti matematiky jsou na úrovni absolventa Karlovy univerzity ve středověku, kde se v aritmetice přednášelo sčítání, odčítání, násobení, dělení, a pro nadané studenty umocňování a odmocňování. To vše v desítkové početní soustavě, tj. podle Al-Chvárízmího. V geometrii pak se přednášelo podle Eukleida z Alexandrie. Podle Ptolemaiova Almagestu (původním názvem $Μαθηματικη Συνταξις$ Mathématiké syntaxis – Matematická soustava) se přednášela astronomie a geografie.

Připomeňme si několik základních, elementárních pojmů a vlastností s nimi související.

1.1.1 Množina všech přirozených čísel

Množina všech ***přirozených čísel***

$$\mathbb{N} = \{1, 2, 3, \ldots, n, n+1, \ldots\}$$

Základní vlastnost (princip indukce):

$$\text{Je-li } \; k \in \mathbb{N} \text{ potom } \; k+1 \in \mathbb{N}\,;$$

Stručně:

$$k \in \mathbb{N} \;\Rightarrow\; k+1 \in \mathbb{N}\,.$$

Slovně:
Když k je přirozené číslo, potom jeho první následník je také přirozené číslo.

Teorie přirozených čísel je dána tzv. *Peanovými axiomy*. Formulují se v predikátové logice. V „lidštině“ zní první tři takto:

1. Existuje přirozené číslo, které není následníkem žádného přirozeného čísla.
2. Ke každému přirozenému číslu n existuje přirozené číslo n', které je jeho následovníkem.
3. Číslo 1 není následovníkem žádného přirozeného čísla.

Množina $\mathbb{N}$ je uzavřená vzhledem k operacím *sčítání* a *násobení* (tj. výsledek operace dvou libovolných čísel $\in \mathbb{N}$, je opět v $\mathbb{N}$).

V množině $\mathbb{N}$ je řešitelná rovnice:

$$m + x = n, \text{ pouze pro některé } m, n \in \mathbb{N}.$$

Rovnice

$$5 + x = 1 \text{ není řešitelná v } \mathbb{N},$$

kdežto rovnice

$$5 + x = -1 \text{ nemá v } \mathbb{N} \text{ smysl.}$$

Rozšířená množina přirozených čísel:

$$\mathbb{N}_0 = 0, 1, 2, 3, \ldots, n, n+1, \ldots$$

se potřebuje například v teorii signálů. Množina N_0 je uzavřená vzhledem k operaci dělení se zbytkem. V množině N_0 je řešitelná rovnice:

$$m \cdot x = 0.$$

Jediným řešením je číslo $x = 0$, $\forall m \in \mathbb{N}_0$.

Blok 1.1.1: Přirozená čísla

Poznámka:

Množina $\mathbb{N}$ má nekonečně mnoho prvků. Každé množině, která má konečný počet prvků („dají se spočítat“) přiřazujeme číslo, které nazýváme počet prvků. U nekonečných množin užíváme termín *mohutnost množiny*, nebo *kardinalita*. Říkáme, že množina $\mathbb{N}$ je spočetná a má mohutnost $\aleph_0$ (alef nula).

Od nepaměti se lidé setkávali s tím, že soubory měly konečný počet objektů. Pravděpodobně jedna z prvních matematických otázek byla, jaké číslo je největší. Existovaly (v Indii, v Číně) symboly pro velké množství objektů. Cesta k pojmu ***nekonečno*** vedla přes mnoho generací a byla velmi dlouhá, ale jak víme, byla konečná.

Princip matematické indukce (opět upravená verze jednoho Peanova axiomu)

Mějme nějaký výrok $T(k)$ pro přirozené číslo k. Dále předpokládejme:

1. Pro nějaké přirozené číslo s platí výrok $T(s)$.
2. Pro přirozené číslo $k \geq s$ platí *implikace* $T(k) \Rightarrow T(k+1)$.

Potom výrok $T(k)$ platí pro všechna přirozená čísla $k \geq s$.

Základem důkazu matematickou indukcí (úplná indukce) je klíčový předpoklad (2).

Ilustrace:

$$\text{pro přirozené číslo } k \text{ je hodnota } k^2 - k + 41 \text{ také přirozené číslo.}$$

Ale všimneme si, že navíc pro $k = 1, 2, 3, \ldots, 40$ je uvedená hodnota *prvočíslo* (neúplná indukce).
Pro:

$$k = 41 \text{ dostaneme } 41 \cdot 41 = 1681, \text{ což není prvočíslo!}$$

Předpoklad (2) nebyl splněn, nelze dokázat pravdivost požadované implikace.

Blok 1.1.2: Princip matematické indukce

1.1.2 Množina všech celých čísel

Množina všech ***celých čísel***

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

má tyto základní vlastnosti:

1. Množina $\mathbb{Z}$ je uzavřená vzhledem k operaci sčítání, násobení, odčítání (výsledek operace je opět v $\mathbb{Z}$)
2. Rovnice $m + x = n$ je v $\mathbb{Z}$ vždy řešitelná. Ke každému celému číslu existuje v $\mathbb{Z}$ opačný prvek. Rovnice $m \cdot x = n$ není v $\mathbb{Z}$ vždy řešitelná ($\mathbb{Z}$ není uzavřená vzhledem k operaci dělení).

3. Množina $\mathbb{Z}$ je *spočetná*, tj. má stejnou *mohutnost* jako množina $\mathbb{N}$.

4. Každé celé číslo se dá vyjádřit jako rozdíl dvou přirozených čísel.

5. $\mathbb{N} \subset Z$, tj. $\mathbb{N}$ je podmnožina $\mathbb{Z}$. Znamená to, že když:

$$p \in \mathbb{N}, \text{ potom } p \in \mathbb{Z}.$$

   Pozor: Implikace $p \in \mathbb{Z} \Rightarrow p \in \mathbb{N}$ je nepravdivý výrok!

6. Na množině celých čísel můžeme, podobně jako u přirozených čísel, nadefinovat dělení se zbytkem, jen se musíme vypořádat se zápornými čísly. Takže základní definice bude tentokrát vypadat takto:

$$a = q \cdot b + r,\ q \in \mathbb{Z},\ b \in \mathbb{Z} - \{0\}, \quad 0 \leq r \leq |b|.$$

   V tomto výrazu dělíme $a : b$, číslo q představuje výsledek (kvocient) a číslo r zbytek po dělení. Číslo b musí být různé od nuly, nemůžeme dělit nulou. Zbytek musí být kladný a musí být menší než absolutní hodnota z b, což nám eliminuje případ, kdy bychom dělili záporným číslem. Například:

$$7 : 2,\ a = 7,\ b = 2; \quad 7 = 3 \cdot 2 + 1,\ q = 3,\ r = 1.$$

Blok 1.1.3: Celá čísla

Poznámka: množinové a logické symboly budou připomenuty v kapitole 5.

Velká část nejslavnějších matematických problémů historie i současnosti pochází z oblasti teorie čísel. To je dáno skutečností, že tvrzení o celých číslech jsou v převážné většině případů snadno vyslovitelná a lehce pochopitelná i pro člověka bez matematického vzdělání, avšak jejich dokázání může vyžadovat použití mnoha specializovaných metod z nejrůznějších jiných odvětví matematiky.

1.1.3 Množina všech racionálních čísel

Množina všech ***racionálních čísel*** $\mathbb{Q}$

Přidáme-li k celým číslům zlomky (viz matematika starého Egypta) dostaneme množinu $\mathbb{Q}$.

Základní vlastnosti racionálních čísel:

1. Každé racionální číslo lze vyjádřit ve tvaru *zlomku* s celočíselným čitatelem a celočíselným *nenulovým* jmenovatelem.

2. Každé racionální číslo lze vyjádřit buď konečným, nebo nekonečným, ale periodickým desetinným rozvojem. *Nekonečným desetinným rozvojem* nazýváme výraz:

$$r = c, n_1, n_2, n_3, n_4, \ldots$$

kde c je libovolné celé číslo a každé $n_i \ : \ (i \in N)$ je jedna z deseti číslic $0, 1, 2, 3, 4, \ldots, 8, 9$.

Když například $r = c, n_1, n_2, n_3, n_4, 0, 0, 0, \ldots$, nebo $r = c, n_1, n_2, n_3, \ldots, n_k$, hovoříme o *konečném desetinném rozvoji*.

3. Množina $\mathbb{Q}$ je spočetná.

4. Množina $\mathbb{Q}$ je uzavřená vzhledem k operacím sčítání, odčítání, násobení, dělení, tj.
$$\forall q \in \mathbb{Q}, \exists m \in \mathbb{Z}, \exists n \in \mathbb{Z}, n \neq 0 \ : \ n \cdot q = m.$$
Říkáme, že každá rovnice $n \cdot q = m$ s neznámou q je řešitelná v $\mathbb{Q}$.

Avšak, např. každá kvadratická rovnice již v $\mathbb{Q}$ řešitelná není ($\mathbb{Q}$ není uzavřená vzhledem k odmocňování). Dále například exponenciální rovnice $3^x = 5$ není v $\mathbb{Q}$ řešitelná, ale $3^x = 9$ ano.

5. Množina $\mathbb{Q}$ je *hustá*, tj. mezi každými dvěma racionálními čísly existuje aspoň jedno další racionální číslo; např. jejich aritmetický průměr.

6. Množina $\mathbb{Q}$ není úplná: zobrazíme-li $\mathbb{Q}$ na body přímky, zjistíme „pomocí silné lupy“, že velké množství bodů přímky nejsou obsazeny žádným racionálním číslem. Těmto „dírám“ v $\mathbb{Q}$ říkáme *iracionální čísla*.

Blok 1.1.4: Racionální čísla

Poznámka: Číslice $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ se obvykle nazývají *arabské*, ale je to „historická křivda“. Tyto znaky jsou původně *indické* a vznikly ze znaků jazyka sanskrt.

Připomeňme si:

1. $x \in \mathbb{Q}$: $x = 2,3751; \quad x = \dfrac{3}{25}, \quad x = \dfrac{22}{7} \neq \pi$ **(!)**

2. $y \in \mathbb{Q}$: $y = 1,27353535... = 1,27\overline{35}$ (nekonečný periodický desetinný rozvoj)

3. *Otázka:* Jak se liší čísla $2,5$ a $2,499999...$.
Odpověď: Jsou to dva různé zápisy téhož racionálního čísla.

4. *Otázka:* Je zápis $\pi = 3,14$ správný?
Odpověď: Není: $\pi \notin \mathbb{Q}, \quad 3,14 \in \mathbb{Q}$.

5. Desetinný rozvoj: $827,6305 = 8.10^2 + 2.10^1 + 7.10^0 + 6.10^{-1} + 3.10^{-2} + 0.10^{-3} + 5.10^{-4}$

Úkol:

1. Myslete si 3 jednomístná přirozená čísla.

2. První číslo vynásobte 2

3. Přičtěte k výsledku 5
4. Výsledek vynásobte 5
5. K výsledku přičtěte druhé číslo
6. Výsledek vynásobte 10
7. K výsledku přičtěte třetí číslo
8. Sdělte mi výsledek a já vám řeknu čísla, která čísla jste si mysleli.

Odpověď:

Myslím si:

$$a, b, c;$$

Postupně vykonám pokyny 1 až 7:

$$[(2a+5)5+b]10+c=V \;\rightarrow\; 100a+10b+c+250=V$$

od sděleného výsledku se odečte 250:

$$V-250=100a+10b+c\,;\ (\text{ desetinný rozvoj}).$$

Volba 1. $a=3,\ b=7,\ c=9:\ 629-250=379.$
Volba 2. $a=5,\ b=4,\ c=3:\ 793-250=543.$

1.1.4 Množina všech reálných čísel

Množina všech ***reálných čísel*** $\mathbb{R}$

Základní vlastnosti reálných čísel:

1. Každé reálné číslo lze vyjádřit desetinným rozvojem, buď konečným, nebo nekonečným.
2. Každé reálné číslo je limitou nějaké konvergentní posloupnosti racionálních čísel.
3. Každému bodu dané přímky lze přiřadit jediné reálné číslo. Přímce s tímto přiřazením říkáme *reálná osa*, resp. *osa reálných čísel*.
4. Množina racionálních čísel $\mathbb{Q}$ je podmnožinou množiny reálných čísel $\mathbb{R}$, tj.

$$\mathbb{Q} \subset \mathbb{R}.$$

5. Množina $\mathbb{R}$ je nespočetná.
6. Množina $\mathbb{R}$ je *zúplněním* množiny $\mathbb{Q}$.

Blok 1.1.5: Reálná čísla

Připomeňme si:

1. číslo $\frac{1}{4} = 0,25$ je racionální číslo (konečný desetinný rozvoj)
   avšak: $0,25 = 0,25000\ldots$ je to i reálné číslo (nekonečný desetinný rozvoj).
2. $\pi \in \mathbb{R} : \pi = 3,1415926579\ldots$ (nekonečný neperiodický desetinný rozvoj)

1.1.5 Množina všech iracionálních čísel

Množina všech ***iracionálních čísel***

Základní vlastnosti:

1. Iracionální číslo je takové reálné číslo, které není racionálním číslem. Platí následující ekvivalence:
$$x \in \mathbb{R} - \mathbb{Q} \Leftrightarrow x \in \mathbb{R} \wedge x \notin \mathbb{Q}$$
2. *Iracionální číslo* má nekonečný neperiodický desetinný rozvoj.
3. Každé iracionální číslo se dá reprezentovat posloupností racionálních čísel; dokonce nekonečně mnoha posloupnostmi racionálních čísel. Například číslo e (základ přirozených logaritmů) je iracionální a dá se reprezentovat posloupností racionálních čísel:
$$\left(\frac{n+1}{n}\right)^n.$$
4. Množina všech iracionálních čísel je nespočetná.

Blok 1.1.6: Iracionální čísla

1. *Otázka*: je číslo $\pi = 3,1415926579\ldots$ iracionální?
   Odpověď: Ano, protože ho lze vyjádřit *pouze* nekonečným neperiodickým desetinným rozvojem.
2. *Otázka*: je číslo $\frac{2}{7} = 0.285714285714285714\ldots$ iracionální?
   Odpověď: Ne, protože má sice nekonečný desetinný rozvoj, ale cifry (285714) se stále opakují.
   Toto ovšem není věrohodný argument, protože je nemožné rozhodnout z několika cifer, že se opakují celé skupiny cifer. Co je však zřejmé, že dané číslo je podíl dvou celých čísel, a tudíž je to číslo racionální.
3. Číslo $e = 2,71828182845904523536028747135 2\ldots$ je iracionální. Otázkám i odpovědím kolem tohoto čísla bude věnována pozornost v celé řadě dalších odstavců.

Příklad 1.1.1: Otázky

1.2 Technologie komplexních čísel

Komplexní čísla s operacemi sčítání a násobení tvoří komutativní těleso. Je to největší komutativní algebraické nadtěleso reálných čísel a algebraický uzávěr tělesa reálných čísel. Toto těleso nelze okruhově uspořádat.

Nerozumíte o čem je řeč?
Nevadí, ještě to neznamená, že *to* nepotřebujeme. Ba, naopak. Viz odst. 8...

Technologie komplexních čísel je základem „jednoho z nejhezčích a nejužitečnějších oborů matematiky“. V současnosti se stala velmi populární díky novým podnětům z komplexní dynamiky, strojního a elektrotechnického inženýrství, ve fyzice v teorii strun, která studuje konformní invarianty v kvantové teorii pole.

Jsme přesvědčeni, že právě následující dva odstavce nám nejlépe umožní vysvětlit rozdíl mezi matematikou a matematickou technologií.

V určitém smyslu jsme udělali z nouze ctnost. Nedařilo se nám napsat začátek o technologii komplexních čísel tak, abychom s tím byli spokojeni. Udělali jsme proto výkladový experiment. Jakousi didaktickou segregaci. V teoretickém odstavci jsme ukázali, jak začíná matematický výklad a jak dál pokračuje dokazováním dalších vlastností, včetně rovnosti

$$[x_1, x_2] = x_1 + \imath\, x_2.$$

V technologickém odstavci jsme výraz $x_1 + i\, x_2$ prohlásili za *komplexní číslo* a že $\imath$ je něco, co je sesláno od Boha, pro něž platí $i^2 = -1$. Nic nedokazujeme, jen konstatujeme, že používáme standardní algebraická pravidla a rychle pokračujeme ve výkladu. Konec konců, druhý způsob v sobě obsahuje jakýsi axiomatický princip.

1.2.1 Teoretický úvod

Množina všech ***komplexních čísel***

a) Komplexním číslem z nazýváme uspořádanou dvojici reálných čísel x a y a zapisujeme $z = [x, y]$. Číslu $x \in \mathbb{R}$ říkáme reálná část komplexního čísla z, číslu $y \in \mathbb{R}$ říkáme imaginární část komplexního čísla z a značíme $x = \mathrm{Re}\, z$, $y = \mathrm{Im}\, z$. Množinu všech komplexních čísel označíme $\mathbb{C}$.

b) rovnost: Dvě komplexní čísla $z_1 = [x_1, y_1]$, $z_2 = [x_2, y_2]$ jsou si rovna, právě když jsou si rovny jejich reálné části a jejich imaginární části, tj.

$$[x_1, y_1] = [x_2, y_2] \iff x_1 = x_2 \wedge y_1 = y_2$$

c) součet komplexních čísel:

$$z_1 + z_2 = [x_1, y_1] + [x_2, y_2] = [x_1 + x_2,\ y_1 + y_2];$$

d) součin komplexních čísel:

$$z_1 \cdot z_2 = [x_1, y_1] \cdot [x_2, y_2] = [x_1 \cdot x_2 - y_1 \cdot y_2,\ x_1 \cdot y_2 + y_1 \cdot x_2];$$

e) neutrální číslo

i) ke sčítání:

$$0 = [0,0], \text{ tj. } , z+0 = z; \text{ (nulou značíme dvojici nul)}$$

ii) k násobení:

$$1 = [1,0], \text{ tj. } z \cdot 1 = z; \text{ (značení...)}.$$

Poznámka pro klidné matematické svědomí:
jestliže definujeme v nějaké množině neutrální prvek, ještě to neznamená, že existuje. Například v množině přirozených čísel $\mathbb{N}$ neexistuje neutrální prvek vzhledem ke sčítání, ale vzhledem k násobení ano.

f) komplexně sdružené číslo k číslu $z = [x,y]$ je číslo:

$$\overline{z} = [x,-y].$$

g) vnoření $\mathbb{R}$ do $\mathbb{C}$: každému reálnému číslu $r \in \mathbb{R}$ přiřadíme jednoznačně dvojici $[r,0]$ a zapisujeme $r = [r,0]$.
Zde rovnítko chápeme jako přiřazení!

h) absolutní hodnota komplexního čísla $z = [x,y]$ je nezáporné (reálné) číslo:

$$|z| = (z \cdot \overline{z})^{\frac{1}{2}} = (x^2+y^2)^{\frac{1}{2}};$$

Každé komplexní číslo s, pro které platí, že $|s| = 1$ se nazývá *komplexní jednotka.*

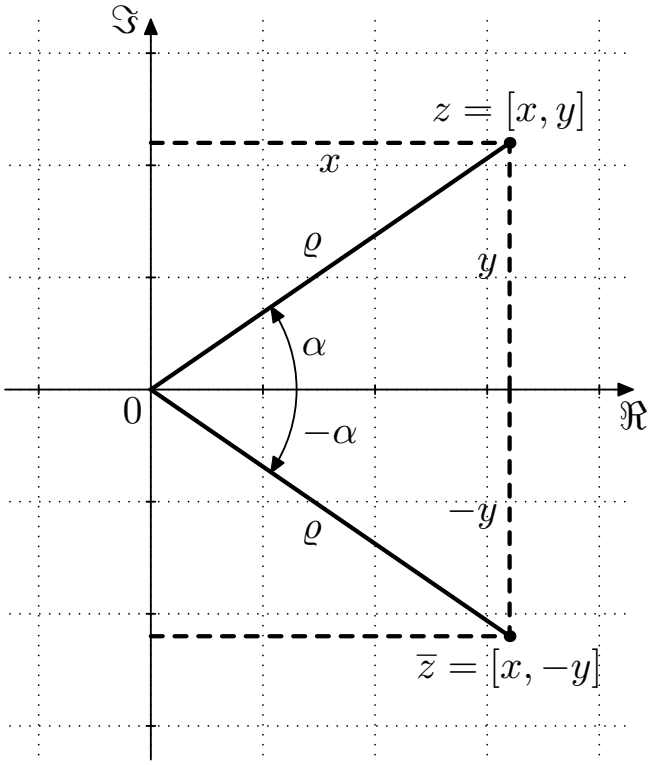
Komplexních jednotek existuje v $\mathbb{C}$ nekonečně mnoho a jejich obrazy v *Gaussově rovině* leží na jednotkové kružnici se středem v počátku *Gaussovy roviny*, tj. v obrazu čísla $0 = [0,0]$.

Komplexní jednotka, kromě reálné jednotky $1 = [1,0]$, *není* neutrálním prvkem vzhledem násobení. Imaginární jednotka je komplexní jednotka $\imath = [0,-1] = j$ (v elektrotechnice)

i) Gaussova rovina: Každé komplexní číslo $z = [x,y]$ lze znázornit v rovině, kde je zavedena kartézská soustava souřadnic $0xy$ takto:

komplexnímu číslu z přiřadíme právě jeden bod z o souřadnicích $x = \mathrm{Re}\, z$, $y = \mathrm{Im}\, z$ a obráceně, tj. každému bodu v rovině je přiřazeno právě jedno komplexní číslo. Rovině s tímto přiřazením říkáme *Gaussova rovina* (viz obrázek 1.2.1).

Blok 1.2.7: Komplexní čísla



Obrázek 1.2.1: Gaussova rovina

Důsledky definicí:

1. Operace sčítání a násobení jsou definovány tak, aby splňovaly komutativní zákon, asociativní zákon a distributivní zákon, tj platí:

komutativnost	$a+b$	$=$	$b+a$	;	$a\cdot b$	$=$	$b\cdot a$;
asociativnost	$a+(b+c)$	$=$	$(a+b)+c$	;	$a\cdot(b\cdot c)$	$=$	$(a\cdot b)\cdot c$;
distributivnost	$a\cdot(b+c)$	$=$	$a\cdot b+a\cdot c$	;			

   Důkaz se opírá platnost těchto zákonů v $\mathbb{R}$ a o princip vnoření $\mathbb{R}$ do $\mathbb{C}$.

2. Podíl $p=\frac{a}{b}$ komplexních čísel a, $b\neq 0$, je takové komplexní číslo, které je kořenem rovnice $a=b\cdot p$; přesněji: Ke každým dvěma komplexním číslům a, b, kde $b\neq 0$, existuje takové komplexní číslo p, pro které platí: $a=b\cdot p$.

   Číslo p se nazývá *podíl komplexních čísel* a,b a zapisuje se: $p=\frac{a}{b}$.

   K důkazu opět vystačíme s definicí součinu.

3. Algebraický tvar: Pro každé komplexní číslo $z=[x,y]$ platí rovnosti:

   $$z=[x,0]+[0,y]=[x,0]+[0,1]\cdot[y,0]=x+\imath y.$$

4. Mocnina a odmocnina - bude podrobněji zpracována v samostatném odstavci.

Poznámka:
Shrneme poznatky o operacích v číselným množinách, které operace s čísly lze v dané

číselné množině provádět, aby výsledek bylo číslo ze stejné číselné množiny, tj. aby daná množina byla *uzavřená* vzhledem k dané operaci:

- $\mathbb{N}$ - sčítání, násobení, umocňování přirozeným exponentem;
- $\mathbb{Z}$ - sčítání, odčítání, násobení, umocňování přirozeným exponentem;
- $\mathbb{Q}$ - sčítání, odčítání, násobení, dělení (kromě dělení nulou!), umocňování celočíselným exponentem (s výjimkou 0!);
- $\mathbb{R}$ - sčítání, odčítání, násobení, dělení (kromě dělení nulou!), umocňování a odmocňování nezáporných reálných čísel reálným exponentem; například v $\mathbb{R}$ neumíme stanovit:

$$(-\pi)^{\pi}, \text{ (ale } \pi^{\pi} \text{ ano)}$$

  viz každá obyčejná kalkulačka.
- $\mathbb{C}$ - všechny operace. Pro technologické využití velkého potenciálu komplexních čísel, musíme odvodit další důsledky definicí, například funkci *obecná mocnina*.

1.2.2 Technologický úvod

V rovině zvolíme dvě navzájem kolmé přímky, vodorovnou a svislou. Průsečík přímek označíme $[0,0]$, vodorovnou přímku nazveme *reálná osa*, svislou *imaginární osa*.

Každému bodu z této roviny přiřadíme *uspořádanou dvojici* reálných čísel. Uspořádaná dvojice $\imath = [0,1]$ se nazývá *imaginární jednotka*, dvojice $[1,0]$ se nazývá *reálná jednotka*.

Jestliže pro všechny uspořádané dvojice reálných čísel definujeme relaci rovnosti a operace sčítání, odčítání, násobení, dělení, umocňování, odmocňování, dostaneme strukturu, kterou nazýváme *množina komplexních čísel* a značíme $\mathbb{C}$.

Základní vlastnosti všech ***komplexních čísel***:

1. komplexní číslo z je číslo:

$$z = x + \imath y, \ \imath^2 = -1 \ , \ x \in \mathbb{R} \ , \ y \in \mathbb{R}.$$

$$\begin{array}{rcll} \mathrm{Re}\, z & = & x & = \text{ se nazývá } \textit{reálná část} \text{ čísla} z, \\ \mathrm{Im}\, z & = & y & = \text{ se nazývá } \textit{imaginární část} \text{ čísla} z \end{array}$$

2. Ke každému číslu $z \in \mathbb{C}$ existuje právě jeden bod Z *Gaussovy roviny* s kartézskými souřadnicemi.

3. *Reálné* číslo:
$$\mathrm{Im}\, z = 0$$

4. *Ryze imaginární* číslo:
$$\mathrm{Re}\, z = 0;$$

5. *Imaginární jednotka* je takové komplexní číslo $\imath$, pro které platí:
$$\imath^2 = -1;$$

6. Číslo *komplexně sdružené* k číslu $z = x + \imath y$ je číslo:

$$\overline{z} = x - \imath y;$$

součin $z \cdot \overline{z} = x^2 + y^2$ je nezáporné reálné číslo.

7. *Rovnost* komplexních čísel:

Dvě komplexní čísla z_1, z_2 jsou si rovna, právě tehdy když jsou si rovny jejich reálné části a zároveň jejich imaginární části, tj.:

$$z_1 = z_2 \quad \Leftrightarrow \quad (x_1 = x_2) \wedge (y_1 = y_2);$$

8. *Součet* komplexních čísel:

$$z_1 + z_2 = (x_1 + x_2) + \imath(y_1 + y_2);$$

tj. sečteme reálné a imaginární části čísel z_1 a z_2.

9. *Rozdíl* komplexních čísel:

$$z_1 - z_2 = (x_1 - x_2) + \imath(y_1 - y_2);$$

tj. odečteme reálné a imaginární části čísel z_1 a z_2.

10. *Součin* komplexních čísel:

Formálně považujeme komplexní čísla $z_1 = x_1 + \imath y_1$ a $z_2 = x_2 + \imath y_2$ za dvojčleny a aplikujeme pravidlo o *násobení dvojčlenů*:

$$z_1 \cdot z_2 = (x_1 + \imath y_1) \cdot (x_2 + \imath y_2) = x_1 \cdot x_2 + x_1 \cdot \imath y_2 + x_2 \cdot \imath y_1 + \imath y_1 \cdot \imath y_2 =$$

$$= (x_1 x_2 - y_1 y_2) + \imath(x_1 y_2 + x_2 y_1).$$

11. *Podíl* komplexních čísel:

$$\frac{z_1}{z_2} = \frac{x_1 + \imath y_1}{x_2 + \imath y_2}\bigg| \cdot \frac{x_2 - \imath y_2}{x_2 - \imath y_2}\ ,\ \left(\text{ tj. } \frac{\overline{z}_2}{\overline{z}_2}\right)\ ;$$

$$\frac{z_1}{z_2} = \frac{(x_1 x_2 + y_1 y_2) + \imath(x_2 y_1 - x_1 y_2)}{x_2^2 + y_2^2} = \frac{x_1 x_2 + y_1 y_2}{x_2^2 + y_2^2} + \imath \frac{x_2 y_1 - x_1 y_2}{x_2^2 + y_2^2}.$$

12. Absolutní hodnota komplexního čísla $z = x + \imath y$ je reálné číslo:

$$|z| = \sqrt{x^2 + y^2}\ \geq 0;$$

13. Polární (goniometrický) tvar komplexního čísla je:

$$z = \varrho(\cos\alpha + \imath \sin\alpha);\ z \neq 0,\ \alpha \in \mathbb{R}$$

kde: $\varrho = |z| = \sqrt{x^2 + y^2}$
$\alpha = \operatorname{Arg} z = \arg z + 2k\pi$ se nazývá *argument komplexního čísla*

Z periodicity funkcí *sinus* a *kosinus* (α je velikost orientovaného úhlu - viz obrázek 1.2.2) plyne, že:

$$\alpha = \alpha_0 + 2k\pi,\ k \text{ je celé číslo},\ \alpha_0 \in < -\pi, \pi > .$$

Množinu všech reálných čísel α pro které platí polární tvar komplexního čísla z označíme:

$$\text{Arg } z = \{\alpha \in \mathbb{R} \ :\ \alpha_0 + 2k\pi,\ k \in \mathbb{Z}\}$$

a nazýváme ji *argumentem komplexního čísla* z. Zapisujeme také:

$$\text{Arg } z = \arg z + 2k\pi,$$

$\arg z = \alpha_0$ nazýváme *hlavní hodnotou argumentu* komplexního čísla z.

14. Exponenciální tvar komplexního čísla:

$$z = \varrho \cdot e^{\imath\alpha}$$

kde: ϱ, α - viz formulace polárního tvaru komplexního čísla
Přiřazovací rovnost:

$$e^{\imath\alpha} = \cos\alpha + \imath \sin\alpha,$$

se nazývá *Eulerova formule* a zde ji přijmeme jako axiom. Algebraické zdůvodnění spočívá v tom, že mocniny čísla $a = \cos\alpha + \imath\sin\alpha$ splňují pravidla z odst. 1.2.3. Číslo e je iracionální a je to limita posloupnosti:

$$\lim_{n\to\infty} \left(\frac{1+1}{n}\right)^n \approx 2,718281828459045235360287471352.$$

15. Termínem *komplexní jednotka* nazýváme taková komplexní čísla pro která platí:

$$s = \cos\alpha + \imath\sin\alpha = e^{\imath\alpha},$$

$$|s| = 1.$$

je zřejmé, že:

(a) komplexních jednotek je nekonečně mnoho,

(b) v Gaussově rovině leží na kružnici s poloměrem 1 a středem v počátku.

(c) imaginární jednotka $\imath$ je také komplexní jednotka:

$$|\imath| = |0 + \imath| = \sqrt{0^2 + 1^2} = 1$$

16. *Součin* komplexních čísel v *polárním nebo exponenciálním tvaru*:
Je-li: $z_1 = \varrho_1(\cos\alpha_1 + \imath\sin\alpha_1) = \varrho_1 \cdot e^{\imath\alpha_1}$,
$z_2 = \varrho_2(\cos\alpha_2 + \imath\sin\alpha_2) = \varrho_2 \cdot e^{\imath\alpha_2}$,
potom:

$$z_1 \cdot z_2 = \varrho_1 \cdot \varrho_2(\cos(\alpha_1 + \alpha_2) + \imath\sin(\alpha_1 + \alpha_2)) = \varrho_1 \cdot \varrho_2 \cdot e^{\imath(\alpha_1+\alpha_2)}.$$

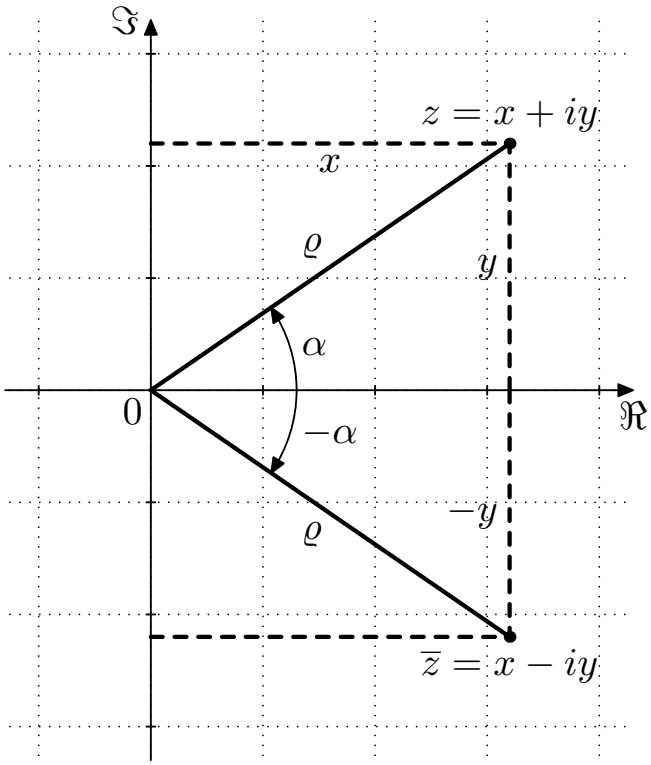
17. *Podíl* komplexních čísel v *polárním nebo exponenciálním tvaru*:

    Je-li: $z_1 = \varrho_1(\cos\alpha_1 + \imath\sin\alpha_1) = \varrho_1 \cdot e^{\imath\alpha_1}$,
    $z_2 = \varrho_2(\cos\alpha_2 + \imath\sin\alpha_2) = \varrho_2 \cdot e^{\imath\alpha_2}$,

    potom:

$$\frac{z_1}{z_2} = \frac{\varrho_1}{\varrho_2}\left(\cos(\alpha_1 - \alpha_2) + \imath\sin(\alpha_1 - \alpha_2)\right) = \frac{\varrho_1}{\varrho_2} \cdot e^{\imath(\alpha_1 - \alpha_2)}.$$

Blok 1.2.8: Vlastnosti komplexních čísel



Obrázek 1.2.2: Gaussova rovina

1.2.3 Mocniny a odmocniny

Umocňování komplexních čísel:

základním principem je opakované násobení, pro $z \in \mathbb{C}$, $n \in \mathbb{N}$:

1. $z^n = z \cdot z \cdot z \ldots z$,
   $z^n = (x + \imath y)^n$ podle *binomické formule*
   $z^n = \varrho^n(\cos(n \cdot \alpha + \imath\sin(n \cdot \alpha)) = \varrho^n \cdot e^{i \cdot n \cdot \alpha}$ je *Moivreova formule*
2. $z^0 = 1$, $z \neq 0$,
3. $z^{-n} = \frac{1}{z^n}$,

4. 0^0 je pouze symbol(neurčitý výraz), který může označovat číslo, ale nemusí.
5. pro: $a^m \cdot a^n = a^{m+n}, \quad a \in \mathbb{C}, m, n \in \mathbb{N}$;
6. pro: $(a^m)^n = a^{mn}, \quad a \in \mathbb{C}, m, n \in \mathbb{N}$;
7. pro: $(ab)^m = a^m \cdot b^m, \quad a, b \in \mathbb{C}, m \in \mathbb{N}$;
8. pro: $\left(\frac{a}{b}\right)^m = \frac{a^m}{b^m}, \quad a, b \in \mathbb{C}, b \neq 0, m \in \mathbb{N}$;
9. $\frac{a^m}{a^n} = a^{m-n}, \quad a \in \mathbb{C}, \quad m, n \in \mathbb{N}, a \neq 0$;

Blok 1.2.9: Mocniny komplexních čísel

Odmocňování komplexních čísel:

základním principem je mocnina $w^n = z$, $n \in \mathbb{N}$.

1. $n-$*tá* odmocnina z nezáporného reálného čísla a je jediné nezáporné reálné číslo r takové, že platí:
$$r^n = a.$$
značíme:
$$r = \sqrt[n]{a}^{\mathrm{R}}$$
Historická poznámka: znak $\sqrt{}$ je stylizované písmeno r. V latině se nazývá *radix*, v angličtině *root (square root)* a znamená *kořen.*
2. pro: $a^{\frac{m}{n}} = \left(a^{\frac{1}{n}}\right)^m = (\sqrt[n]{a})^m = \sqrt[n]{a^m}, \quad m, n \in \mathbb{N}, a \geq 0.$
3. $n-$*tá* odmocnina z komplexního čísla z je takové komplexní číslo w, že platí:
$$w^n = z.$$
značíme:
$$w = \sqrt[n]{z}^{\mathrm{C}}$$
4. Pro z v polárním nebo exponenciálním tvaru lze z uvedené podmínky stanovit:
$$\sqrt[n]{z}^{\mathrm{C}} = \sqrt[n]{\varrho}^{\mathrm{R}}(\cos\frac{\alpha}{n} + \imath \sin\frac{\alpha}{n}) = \sqrt[n]{\varrho}^{\mathrm{R}} \cdot e^{\imath\frac{\alpha}{n}}.$$

V rámci tohoto odstavce u znaku odmocnítka píšeme horní pravý index R v případě, že jde o „reálné“ odmocnítko a píšeme horní pravý index C v případě, že jde o „komplexní“ odmocnítko. Dále to již používat nebudeme a z kontextu to bude (snad) zřejmé.

Blok 1.2.10: Odmocniny komplexních čísel

Problém obecné mocniny a^b nejen pro reálná čísla, ale i pro komplexní čísla, nelze definovat bez poznatků o funkcích a limitách. Prosíme zájemce o strpení a případně o nahlédnutí do odpovídajícího sešitu.

1.2.4 Struktury

Matematická struktura je množina spolu s dodatečnou informací, například *algebraickými operacemi, relacemi* apod. Číselné množiny (obory) jsou příkladem základních matematických struktur.

Matematické struktury jsou zaváděny proto, aby bylo možno dokazovat matematická tvrzení pro mnoho různých objektů najednou. Například dokáže-li se nějaké tvrzení pro každý lineární prostor, není již nutné jej dokazovat zvlášť pro vektory v rovině, zvlášť pro prostor všech funkcí apod., protože každá z těchto množin tvoří lineární prostor. Totéž platí pro další matematické struktury, například grupy, uspořádané množiny, svazy apod.

Používá se též synonymum abstraktní struktura, které zdůrazňuje kontrast mezi konkrétními matematickými objekty (nějaká věta může něco tvrdit o přímkách, jiná o funkcích na reálných číslech...) a obecnými strukturami (kdy věta o grupách něco tvrdí o všech nejrůznějších množinách, které jsou grupami).

Příkladem matematické struktury je monoid, který je definován jako množina $\mathcal{M}$ s binární operací $\circ$, která:

- je asociativní, tj. pro každé $x, y, z \in \mathcal{M}$ platí $(x \circ y) \circ z = x \circ (y \circ z)$, pro jednoduchost lze tedy používat zápis $x \circ y \circ z$
- má neutrální prvek e takový, že $x \in \mathcal{M}$ platí $x \circ e = x$ a $\cdot e \circ x = x$.

Příkladem monoidu je:

- množina celých čísel se sčítáním
- množina všech lichých čísel s násobením
- množina všech čtvercových matic velikosti 4 (nebo kterékoli jiné) s maticovým násobením
- množina všech permutací na jakékoli množině s operací skládání zobrazení.

Příklad struktur, které nejsou monoidy:

- množina kladných reálných čísel s dělením (není asociativní)
- celá čísla od 1 do 10 se sčítáním (výsledek operace někdy neleží v dané množině, například $9 + 2 = 11$)
- celá čísla s operací dělení (není asociativní a dělení nulou není definováno)
- celá kladná čísla se sčítáním (nemá neutrální prvek)

Nesprávné manipulace (nedodržování pravidel operací tj. nedodržení zákonů dané struktury) vedou k nesprávným výsledkům. Nemusí být jednoduché chybu najít.

Uveďme jeden známý případ:

Nechť pro dvě libovolná reálná čísla x, y platí rovnost $x = y$. Obě strany rovnosti vynásobme číslem x:

$$x^2 = xy$$

a odečtěme od každé strany mocninu y^2. Dostaneme:

$$x^2 - y^2 = xy - y^2, \text{ tj. } (x+y)(x-y) = y(x-y).$$

Krácením výrazem $(x-y)$ obdržíme:

$$x + y = y, \text{ odtud pak } x = 0.$$

Závěr (!?): libovolné reálné číslo x se rovná nule.

Kapitola 2
Uspořádání reálných čísel a omezené číselné množiny

Obsah kapitoly

2.1 Axiomy uspořádání v $\mathbb{R}$

1. ***trichotomie:***
   Pro každá dvě čísla $x \in \mathbb{R}$, $y \in \mathbb{R}$ platí právě jeden ze vztahů:
   $$x < y, \quad x = y, \quad x > y.$$
   slovně: je pravdivý právě jeden z výroků.

2. ***tranzitivnost:***
   Pro každá dvě čísla $x \in \mathbb{R}$, $y \in \mathbb{R}$ platí:
   $$\forall x, y, z \in \mathbb{R}: \quad (x < y \wedge y < z) \quad \Rightarrow \quad x < z;$$
   slovně: je-li x menší než y a y je menší než z, pak je x menší než z

3. ***monotonie sčítání:***
   Pro každá dvě čísla $x \in \mathbb{R}$, $y \in \mathbb{R}$ platí:
   $$\forall x, y, z \in \mathbb{R}: \quad x < y \quad \Leftrightarrow \quad x + z < y + z;$$

4. ***monotonie násobení:***
   Pro každá dvě čísla $x \in \mathbb{R}$, $y \in \mathbb{R}$ platí:
   $$\forall x, y, z \in \mathbb{R}: \quad (x > 0 \wedge y > 0) \quad \Rightarrow \quad xy > 0;$$
   (obrácená implikace neplatí !)
   kde znak „$\Leftrightarrow$“ je znak ekvivalence a znak „$\wedge$“ je znak konjunkce.

Říkáme, že množina reálných čísel je ***uspořádaná***.

Těmito axiomy (víc jich není třeba) zavádíme vztahy (*relace*):

- „je menší než“
- „je větší než“

V množině komplexních čísel tyto axiomy nelze zavést!

Blok 2.1.1: Uspořádání v $\mathbb{R}$

U komplexních čísel neumíme rozhodnout, které ze dvou komplexních čísel (s nenulovou imaginární částí) je větší; umíme rozhodnout pouze o rovnosti.

Úkoly:

1. Co znamená zápis:
$$a \leq b?$$
(a je menší nebo rovno b; disjunkce dvou výroků.)

2. Co je správně (tj. která nerovnost je pravdivý výrok):
$$3 < 4, \quad 3 \leq 4, \quad 4 \leq 4, \quad 4 < 4$$
(pouze poslední výrok je nepravdivý)

3. Jaký je rozdíl mezi rovností a rovnicí?
(Rovnost je vztah (výrok). Rovnice je úloha!)

4. Jaký je rozdíl mezi nerovností a nerovnicí?
(Nerovnost je vztah (výrok); nerovnice je úloha - viz uvedené úlohy.)

5. Stanovte $x \in \mathbb{R}$ takové, aby platilo:
$$3 + x < 7.$$

6. Stanovte $x \in \mathbb{R}$ takové, aby platilo:
$$5 - 3x > 17.$$

7. Prověřte, zda platí:
$$2^{3,14} < 1,9^{\pi}?$$

2.2 Intervaly - podmnožiny množiny $\mathbb{R}$

1. $\langle a,b\rangle = \{x \in \mathbb{R} : \ a \leq x \leq b\}$ - uzavřený interval;
2. $(a,b) = \{x \in \mathbb{R} : \ a < x < b\}$ - otevřený interval;
3. $(a,b\rangle = \{x \in \mathbb{R} : \ a < x \leq b\}$ - polouzavřený (polootevřený) interval;
4. $\langle a,b) = \{x \in \mathbb{R} : \ a \leq x < b\}$ - polouzavřený (polootevřený) interval;
5. $\langle a,+\infty) = \{x \in \mathbb{R} : \ a \leq x\}$
6. $(a,+\infty) = \{x \in \mathbb{R} : \ a < x\}$
7. $(-\infty,b\rangle = \{x \in \mathbb{R} : \ x \leq b\}$
8. $(-\infty,b) = \{x \in \mathbb{R} : \ x < b\}$

Blok 2.2.2: Intervaly

2.3 Maximum, minimum, supremum, infimum

Číselná množina $\mathcal{A} \subset \mathbb{R}$ je ***shora omezená***, když:

$$\exists c_2 \in \mathbb{R}, \quad \forall x \in \mathcal{A} : \ x \leq c_2.$$

Základní vlastnosti:

1. Existuje reálné číslo c_2 takové, že každé číslo $x \in \mathcal{A}$ je nejvýše rovno číslu c_2.

Blok 2.3.3: Omezenost shora

Číselná množina $\mathcal{A} \subset \mathbb{R}$ je ***zdola omezená***, když:

$$\exists c_1 \in \mathbb{R}, \quad \forall x \in \mathcal{A} : \ x \geq c_1.$$

Základní vlastnosti:

1. Existuje reálné číslo c_1 takové, že každé číslo $x \in \mathcal{A}$ není menší než číslo c_1.

Blok 2.3.4: Omezenost zdola

Číselná množina $\mathcal{A} \subset \mathbb{R}$ je ***omezená***, když:

$$\exists c_1, c_2 \in \mathbb{R}, \quad \forall x \in \mathcal{A} : \; c_1 \leq x \leq c_2.$$

Základní vlastnosti:

1. Existují dvě reálná čísla c_1 a c_2 taková, že každé číslo $x \in \mathcal{A}$ je nejvýše rovno číslu c_2 a zároveň není menší než číslo c_1.

Blok 2.3.5: Omezenost oboustranná

Úkol:
Formulujte následující výroky pro množinu $\mathcal{M} \subset \mathbb{R}$:

1. množina $\mathcal{M}$ je shora neomezená
2. množina $\mathcal{M}$ je zdola neomezená
3. formulujte negaci předchozích výroků

Maximum číselné množiny $\mathcal{A} \subset \mathbb{R}$:

$$\text{je takové reálné číslo } M, \text{ pro které platí } \forall x \in \mathcal{A} : \; x \leq M;$$

Označujeme: $M = \max \mathcal{A}.$
Čteme:

M je největší číslo množiny $\mathcal{A}$

Blok 2.3.6: Maximum číselné množiny

Minimum číselné množiny $\mathcal{A} \subset \mathbb{R}$:

$$\text{je takové reálné číslo } m, \text{ pro které platí } \forall x \in \mathcal{A} : \; x \geq m;$$

Označujeme: $m = \min \mathcal{A}.$
Čteme:

m je nejmenší číslo množiny $\mathcal{A}$

Blok 2.3.7: Minimum číselné množiny

Supremum číselné množiny $\mathcal{A} \subset \mathbb{R}$:

je takové reálné číslo S, pro které platí požadavky:

1. $\forall x \in A: \; x \leq S$;
2. $\forall x' \in \mathbb{R}: \; x' < S \quad \exists x'' \in A: \quad x'' > x'$;

Označujeme: $S = \sup \mathcal{A}$.
Čteme:
Pro každé reálné číslo x' menší než S existuje v množině $\mathcal{A}$ reálné číslo x'', které je větší než číslo x'.

Blok 2.3.8: Supremum číselné množiny

Infimum číselné množiny $\mathcal{A} \subset \mathbb{R}$:

je takové reálné číslo S, pro které platí požadavky:

1. $\forall x \in A: \; x \geq s$;
2. $\forall x' \in \mathbb{R}: \; x' > s \quad \exists x'' \in A: \quad x'' < x'$;

Označujeme: $s = \inf \mathcal{A}$.
Čteme:
Pro každé reálné číslo x' větší než s existuje v množině $\mathcal{A}$ reálné číslo x'', které je menší než číslo x'.

Blok 2.3.9: Infimum číselné množiny

Například:

$\max \langle -2, 1\rangle$	$= 1$;	$\sup \langle -2, 1\rangle$	$= 1$
$\min \langle -2, 1)$	neexistuje;	$\sup \langle -2, 1)$	$= 1$;
$\max \langle -5, +\infty)$	neexistuje;	$\sup \langle -5, +\infty)$	neexistuje;

Otázky:

1. Je maximum (minimum) vždy supremem (infimem)?
2. Je supremum (infimum) vždy maximem (minimem)?

Absolutní hodnota reálného čísla x je větší z čísel x a $-x$; Značíme:

$$|x| = \max\{x, -x\}$$

Blok 2.3.10: Absolutní hodnota

Důsledky definice:

Důsledek 1: Absolutní hodnota čísla x je vždy nezáporné číslo, tj. $|x| \geq 0$.

Důsledek 2: pro $x > 0$ je $|x| = x$;
pro $x < 0$ je $|x| = -x$;
pro $x = 0$ je $|x| = 0$;

Důsledek 3: Množina $\mathcal{B} \subset \mathbb{R}$ je omezená právě tehdy, když
$\exists c > 0, \forall x \in \mathcal{B}: \quad |x| \leq c.$

Důsledek 4: $\forall a, b \in \mathbb{R}: \quad |a+b| \leq |a| + |b|$ (trojúhelníková nerovnost)

Důsledek 5: $||a| - |b|| \leq |a-b| \leq |a| + |b|$

Důsledek 6: $\forall a \in \mathbb{R}:$

$$|a|^2 = a^2;$$
$$\sqrt{a^2} = |a|$$
$$|ab| = |a| \cdot |b|; \quad \left|\frac{a}{b}\right| = \frac{|a|}{|b|}, \; b \neq 0$$

Blok 2.3.11: Důsledky definice absolutní hodnoty

Kapitola 3
Číselné soustavy

Obsah kapitoly

Termín *číselná soustava* je obvykle chápán ve významu jakési reprezentace, zápisu čísel pomocí určitých symbolů. Způsobů, jak reprezentovat, zapisovat čísla je poměrně značné množství. Ne všechny se však „v průběhu věků“ široce rozšířily, popřípadě udržely. Pouze některé reprezentace čísel však měly díky svým vlastnostem zásadní, přímo fundamentální vliv na další technologický rozvoj civilizace, zejména pak matematických věd.

Jedno z možných rozdělení, klasifikací číselných soustav je:

- nepoziční číselné soustavy,
- poziční číselné soustavy,
- ostatní.

3.1 Nepoziční číselné soustavy

Nepoziční číselná soustava je způsob reprezentace, zápisu čísel pomocí specifických symbolů, číslic. Hodnota čísla není dána pozicí číslice v čísle, ale (například) počtem výskytů určitého symbolu, cifry.

Nepoziční číselné soustavy jsou nyní považovány za zastaralé a používají se výjmečně, případně pouze ve specifických (nejčastěji historicky daných) situacích, či souvislostech. Například při výuce elemetárních základů matematiky, sčítání a odčítání a na prvním stupni základní školy se často používá pomůcka kuličkové počítadlo, často zvaná „vlaštovičky“.

Mezi nepoziční číselné soustavy řadíme zejména:

- římské číslice - původ symbolů:
- egyptské číslice
- řecké číslice
- etruské číslice

Případně další, poněkud „speciální typy“ soustav.

3.1.1 Římské číslice

Římská čísla (číslice) vznikla přirozenou cestou, kdy jednotlivé symboly mají svůj původ v používání lidské ruky, jako „pomocníka“ pro vyjádření potřebného množství. Římané počítali na prstech

- *symboly I, II, III*:
  Čísla jako $1, 2$ a 3 a jim odpovídající znaky I, II a III graficky vyjadřují jednotlivé prsty na ruce.
- *symboly V a X*:
  Také tato dvě římská čísla mají svůj původ v lidské ruce. Římská číslice V (5) je vyjádřením dlaně s pěti prsty - V tvoří tvar mezi palcem a malíčkem. Římská číslice X (10) jsou dvě dlaně u sebe (10 prstů).
- *symboly L a C*:
  Latinsky sto je *centum* - odtud C. Padesát je polovina ze stovky. L tedy vzniklo „rozpůlením“ znaku pro 100 (C).
- *symboly D a M*:
  Tisíc je latinsky *mille*, odtud M pro 1000. Znak D pro 500 vznikl opět grafickým „půlením“ znaku M, tentokrát svisle. Vznikl tak znak podobný písmenu D.

Římské číslice nemají symbol pro číslo „nula“, přestože její význam dobře znali.

Římané zapisovali čísla:

- číslo 4 jako IIII,
- číslo 40 jako XXXX,
- číslo 999 jako DCCCCLXXXXVIIII.

Ke zkrácení zápisu (z úsporných důvodů) došlo až ve středověku, kdy byl zaveden systém předcházení menšího symbolu před větším k vyjádření čísla:

- IV = 4
- IX = 9
- XL = 40
- XC = 90
- CD = 400
- CM = 900

a jiné kombinace povoleny nebyly. Takže číslo 999 bylo nutné zapisovat ve tvaru:

$$999 = CMXCIX,$$

a nikoliv ve tvaru:

$$999 = IM.$$

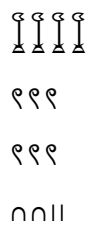
3.1.2 Egyptské číslice

Starověké egyptské číslice byly používány ve starověkém Egyptu přibližně od roku 4000 př. n. l. do začátku 1. tisíciletí n. l. Staří egypťané používali desítkovou soustavu, nikoliv však poziční, ale pro každou mocninu deseti měli jiný znak, který se podle potřeby opakoval:

1	=	𓏺	10 000	=	𓂭
10	=	𓎆	100 000	=	𓆐
100	=	𓍢	1 000 000	=	𓁨
1 000	=	𓆼			

Tabulka 3.1.1: Egyptské číselné symboly

Násobky těchto hodnot byly vyjadřovány opakováním příslušných symbolů tolikrát, kolikrát bylo třeba. Například na rytině z Karnaku je číslo 4622 zapsáno jako:



Tabulka 3.1.2: Zápis čísla 4622

Starověcí Egypťané měli poměrně volný způsob zápis čísel. Egyptské hieroglyfy se mohou psát oběma směry (zprava doleva i zleva doprava), dokonce i vertikálně. Příklad v tabulce 3.1.2 je psán zleva doprava a odshora dolů.

Pro sčítání a odčítání Egypťané používají symboly (pata):

sčítání: 𓂝 ; odčítání: 𓂡

Pokud paty směřují ve směru psaní, jde o symbol pro sčítání (plus). V opačném případě se jedná o symbol pro odčítání (mínus).[1]

3.1.3 Řecké číslice

Řekové používali (používají) k zápisu číslic písmenné symboly řecké alfabety. Díky tomu, že číselný systém je nepoziční, opakovali (podobně jako Egypťané) použitý symbol tolikrát, kolikrát bylo potřeba.

Prvních devět znaků řecké alfabety odpovídalo postupně číslům $1, 2, 3$, až 9. Druhých deset znaků alfbety odpovídalo postupně číslům $10, 20, 30$, až 100. Pro některé vyšší řády se pak používaly speciální symboly, jelikož nebyl k dispozici potřebný počet znaků alfabety.

Řecké číslice se v Řecku v současnosti používají podobně jako římské číslice v Česku, pouze příležitostně a i k podobným účelům.

Starověké Řecko dalo světu mnoho vynikajících matematiků, jejichž díly je možné se inspirovat i v současnosti. Nejznámější jména jsou pravděpodobně Pythagoras, Archimédes, případně Thales z Milétu. Nejsou však jediní, kteří obohatili matematické poznání. Připomeňme několik méně známých jmen:

Apollónios z Pergy byl starověký řecký geometr a matematik. Narodil se kolem roku 200 př. n. l. ve městě Perga v Pamýlii, působil v egyptské Alexandrii a v Pergamu. Proslul zejména dílem Kónika – Pojednání o kuželosečkách.

Aristarchos ze Samu (asi 310 – 230 př. n. l.) byl řecký matematik a astronom, tvůrce heliocentrického modelu vesmíru, někdy nazývaný starověký Koperník. Aristarchova vědecká zjištění a názory ale pobuřovaly veřejné mínění té doby. Prohlásil, že vesmír je nekonečný a hvězdy jsou jiná Slunce. To byly v té době kacířské myšlenky a Aristarchos byl obžalován z bezbožnosti. Jeho model byl zamítnut a posléze na dlouho zapomenut.

Diofantos z Alexandrie byl starověký řecký matematik působící ve 3. století n. l. v egyptské Alexandrii. Je prvním matematikem, který řešil ryze algebraické problémy a zavedl pro ně promyšlený systém značek. Své nejdůležitější dílo, Aritmetika sepsal ve 13 knihách. Bohužel, do dnešní doby se zachovala přibližně polovina jeho díla. Jeho jméno nesou rovnice, které studoval (diofantické rovnice).

Eratosthenés z Kyrény (mezi 276-272 v Kyréně – 194 př. n. l. v Alexandrii) byl matematik, astronom a zřejmě největší geograf antického Řecka. Působil též jako správce alexandrijské knihovny.

Je znám jako autor jednoho z prvních výpočetních postupů vůbec, dnes bychom řekli algoritmu, pro nalezení všech prvočísel v daném intervalu.

Hérón Alexandrijský zvaný „Méchanikos“, byl starověký matematik a vynálezce. Působil v 1. století n.l. (ca 10 - 70). Zabýval se praktickými úlohami z matematiky, mechaniky a dalších oblastí fyziky. I když je jeho dílo mnohem obsáhlejší, je dnes znám především jako autor *Heronova vzorce* pro výpočet obsahu trojúhelníku. Sestrojil takzvanou *Heronovu baňku*, což byl první pokus o vytvoření parního stroje.

Ve svém díle Automata se také věnuje automatizaci a tak je právě toto dílo označováno jako první kniha o robotech.

Hypatia z Alexandrie (mezi lety 350(370) – 415, ev. 416) byla pohanská novoplatonská filosofka a první historicky známá matematička žijící v závěru řecko-římské doby v egyptské Alexandrii. Je považovaná za jednu z hlavních postav pohanské filosofie a víry té doby. Svými současníky byla pokládána za charismatického učitele a významného učence se značným rozsahem odbornosti. Současně požívala ve městě všeobecné obliby a úcty.

Tím bohužel dráždila představitele křesťanské církve, tehdy již dominující náboženskou sílu. Na popud mocichtivého, svárlivého a agresivního křesťanského patriarchy Kyrila (Cyrila) Alexandrijského byla pod falešnou záminkou brutálně napadena a zavražděna zfanatizovaným davem.

3.1.4 Etruské číslice

Ve starší populárně-historické literatuře lze nalézt jazyk Etrusků v jedné skupině s protoindickým písmem a krétským lineárním písmem A. Což bohužel znamená, že stále zůstává v kolonce „dosud nevyluštěno“. Tedy, částečně.

Texty, které jsou k dispozici, jsou v částečně srozumitelné. Protože však jde vesměs o texty krátké, chybí představa o jazyce jako celku. Protože značná část textů pochází z hrobů, rozumíme slovům označujícím příbuznost (otec toho a toho), číslovkám (zemřel v tolika a tolika letech).

Uvádí se, že etruské číslovky sloužily jako základ, východisko pro římské číslice. To je bohužel zatím vše, co se povedlo etruskologům zjistit. Nelze tak s jistotou říci jaký byl vlastně jejich číselný systém, jak prováděli početní operace, zda objevili nějaké základní matematické zákonitosti. Nelze s jistotou říci jak Římané uzpůsobili původní etruský číselný systém. I to se bohužel v dějinách stává.

3.2 Poziční číselné soustavy

Druhá skupina číselných soustav, *poziční číselné soustavy* je dnes převládající systém reprezentace, zápisu čísel. Vyznačuje se zejména velkou pružností při zobrazování velkých hodnot při současné potřebě poměrně malého počtu symbolů.

Poziční číselné systémy jsou založeny na vlastnosti, že *hodnota číselného symbolu* je dána *jeho pozicí* ve vyjádření čísla. Nejčastěji používané poziční číselné soustavy, hlavně díky rozvoji výpočetní techniky, jsou:

- desítková soustava (dekadická), viz Al-Chvárizmí: Algorithmi de numero indorum
- dvojková soustava (binární)
- osmičková soustava (oktalová)
- šestnáctková soustava (hexadecimální)

Připomeňme některé další, spíše historické poziční číselné soustavy, například:

- *dvacítková soustava starých Mayů*,

- *dvanáctková soustava*,
- *šedesátková soustava starých Sumerů*.

Termín „dvojková soustava“, příp. „desítková soustava“ v podstatě říká, kolika různými symboly (znaky) daná soustava „disponuje“. Je to *přirozené číslo* a označuje se termínem *radix*, z anglického *báze, kořen*, či *základ*. Používá se obvykle symbol r.

Dále platí „dohoda“ (čti: je stanoveno), že symbol umístěný *více vlevo*[1] má větší význam,*váhu* v daném čísle, než symbol umístěný více vpravo. Velikost tohoto významu, váhy, je určena právě hodnotou *radixu*.

Úkol:
Zjistěte význam (váhu) jednotlivých číslic v čísle 1234 zapsané v desítkové soustavě

$$\begin{aligned} 1234 &= 1\cdot 1000 + 2\cdot 100 + 3\cdot 10 + 4\cdot 1 \\ &= 1\cdot 10^3 + 2\cdot 10^2 + 3\cdot 10^1 + 4\cdot 10^0 \end{aligned}$$

Z uvedené úlohy je patrné, že význam, vliv na výslednou hodnotu čísla, symbolu 3 je významně větší (desetinásobně), než vliv, význam symbolu 4. Stejný závěr platí pro libovolnou spolu sousedící dvojici symbolů (číslic) ve zkoumaném čísle.

Zobecněním úlohy pro libovolný číselný základ získáme:

Zápis celého čísla v poziční číselné soustavě:

$$a = \operatorname{sgn} a \left(\sum_{k=n}^{0} a_k \cdot r^k \right)$$

případně v „rozepsané“ podobě:

$$a = \operatorname{sgn} a \cdot \left(a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + a_{n-2} \cdot r^{n-2} + \cdots + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 \cdot r^0 \right).$$

kde:

$\operatorname{sgn} a$	znaménko čísla a
r	základ číselné soustavy (radix)
a_k	$k-tý$ symbol čísla ($k-tá$ číslice)
n	počet číslic v čísle

Blok 3.2.1: Zápis celého čísla v poziční číselné soustavě

Takto formulovaný zápis čísla (viz 3.2.1) dovoluje zapsat pouze čísla celá.

Často je potřeba zapsat i čísla, která obsahují zlomkovou část, *racionální čísla*. Tuto část pak oddělujeme od celé části zvláštním symbolem, obvykle nazývaný *desetinná čárka*[2]. Rozšíření pro desetinná, obecně zlomková čísla lze formulovat (viz 3.2.2):

[1] může být ovlivněno lokálními kulturními odlišnostmi každého národa...

[2] opět jde o lokálně závislý symbol. „Anglosaské“ země používají symbol *desetinná tečka*

Zápis racionálního čísla v poziční číselné soustavě:

$$a = \operatorname{sgn} a \left(\sum_{k=n}^{-l} a_k \cdot r^k \right)$$

případně v „rozepsané“ podobě:

$$a = \operatorname{sgn} a \cdot \left(a_n \cdot r^n + a_{n-1} \cdot + \cdots + a_1 \cdot r^1 + a_0 \cdot r^0 + a_{(-1)} \cdot r^{-1} + \cdots + a_{(-l)} \cdot r^{-l} \right).$$

kde:

$\operatorname{sgn} a$	znaménko čísla a
r	základ číselné soustavy (radix)
a_k	$k-tý$ symbol čísla
n	počet číslic celé části čísla
l	počet zlomkové části čísla, obvykle desetinná část

Blok 3.2.2: Zápis racionálního čísla v poziční číselné soustavě

3.2.1 Dekadická číselná soustava

Dekadická soustava (desítková soustava) je poziční číselná soustava se základem 10. Pro zápis čísel používá symboly $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.

Tato číselná soustava je dnes nejpoužívanější jak v běžném životě, tak ve vědeckých a technických oborech.

Zápis čísla v dekadické soustavě:

$$a = (a_2 a_1 a_0, a_{-1} a_{-2})_{10}$$

případně:

$$a = \operatorname{sgn} a \left(\sum_{k=2}^{-2} a_k \cdot 10^k \right)$$

kde:

$\operatorname{sgn} a$	znaménko čísla a
10	základ číselné soustavy (radix)
a_k	$k-tý$ symbol čísla

Je-li $r = 10$ (dekadická číselná soustava), pak číselný základ neuvádí, tj zápis:

$$(a_2 a_1 a_0, a_{-1} a_{-2})_{10} = a_2 a_1 a_0, a_{-1} a_{-2}$$

je ekvivalentní.

Blok 3.2.3: Zápis racionálního čísla v dekadické číselné soustavě

Úloha:
Zjistěte význam (váhu) jednotlivých číslic v čísle $1234,56$ zapsané v desítkové soustavě

$$\begin{aligned} 1234,56 &= 1\cdot 1000+2\cdot 100+3\cdot 10+4\cdot 1+5\cdot 0,1+6\cdot 0,01 \\ &= 1\cdot 10^3+2\cdot 10^2+3\cdot 10^1+4\cdot 10^0+5\cdot 10^{-1}+6\cdot 10^{-2} \\ &= 1\cdot 10^3+2\cdot 10^2+3\cdot 10^1+4\cdot 10^0+5\cdot \tfrac{1}{10^1}+6\cdot \tfrac{1}{10^2} \end{aligned}$$

Ve vědeckém světě (v běžném životě také) vznikla potřeba „pojmenovat“ některé (nejčastěji používané) násobky čísla 10. Postupně se vyvinul a posléze standardizoval systém označení těchto násobků. Bohužel, ne ve všech částech světa stejně.

Postupem doby se ustálily dva používané systémy:

- tzv. *krátká soustava* (z francouzského échelle courte) označuje slovem *bilion* číslo, které se rovná tisíci milionům (10^9) a další pojmenování následují vždy po tisícinásobku. Tato soustava nezná slovo *miliarda*. Krátká soustava je užívána ve Spojených státech a zhruba od sedmdesátých let 20. století také ve většině anglicky mluvících zemí (Velká Británie, Austrálie, Kanada s výjimkou frankofonních částí, Irsko ap.)

- v kontinentální Evropě, tedy i v České republice je zpravidla užívána tzv. dlouhá soustava (échelle longue), v níž pro tisíc milionů (10^9) je užíván termín *miliarda*, *bilion* má význam *milion milionů* (10^{12}).

Existují státy, které kombinují oba používané systémy označení (bývalý Sovětský svaz, Turecko, ...).

Názvy zvolených násobků byly standardizovány a označují se termínem *předpony dle soustavy SI*:

Hodnota	*Název v krátké soustavě*	*Název v dlouhé soustavě*	*Předpona*
10^{-24}	septiliontina	kvadriliontina	yokto
10^{-21}	sextiliontina	triliardtina	zepto
10^{-18}	quintiliontina	triliontina	atto
10^{-15}	quadriliardtina	biliardtina	femto
10^{-12}	triliontina	biliontina	piko
10^{-9}	biliontina	miliardtina	nano
10^{-6}	miliontina	miliontina	mikro
10^{-3}	tisícina	tisícina	mili
10^{-2}	setina	setina	centi
10^{-1}	desetina	desetina	deci
10^{0}	jednotka	jednotka	-
10^{1}	desítka	desítka	deka
10^{2}	sto	sto	hekto
10^{3}	tisíc	tisíc	kilo
10^{6}	milion	milion	mega
10^{9}	bilion	miliarda	giga
10^{12}	trilion	bilion	tera
10^{15}	quadrilion	biliarda	peta

Názvy předpon v soustavě SI→

Hodnota	*Název v krátké soustavě*	*Název v dlouhé soustavě*	*Předpona*
10^{18}	quintilion	trilion	exa
10^{21}	sextillion	triliarda	zetta
10^{24}	septillion	kvadrilion	yotta
10^{27}	octillion	kvadriliarda	-
10^{30}	nontillion	kvintilion	-
10^{33}	decillion	kvintiliarda	-
$10^{3}6$	undecillion	sextilion	-
10^{39}	bidecillion	sextiliarda	-
10^{42}	tredecillion	septilion	-
10^{45}	quadrodecillion	septiliarda	-
10^{48}	quintodecillion	oktilion	-
10^{51}	sexdecillion	oktiliarda	-

Názvy předpon v soustavě SI

Tabulka 3.2.3: Názvy předpon v soustavě SI

3.2.2 Binární číselná soustava

Binární (dvojková) číselná soustava je poziční číselná soustava se základem 2. Pro zápis čísel používá pouze symboly $0, 1$.

Tato číselná soustava je dnes nejpoužívanější ve světě výpočetní, automatizační a regulační techniky.

Důvod je jednoduchý. Ve světě elektrotechniky lze poměrně snadno sestrojit elektrické obvody, které jsou ve stavu:

- zapnuto, žárovka svítí
- vypnuto, žárovka nesvítí

Tyto stavy je velmi jednoduché technickými prostředky detekovat, případně sestrojit a provozovat.

Stavy: žárovka svítí „trochu“, příp. „méně“, nebo „o něco více“ současná technika umí rozlišovat poměrně obtížně a nespolehlivě. Proto se nepoužívají[3].

Formalizací, abstrahováním od technických podrobností dojdeme k používání *dvojkové číselné soustavy*:

Zápis čísla v binární číselné soustavě:

$$(a_2 a_1 a_0, a_{-1} a_{-2})_2$$

případně:

$$a = \operatorname{sgn} a \left(\sum_{k=2}^{-2} a_k \cdot 2^k \right)$$

[3]existují výjimky. V bývalém Sovětském svazu se pokoušeli sestrojit *ternární počítač*, tj. počítač pracující v trojkové číselné soustavě.

znamená:

$\operatorname{sgn} a$	znaménko čísla a
2	základ číselné soustavy (radix)
a_k	$k-tý$ symbol čísla

Blok 3.2.4: Zápis racionálního čísla v binární soustavě

Úloha:
Zjistěte význam (váhu) jednotlivých číslic v čísle $(110101,01)_2$ zapsané v binární soustavě

$$\begin{aligned}(1101,01)_2 &= 1\cdot 8+1\cdot 4+0\cdot 2+1\cdot 1+0\cdot 0,5+1\cdot 0,25\\ &= 1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0+0\cdot 2^{-1}+1\cdot 2^{-2}\\ &= 1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0+0\cdot \tfrac{1}{2^1}+1\cdot \tfrac{1}{2^2}\\ &= (13,25)_{10}=13,25\end{aligned}$$

Explozivní rozvoj výpočetní techniky v posledních desetiletích, zejména pak její využití ve všech možných oborech lidské činnosti vedl k tomu, že si určité, hlavně technické obory (automatizace, telekomunikace, ...) vytvořily své vlastní názvosloví, případně specificky „implementovaly“ binární číselný systém.

Vznikly tak různé *druhy kódů*, které odrážely konkrétní podmínky použití v dané oblasti. Uveďme příklady některých z nich:

Grayův kód zrcadlový binární kód, známý podle jeho tvůrce, *Franka Graye*, je binární číselná soustava, ve které se každé dvě po sobě jdoucí hodnoty liší v bitovém vyjádření změnou pouze jedné bitové pozice. Zrcadlový binární kód byl původně navržen pro zabránění rušivého výstupu z elektromechanických přepínačů (hazardy relé).

V reálném systému není nikdy možné zaručit, aby se změnilo více logických hodnot naprosto současně a není možno zajistit ani jejich naprosto současné přečtení a vyhodnocení. Toto bývá u elektroniky způsobeno různým zpožděním logických členů, přechodovými charakteristikami, parazitními kapacitami, nesynchronním snímání optického kotouče snímači a dalšími vlivy.

Dnes je Grayův kód používán pro podporu opravy chyb v digitální komunikaci. Také ho používá digitální pozemní televize a některé systémy kabelové televize.

Brownův kód patří do skupiny kódů GA+F. Tyto kódy jsou nesystematické bezpečnostní kódy (nelze je rozdělit na informační a kontrolní část). Jde o libovolné číslo (popř. znak z řetězce převedený do ASCII kódu), které vynásobíme konstantou a další konstantu převedeme. Výsledek poté převedeme do binární podoby.

Kód 1 z n je binární kód, kde hodnotu čísla určuje pořadí hodnoty 1 v čísle. V praxi se využívá například při fyzickém zobrazení binárního čísla na zobrazovací jednotku.

Dekadicky	Binárně	Kód 1 z n
0	0000	000000000
1	0001	100000000
2	0010	010000000
3	0011	001000000
4	0100	000100000
5	0101	000010000
6	0110	000001000
7	0111	000000100
8	1000	000000010
9	1001	000000001

Tabulka 3.2.4: Kód 1 z n

Binary Coded Decimal (zkráceně BCD, dvojkově reprezentované dekadické číslo) je způsob kódování celých čísel. Ukládá pouze desítkové číslice $(0-9)$, a to na úrovni čtveřic bitů (nibblů) tím způsobem, že každý nibble odpovídá jedné desítkové číslici.

Vzhledem k tomu, že pro čtveřici bitů existuje šestnáct různých kombinací, a desítkových číslic je jen deset, je šest kombinací nevyužito. V porovnání s hexadecimální soustavou, kde je pro každé čtyři bity využíváno všech šestnáct hodnot (1010 až 1510 jako písmena AH až FH), je BCD kód z hlediska využití paměti neúsporný. BCD kód snižuje efektivitu využití paměti, realizuje právě opačnou myšlenku než *Huffmanovo kódování*.

Přirozená hodnota			BCD kód	
bin	hexa	dekadická	dekadická	zacyklení
0000	0	0	0	první hodnota, umělé zacyklení zpět na 9, umělý převod z vyššího řádu
0001	1	1	1	
0010	2	2	2	
0011	3	3	3	
0100	4	4	4	
0101	5	5	5	
0110	6	6	6	
0111	7	7	7	
1000	8	8	8	
1001	9	9	9	poslední hodnota, umělé zacyklení dál na 0, umělý přechod na vyšší řád
1010	A	10	neplatná hodnota, nevyužitá bitová kombinace	
1011	B	11	neplatná hodnota, nevyužitá bitová kombinace	
1100	C	12	neplatná hodnota, nevyužitá bitová kombinace	
1101	D	13	neplatná hodnota, nevyužitá bitová kombinace	
1110	E	14	neplatná hodnota, nevyužitá bitová kombinace	
1111	F	15	neplatná hodnota, nevyužitá bitová kombinace	

Tabulka 3.2.5: BCD kód

Svoji „největší slávu“ zažil BCD kód s nástupem jednoduchých kalkulaček v 70 a 80 letech minulého tisíciletí, kdy se jevilo výhodné manipulovat s jednotlivými ciframi.

S rozvojem mikroprocesorové techniky a stále masivnějším užíváním binárního kódu tento ustupuje poněkud do pozadí, byť všechny mikroprocesory všech generací obsahují instrukce pro BCD výpočty.

3.2.3 Oktalová číselná soustava

Oktalová (osmičková) číselná soustava je poziční číselná soustava se základem 8. Pro zápis čísel používá pouze symboly $0, 1, 2, 3, 4, 5, 6, 7$.

Tato číselná soustava bývala ve své době (50 a 60 léta minulého století) poměrně hojně používána. Důvodů bylo několik:

- každý výrobce produkoval výpočetní systém s odlišnou bitovou šíří, typicky v rozsahu 6 až 23 bitů
- systémy byly navzájem absolutně nekompatibilní
- chybí standardy pro tuto oblast. Ty přináší až norma IEEE-754

Absence standardů pro zápis dat v oktalové soustavě vedl k tomu, že si každý výrobce „zvolil“ svůj způsob reprezentace dat v oktalové soustavě. Což často vedlo ke zmatkům, jelikož tento zápis nebyl jednoznačný.

Záleželo totiž na tom, z které strany jste začali „odpočítávat“ jednotlivé trojce pro oktalový zápis:

$$(10100111011) \rightarrow (101)(001)(110)(11) = (5163)_8$$

ve srovnání:

$$(10100111011) \rightarrow (10)(100)(111)(011) = (2473)_8$$

Zápis čísla v oktalové číselné soustavě:

$$(a_2 a_1 a_0, a_{-1} a_{-2})_8$$

případně:

$$a = \operatorname{sgn} a \left(\sum_{k=2}^{-2} a_k \cdot 8^k \right)$$

znamená:

$\operatorname{sgn} a$	znaménko čísla a
8	základ číselné soustavy (radix)
a_k	$k-tý$ symbol čísla

Blok 3.2.5: Zápis racionálního čísla v oktalové číselné soustavě

Úkol:
Zjistěte význam (váhu) jednotlivých číslic v čísle $(263,42)_8$ zapsané v oktalové soustavě

$$\begin{aligned}(263,42)_8 &= 2\cdot 64+6\cdot 8+3\cdot 1+4\cdot 0,125+2\cdot 0,015625\\ &= 2\cdot 8^2+6\cdot 8^1+3\cdot 8^0+4\cdot 8^{-1}+2\cdot 8^{-2}\\ &= 2\cdot 8^2+6\cdot 8^1+3\cdot 8^0+4\cdot \tfrac{1}{8^1}+2\cdot \tfrac{1}{8^2}\\ &= (179,53125)_{10} = 179,53125\end{aligned}$$

Oktalové vyjádření se používá i v současnosti, zejména tam, kde to přirozený charakter informace podporuje. Typický příklad je zápis přístupových práv k souboru v operačních systémech Unixového typu, kde tvoří významově ucelené trojice:

`owner` : `read`, `write`, `execute`,
`group` : `read`, `write`, `execute`,
`other` : `read`, `write`, `execute`,

3.2.4 Hexadecimální číselná soustava

Hexadecimální (šestnáctková) číselná soustava je poziční číselná soustava se základem 16. Protože „přirozená“ číselná soustava má pouze 10 symbolů (číslic), vzniká problém, jakými symboly označit zbývajících 6 hodnot.

Místo tvorby speciálních „obrázků“ se přistoupilo k použití prvních 6 písmen latinské alfabety. Proto se pro zápis čísel používají symboly $0,1,2,3,4,5,6,7$ a k nim se přidaly symboly A,B,C,D,E,F

Dekadicky	*Binárně*	*Oktalově*	*Hexadecimálně*
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Dekadické, binární, oktalové a hexadecimální hodnoty

Tabulka 3.2.6: Dekadické, binární, oktalové a hexadecimální hodnoty

Tato číselná soustava je v současnosti velmi populární a je masivně používána. Důvodů je opět několik:

- došlo de facto k ustálení pojmu „bajt“ a jeho bitové délky na počet 8bitů.
- systémy jsou mnohem více kompatibilní
- vytvořen chybějící standard. Norma IEEE-754
- hexadecimální zápis je velmi úsporný při zápisu i dlouhého binárního čísla:

$$(11000011)_2 = (B3)_H$$

Pro zápis hodnoty jednoho bytu tak stačí dva hexadecimální symboly, což se jeví jako „stále větší“ výhoda v souvislosti s rychlým vývojem výpočetní techniky a jejího (skoro) stálého zvětšování (datové) bitové šíře aktuálních mikroprocesorů.

Úkol:
Zjistěte význam (váhu) jednotlivých číslic v čísle $(A3D,C)_{16}$ zapsané v hexadecimální soustavě

$$\begin{aligned}(A3D,C)_{16} &= 10\cdot 256+3\cdot 16+13\cdot 1+12\cdot 0{,}0625\\ &= 10\cdot 16^2+3\cdot 16^1+13\cdot 16^0+12\cdot 16^{-1}\\ &= 10\cdot 16^2+3\cdot 16^1+13\cdot 16^0+12\cdot \tfrac{1}{16^1}\\ &= (2621,75)_{10}=2621,75\end{aligned}$$

3.2.5 Mayská číselná soustava

Mayská číselná soustava je poziční číselná soustava se základem 20. Zmiňujeme se o ní v souvislosti s davovým šílenstvím, které propuklo v některých částech věta ke konci roku 2012, kdy různí samozvaní chiliasté hlásali brzký zánik světa, aniž k tomu měli nějaká fakta.

Mayská číselná soustava patří do skupiny starověkých číselných systémů a v současnosti se nepoužívá.

Mayové ve své dvacítkové početní soustavě používali poziční zápis, kdy číslice jednotlivých řádů psali nad sebe. Oni nebo jejich předchůdci Olmékové nezávisle na sobě objevili důležitý pojem nuly, což bylo někdy kolem roku 357.

Evropané neznali nulu až do 12. století, kdy ji převzali od Arabů spolu s číselným systémem nazývaným od té doby *arabské číslice*. Mayské nápisy ukazují, že „uměli“ pracovat se čísly většími než stovky milionů.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19

Tabulka 3.2.7: Mayská číselná soustava

Číslice byly tvořeny ze tří znaků:
-  : nula (lastura)
- · : jednička (tečka)
- — : pětka (čára)

Například číslo 19 se napíše jako čtyři tečky v horizontální řadě nahoře a k tomu tři čáry rovnoběžně pod sebou:

$$19 = \overset{\cdot\cdot\cdot\cdot}{\equiv}$$

Dvě tečky pod sebou se dají přečíst jako 1 x 20 + 1 = 21:

$$21 = \begin{matrix}\cdot\\ \cdot\end{matrix}$$

Čísla se psala zespodu nahoru.

Jelikož Mayové neměli výraz pro číslo 20 psali číslo např. 421 jako 3 tečky pod sebou. Nejprve napsali:

- počet jedniček(20^0)
- nad něj dvacítek(20^1)
- nad něj počet čtyřstovek(20^2)
- nad něj počet osmitisícovek(20^3)
- atd.

Mayové měli ve své době poměrně několik složitých a vyspělých kalendářů, založený na velmi přesném astronomickém pozorování. Obsahovaly několik do sebe vnořených cyklů, kdy uplynutím vnitřního cyklu se „posunul“ o krok i vnější cyklus.

3.2.6 Převody číselných soustav

Často je potřeba převést číslo z jedné číselné soustavy do jiné. V souvislosti s intenzivním využíváním výpočetní techniky je potřeba mimořádně užitečná.

Pro převody čísel z jedné číselné soustavy do druhé se používá asi nejvíce používaná metoda *postupného dělení se zbytkem* , viz vlastnosti celých čísel uvedené v tématickém bloku 1.1.3 v odstavci 1.1.2. Tj. v každém kroku řešíme rovnici:

$$a = p \cdot b + r,$$

kde:
- a je celé číslo v dané číselné soustavě (například v desítkové),
- b je celé číslo v hledané (jiné) číselné soustavě (například dvojkové)
- p je hledaný částečný podíl,
- r je hledaný zbytek po dělení splňující podmínku: $0 \leq r \leq |b|$.

Ukážeme proceduru na příkladu převodu čísla $61 = (61_{10})$ z desítkové soustavy do dvojkové soustavy, tj. $b = 2$:

Převod čísla $61 = (61)_{10}$ z desítkové soustavy do dvojkové soustavy.

$$\begin{array}{lllllll} p_0 & = & 61 & = & p_1 \cdot 2 + r_0 & = & 30 \cdot 2 + 1, \\ p_1 & = & 30 & = & p_2 \cdot 2 + r_1 & = & 15 \cdot 2 + 0, \\ p_2 & = & 15 & = & p_3 \cdot 2 + r_2 & = & 7 \cdot 2 + 1, \\ p_3 & = & 7 & = & p_4 \cdot 2 + r_3 & = & 3 \cdot 2 + 1, \\ p_4 & = & 3 & = & p_5 \cdot 2 + r_4 & = & 1 \cdot 2 + 1, \\ p_5 & = & 1 & = & p_6 \cdot 2 + r_5 & = & 0 \cdot 2 + 1, \end{array}$$

tj. $p_k = 2 \cdot p_{k+1} + r_k\,; \quad 0 \leq r_k < 2.$

Takže $(61)_{10} = r_5 \cdot 2^5 + r_4 \cdot 2^4 + r_3 \cdot 2^3 + r_2 \cdot 2^2 + r_1 \cdot 2^1 + r_0 \cdot 2^0 =$
$= 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$
$(61)_{10} = (111101)_2$
$= 2(2(2(2(r_2 \cdot 2 + r_4) + r_3) + r_2) + r_1) + r_0.$

Příklad 3.2.1: Převod čísla z desítkové soustavy do dvojkové

Využitím principu *Hornerova algoritmu* dostaneme:

Vstup: n, a, b (v příkladu 3.2.1 $n = 5$, $a = 61$, $b = 2$).
$n =$ celá část čísla $\dfrac{\ln a - \ln r_n}{\ln b}$
(řešení nerovnice $q \leq r_n \cdot b^n$, první člen rozvoje o základu b)
$p_0 = a$
pro $k = 0, 1, 2, \ldots, n$:
p_{k+1}: řešení rovnice $p_k = b \cdot p_{k+1}$ splňující podmínku $0 \leq r_k < |b|$.
$r_k = p_k - b \cdot p_{k+1}$.
Výstup: $(r_n, r_{n-1}, t_{n-2}, \ldots, r_1, r_0)$ vyjádření čísla a v *b-adickém* systému.

Poznámka: Pro označení celé části čísla se používá symbol $\lfloor$ a $\rfloor$. Takže $\lfloor 2,3 \rfloor = 2$.

Vstup: $a = 61, b = 8$
$n = \left\lfloor \dfrac{\ln 61 - \ln r_n}{\ln 8} \right\rfloor = \lfloor 1,015932\ldots \rfloor = 1$
$p_0 = 61$
pro $k = 0, 1$:

$$\begin{array}{lllllll} p_0 & = & 61 & = & p_1 \cdot 8 + r_0 & = & 7 \cdot 8 + 5, \\ p_1 & = & 7 & = & p_2 \cdot 8 + r_1 & = & 0 \cdot 8 + 7, \end{array}$$

Výstup: $(61)_{10} = r_1 \cdot 8^1 + r_0 \cdot 8^0 =$
$= 7 \cdot 8^1 + 5 \cdot 8^0 =$
$= (75)_8$

Příklad 3.2.2: Převod čísla z desítkové soustavy do osmičkové

Pro ruční použití je nejčastěji výchozí (zdrojová) číselná soustava desítková. Cílová číselná soustava pak je libovolná.

Pak je postup výpočtu následující:

1. původní číslo vydělíme číslem, radixem cílové číselné soustavy
2. zjistíme a zapíšeme zbytek po dělení
3. vzniklý podíl opět vydělíme radixem cílové číselné soustavy
4. dále pokračujeme bodem 2 dokud je vzniklý podíl větší než nula
5. postupně získané zbytky po dělení zapíšeme v opačném pořadí, než byly získány výpočtem. Výsledek je převedené číslo.

V praxi se často vyskytuje potřeba převádět čísla mezi soustavami:

- binární
- oktalová
- hexadecimální

Věnujme nyní několik zamyšlení zmiňovaným číselným soustavám. Lze si všimnout, že *radix* uvažovaných soustav je *mocnina čísla* 2. Jinými slovy, počet možných hodnot číselné soustavy s nižším radixem je *celistvý násobek* možných hodnot číselné soustavy s vyšším radixem.

Z tohoto prostého faktu plynou některé zajímavé důsledky pro praktické použití. Vraťme se znovu k získaným výsledkům v uvedených příkladech (úlohách):

$$(61)_{10} = (111101)_2 = (75)_8$$

Rozdělme nyní šestici binárních cifer na skupiny po třech cifrách, trojice „odpočítáváme“ od nejnižších řádů:

$$(111101)_2 :\rightarrow\ (111\ \ 101)_2$$

Je patrné, že každá trojice v binární reprezentaci *přesně* odpovídá právě jedné cifře v oktalové reprezentaci!

Rozdělme nyní šestici binárních cifer na skupiny po čtyřech cifrách, čtveřice zase „odpočítáváme“ od nejnižších řádů:

$$(111101)_2 :\rightarrow\ (11\ \ 1101)_2$$

Pohledem do tabulky 3.2.4 zjistíme že každá čtveřice *přesně* odpovídá právě jedné hexadecimální cifře:

$$(111101)_2 :\rightarrow\ (11\ \ 1101)_2 = (3D)_{16}$$

Kontrolní výpočet potvrdí, že převod čísla je správně.

Této vlastnosti lze velmi efektivně využívat, vznikne-li potřeba převádět čísla (například) z šestnáctkové soustavy do osmičkové a naopak. Ušetříme si tím pracné a zdlouhavé dělení.

Je tedy zřejmé, že kdybychom „náhle“ potřebovali převést dané číslo do dvaatřicítkové číselné soustavy, prostě rozdělíme (původní) binární číslo na pětice a najdeme odpovídající symboly.

3.3 Unární číselná soustava

Jedničková soustava (unární soustava), anglicky *unary numeral system* je v matematice označení bijektivní poziční číselné soustavy se základem (radixem) rovným jedné ($r = 1$). Je to nejjednodušší možná číselná soustava a často je pokládána za historický základ všech ostatních číselných soustav.

Má poněkud nejednoznačné postavení v klasifikaci číselných soustav. Je považována za speciální poziční soustavu, může být řazena mimo poziční soustavy. Nelze ji ale zařadit mezi nepoziční číselné soustavy.

Čísla se zapisují jediným znakem ve významu jedna, zpravidla se používá rovná čára, nejčastěji svislá. Pro účely strojního zpracování se pak používá symbol „1“. Nemá jako jediná poziční soustava znak „0“ (nula).

Obrázek 3.3.1: Příklad unární číselné soustavy

Větší počty (než jedna) a čísla se zapisují opakováním tohoto znaku tolikrát, až je dosaženo potřebného počtu, například:

$$\begin{aligned} 4 &= 1111 \\ 10 &= 1111111111 \end{aligned}$$

Jde o aditivní číselnou soustavu, protože 4, zapsané jako = „1111“ plyne z faktu:

$$1+1+1+1=4.$$

Vykazuje některé znaky pozičního systému (se základem 1), například číslo 4 zapsané jako „1111“ odpovídá pozičnímu zápisu:

$$1\cdot 1^3+1\cdot 1^2+1\cdot 1^1+1\cdot 1^0=1+1+1+1=4.$$

Jiné vlastnosti standardních pozičních systémů však tato soustava nemá. Pozice jejích „řádů“ jsou totiž zaměnitelné a tak pozice desetinné čárky nemá smysl. Tudíž nelze pomocí této soustavy zapisovat zlomková (necelá) čísla.

Aritmetické výpočty probíhají poněkud zvláštně. spočívají v prostém *spojování*, případně *rozdělování* pásu čísel, například operace sčítání:

$$11+111111=11111111 \quad (2+6=8)$$

Operace odečítání spočívá v prostém, porovnání délek pásů čísel. Výsledek je to co zůstane „je navíc“:

$$\begin{array}{rr} 111111 & 6 \\ -111 & -3 \\ \hline 111 & 3 \end{array}$$

Ač se to nezdá, unární číselná soustava je široce a intenzivně používána i v každodenním životě. Například prsty na ruce sloužící jako početní pomůcka při nácviku sčítání. Nebo zápis počtu spotřebovaných, zkonzumovaných nápojů při návštěvě restauračního zařízení. Nebo ...

Kapitola 4
Množina počítačových čísel

Obsah kapitoly

V současnosti, s rozvojem a využitím výpočetní techniky ve všech oborech lidského konání je kriticky důležité rozumět základním principům, funkcím a vlastnostem elektronického počítače. Samozřejmě, že ona „ kritická důležitost“ je myšlena ve vztahu ke specialistům, odborníkům, kteří „mají v popisu práce“ vytvářet programové systémy pro své uživatele, případně tyto systémy opravovat, konfigurovat, či „jen“ efektivně používat.

Jednou z fundamentálních znalostí v tomto smyslu je i znalost o způsobu uchování, *reprezentaci* čísel v paměti počítače včetně způsobů jak s těmito čísly vhodně manipulovat. Prostě, jak něco „dostatečně přesně“ spočítat.

4.1 Zobrazení čísel v počítači

Každý počítač provádějící vědeckotechnické výpočty může pracovat (pouze) s čísly, která se dají vyjádřit v *semilogaritmickém tvaru s normalizovanou mantisou*.

$$\alpha = \operatorname{sgn}\alpha \left(\frac{\alpha_1}{q^1} + \frac{\alpha_2}{q^2} + \cdots + \frac{\alpha_t}{g^t} \right) q^b , \quad \alpha_1 \neq 0$$

kde

$q > 1$	je *základ* číselné soustavy
$\alpha_i = \{1,2,3,\ldots,t\}$	číslice *mantisy*
b	je exponent

Blok 4.1.1: Semilogaritmický tvar čísla

Například pro číslo $(193754,2015)_{10}$ je:

$1,937542015 \cdot 10^5$	obyčejný semilogaritmický tvar;
$0,1937542015 \cdot 10^6$	semilogaritmický tvar s normalizovanou mantisou.

Pro číslo $(11010001,101)_2 = 1\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 0\cdot 2^{-3}$

máme $0,11010001101 \cdot 2^8$.

Poznámka: V jiných číselných soustavách se místo „desetinná čárka“ užívá termín oddělovač, fixní čárka, pevná čárka, plovoucí (pohyblivá) čárka, řádová čárka.

Příklad 4.1.1: Semilogaritmický tvar čísla

Počet číslic mantisy je určen přirozeným číslem t. To je dáno konstrukcí počítače, stejně jako přirozené číslo q. Konstrukcí počítače jsou rovněž dány hranice m_1 a m_2 celého čísla b.

Množina čísel α není nekonečná. Má přesně

$$2(q-1)q^{t-1}(m_2 - m_1 + 1) + 1 \text{ prvků}$$

a označujeme ji symbolem:

$$M(q,t,m_1,m_2), \quad \text{nebo zkráceně} \quad M(q,t).$$

Počítači, výpočetnímu systému, který pracuje s čísly z množiny $M(q,t)$, někdy říkáme *q-adický t-místný počítač.*

Poznamenejme jen, že popsaný systém zobrazení čísel v počítači se nazývá *systém s pohyblivou řádovou čárkou* a od roku cca 1985 je normalizován normou *IEEE 754*. Podrobnosti jsou uvedeny v odstavci (4.3).

Zobrazení čísla $\alpha = 13,25$ do množiny $M(q,t)$ bez vstupní chyby:

$M(10,4)$	neboť	$\alpha = 0,1325 \cdot 10^2$
$M(2,6)$	neboť	$\alpha = 0,110101 \cdot 2^4$
		$(13,25 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2})$
$M(8,3)$	neboť	$\alpha = 0,152 \cdot 8^2$
		$(13,25 = 1\cdot 8^1 + 5\cdot 8^0 + 2\cdot 8^{-1})$
$M(16,2)$	neboť	$\alpha = 0,D4 \cdot 16^1$
		$(13,25 = (\frac{13}{16^1} + \frac{4}{16^2}) \cdot 16^1)$

Příklad 4.1.2: Zobrazení čísla do $M(q,t)$

Poznámka: termínem „vstupní chyba“ je myšleno:

- přesnost zobrazení vstupního čísla v množině $M(q,t)$

Potom formulace „bez vstupní chyby“ znamená zobrazení všech cifer daného čísla v množině $M(q,t)$.

Pro některé druhy výpočtů je výhodnější pracovat s čísly v tzv. *pevné řádové čárce*. Pracuje s pevným počtem desetinných míst. Typický představitel tohoto typu zobrazení čísel v počítači je datový typ *Integer*[1].

4.2 Zaokrouhlování

Při ukládání vstupních dat do paměti počítače musí počítač reálnému číslu $x \in \mathbb{R}$ přiřadit číslo $\alpha \in M(q,t)$. To lze provést dvěma způsoby, *řezáním*, nebo *zaokrouhlováním*.

Je-li:

$$x = aq^b = \operatorname{sgn} x \left(\sum_{k=1}^{\infty} x_k q^{-k} \right) q^b \in \mathbb{R},$$

potom obraz čísla $x \in \mathbb{R}$ označíme

$$\alpha = \tilde{a}q^b = \operatorname{sgn} \alpha \left(\sum_{k=1}^{t} \alpha_k q^{-k} \right) q^b \in M,$$

Operace *řezání* pak znamená, že klademe:

$$\alpha_k = x_k \quad \text{pro} \quad k = 1, 2, \ldots, t$$

Operace *zaokrouhlování* pak znamená, ze klademe:

$$\alpha_k = x_k \quad \text{pro} \quad k = 1, 2, \ldots, t-1$$

$$\alpha_i = \begin{cases} x_i + 1 & \text{pro} \quad x_{i+1} \geq \frac{q}{2} \\ x_1 & \text{pro} \quad x_i < \frac{q}{2} \end{cases}$$

Operace *zaokrouhlování* čísla $\alpha \in M(q,t)$ čísla $x \in \mathbb{R}$ je obecně definována podmínkou:

$$|x - \alpha| = \min |x - \beta|$$

Operace *řezání* čísla $\alpha \in M(q,t)$ čísla $x \in \mathbb{R}$ je obecně definována podmínkou:

$$\alpha = \begin{cases} \max \beta \in M \mid \beta \leq x & \text{pro} \quad x > 0 \\ \min \beta \in M \mid \beta \geq x & \text{pro} \quad x < 0 \end{cases}$$

Označíme-li $\gamma : \mathbb{R} \to M$ takto definované zobrazení množiny reálných čísel do množiny M, tj. $\alpha = \gamma(x)$. Protože $q \geq q^{-1}$, platí pro relativní chybu odhad:

$$\left| \frac{x - \gamma(x)}{x} \right| \leq kq^{1-t} \,,$$

[1] používaný v jazycích Pascal, C, Fortran, případně jejich „odvozeninách“...

kde $k = 1$ při řezání a $k = \frac{1}{2}$ při zaokrouhlování, neboť:

$$\left|\frac{x-a}{x}\right| = \frac{q^b(a-\widetilde{a})}{b^b a} \quad \text{a} \quad |a-\widetilde{a}| \leq kq^{-1}$$

Číslu kq^{1-t} se také často říká *strojová přesnost*, případně *strojové epsilon*.

1. ***operace řezání*** $(k = 1)$
   V množině $M(10,5)$; bude číslo $\alpha = \gamma(x) = 0,31415 \cdot 10^1$, neboť
   $|0,3141592 - 0,31415| = 9,26 \cdot 10^{-6} \leq 1 \cdot 10^{-5}$.

2. ***operace zaokrouhlování*** $(k = \frac{1}{2})$
   V množině $M(10,5)$; bude číslo $\alpha = \gamma(x) = 0,3146 \cdot 10^1$, neboť
   $|0,3141592 - 0,31416| = 0,8 \cdot 10^{-6} \leq 0,5 \cdot 10^{-5}$.

Příklad 4.2.3: Zobrazení čísla $x = 3,141592$ do $M(q,t)$

4.3 Norma IEEE-754

Rozvoj výpočetní techniky a její využití stále širší skupinou uživatelů vytvářel sílící tlak na standardizaci klíčových částí výpočetních systémů. Tato potřeba byla dále akcelerována nástupem mikroprocesorové techniky a následně vznikem „světa osobních počítačů“.

Jedním z velmi palčivých problémů té doby byla absolutní programová nekompatibilita provozovaných programových systémů. Přechod z jednoho systému na jiný (jiného výrobce) byl v podstatě nemožný.

Ukázalo se však, že nahradit programové systémy konkurenčními není ten nejhorší problém. Daleko horší byl fakt, že data, která byla pořízena v jednom systému v podstatě nelze přenést do jiného, konkurenčního. Bylo to způsobeno odlišným způsobem práce s uloženými, odlišnostmi používaných datových formátů, který se lišil výrobce od výrobce a způsobem práce s těmito daty.

Tak postupně „dozrály“ podmínky pro vznik standardu, který má za úkol řešit tento problém, „vznikla“ norma IEEE-754 popisující základní datové typy a způsoby práce s nimi v (mikro)procesorové jednotce počítače.

Norma IEEE-754 má dvě „větší“ revize. První revize, vlastně i první vydání normy je z roku 1985. Druhé vydání normy z roku 2008 pak „reaguje“ na explozivní rozvoj výpočetní techniky, zejména pak mikroprocesorové techniky a její výpočetní možnosti a schopnosti. Postupně pak výsledky normy IEEE-754 převzaly i další mezinárodní normy a standardy pracující v jichých oblastech informačních technologií[2].

[2] ISO/IEC/IEEE 60559:2011...

Hlavním „tahounem“ prací na normě byla firma Intel. Ta potřebovala pro svůj připravovanou „žhavou“ technologickou novinku[3], mikroprocesor 8086 a jeho koprocesor 8087 „standardizovat“ (čti vnutit „zbytku světa“ své řešení) metodiku práce s čísly v pohyblivé čárce.

Proto v druhé polovině 70 let zahájil Dr. John Palmer práce na „standardu“. Původní návrh Dr. Palmera předpokládal využití množiny $M(10,t)$. Protože však práce na přípravě koprocesoru 8087 byly již v poměrně pokročilé fázi, nebylo tak možné původní Palmerův návrh akceptovat.

S pomocí externího konzultanta Williama Kahana (univerzita Toronto) byl připraven návrh normy (draft) s využitím množiny $M(2,t)$. Což v důsledku vedlo k některým „podivným“ vlastnostem čísel v plovoucí čárce a tím i k protahování přijetí normy.

Firma Intel tak byla první která v roce 1980 implementovala normu IEEE-754 ve svých mikroprocesorových systémech, byť ještě nebyla oficiálně schválena. Zvolené řešení se však ukázalo jako poměrně efektivní a bylo široce a relativně rychle akceptováno i ostatními výrobci hardware i software. Tím se z návrhu stal standard „de-facto“.

Norma IEEE-754 definuje nejenom vlastní formát dat, ale i příslušné výpočetní operace. Toto plyne z vlastností množiny $M(q,t)$. Ta totiž není z hlediska aritmetických operací *uzavřená*, tj. výsledek nějaké operace s čísly v množině $M(q,t)$ nemusí být v $M(q,t)$. Například součin dvou čísel s $t-$*místnou mantisou* má obecně $2t$, případně $2t-1$ číslic.

4.4 Vybrané pojmy a vlastnosti normy IEEE-754

Paměť výpočetního systému je elektronická součástka, která je „schopna“ zapamatovat si určitou kombinaci elektrických signálů. Víme, že současné počítače pracují s využitím binární číselné soustavy. Jedním z důsledků je to, že paměť počítače si umí pamatovat *pouze* celá čísla přesně. Pro uložení ostatních čísel a manipulaci s nimi je třeba použít jisté „techniky“.

4.4.1 Normalizace čísel

Jak tedy uložit po paměti počítače takové číslo, které matematika zná pod označením *reálné číslo*? Rozbor vlastností převodu čísla z jedné číselné soustavy (desítkové) do druhé (dvojkové), viz odstavce 3.2.1, 3.2.2 případně 3.2.4 ukazuje, že přesný převod obecně není možný. Je možná pouze více, nebo méně přesná *aproximace* takového čísla do množiny $M(q,t)$, (v paměti počítače).

Ukažme si na jednoduchém příkladu možná úskalí, která jsou spojena s tímto úkolem. Připomeňme stručně možný tvar čísla $a = (a_3a_2a_1a_0a_{-1}a_{-2})$ v semilogaritmickém tvaru s normalizovanou mantisou:

[3] ve své době...

$$\begin{aligned}(a)_2 &= (a_{-1}\cdot 2^{-1} + a_{-2}\cdot 2^{-2} + a_{-3}\cdot 2^{-3} + a_{-4}\cdot 2^{-4} + a_{-5}\cdots 2^{-2})\cdot 2^3 \\ &= (a_{-1}\cdot \tfrac{1}{2^1} + a_{-2}\cdot \tfrac{1}{2^2} + a_{-3}\cdot \tfrac{1}{2^3} + a_{-4}\cdot \tfrac{1}{2^4} + a_{-5}\cdot \tfrac{1}{2^2})\cdot 2^3\end{aligned}$$

Pak $k-tý$ člen přejde na tvar:

$$p_k = \frac{1}{2}p_{k+1} + r_k$$

Pro ruční použití, kdy je výchozí číselná soustava desítková je postup výpočtu následující:

1. původní číslo vynásobíme radixem cílové číselné soustavy
   (dělení číslem $\frac{1}{2}$ $\rightarrow$ násobení číslem 2)
2. zjistíme a zapíšeme celou část čísla po násobení
3. odečteme celou část čísla od součinu
4. výsledek opět násobíme radixem cílové číselné soustavy
5. dále pokračujeme bodem 2 dokud je vzniklý součin větší než nula
6. postupně získané celé části součinů čísla zapíšeme ve stejném pořadí, než byly získány výpočtem. Výsledek je převedené číslo.

Máme za úkol uložit do paměti počítače číslo $(0,2)_{10} = (2\cdot 10^{-1})\cdot 10^0$. Číslo je semilogaritmickém normalizovaném tvaru.

Zobrazme si kontrolní tabulku několika hodnot exponentu, která použijeme ke kontrole našemu výpočtu:

exponent	*hodnota*
-1	0,500 000 000
-2	0,250 000 000
-3	0,125 000 000
-4	0,062 500 000
-5	0,031 250 000
-6	0,015 625 000
-7	0,007 812 500
-8	0,003 906 250
-9	0,001 953 125

Tabulka 4.4.1: Zlomkové hodnoty několika binárních čísel

Převod čísla $0,2$ do binární číselné soustavy:

1. převáděné číslo: $0,2$, cílový radix: 2
   (a) násobíme: $0,2\times 2 = 0,4$; $a_{-1} = 0$,
   (b) násobíme: $0,4\times 2 = 0,8$; $a_{-2} = 0$,
   (c) násobíme: $0,8\times 2 = 1,6$; $a_{-3} = 1$,
   (d) násobíme: $0,6\times 2 = 1,2$; $a_{-4} = 1$,

(e) násobíme: $0,2 \times 2 = 0,4$; $a_{-5} = 0$,

(f) násobíme: $0,4 \times 2 = 0,8$; $a_{-6} = 0$,

(g) násobíme: $0,8 \times 2 = 1,6$; $a_{-7} = 1$,

(h) násobíme: $0,6 \times 2 = 1,2$; $a_{-8} = 1$,

(i) násobíme: $0,2 \times 2 = 0,4$; $a_{-9} = 0$,

(j) ...

2. výpočet nemá konečný počet kroků

3. číslo $0,2$ má *nekonečný binární rozvoj*

Rychlou kontrolou (s pomocí tabulky 4.4.1) zjistíme, že pokud sečteme zlomkové části čísla pro nenulové koeficienty příslušných exponentů zjistíme, že se budeme s libovolnou přesností přibližovat přesné hodnotě čísla $0,2$, ale nikdy ji nedosáhneme!

Jinými slovy, pro přesnou reprezentaci čísla $0,2$ v paměti počítače potřebuje *nekonečný* počet bitů. Což by mohl být drobný technický problém.

Řešení tohoto problému je nasnadě. Použijeme, uložíme do paměti počítače jen tolik prvních bitů, kolik jich budeme ochotni v paměti počítače vyhradit pro uložení přibližné hodnoty čísla $0,2$. Zbytek zanedbáme.

Zobecněním uvedeného příkladu lze formulovat úlohu:

> Najděte, navrhněte takový způsob ukládání dat do paměti počítače, aby bylo možné uložit co možná největší rozsah hodnot s největší možnou přesností.

Příklad 4.4.4: Ukládání dat v paměti počítače

Ukazuje se, že čísla s nekonečným binárním rozvojem lze uložit v paměti počítače více různými způsoby. Například můžeme zvolit různé počty bitů pro uložení jednotlivých cifer daného čísla...

Konkrétní volba je vždy určitým kompromisem mezi zadanými požadavky a konkrétními možnostmi dostupné techniky.

4.4.2 Formát plovoucí čárky

Praktický postup je následující:

1. převedení čísla do *normalizovaného tvaru*

2. převod mantisy z desítkové číselné soustavy do dvojkové

3. převod exponentu z desítkové číselné soustavy do dvojkové

4. volba, rozhodnutí o:

   (a) počtu bitů vyhrazených pro uložení hodnoty *mantisy*

(b) počtu bitů vyhrazených pro uložení hodnoty *exponentu*

(c) počtu bitů vyhrazených pro uložení *znaménka*

5. číslo je uloženo ve formátu *plovoucí čárky* (floating point)

Příklady normalizace čísla:

$$-1256,123 \cdot 10^5 \rightarrow \underbrace{-0,1256123}_{\text{mantisa}} \cdot \underbrace{10^9}_{\text{exponent}}$$

$$113,58 \cdot 10^{-8} \rightarrow 0,11358 \cdot 10^{-5}$$

$$0,000123 \rightarrow 0,123 \cdot 10^{-3}$$

Obecný zápis:

$$a = \operatorname{sgn} a \left(\sum_{k=n}^{-l} a_k \cdot 10^k \right) \rightarrow \operatorname{sgn} a \left(\sum_{k=0}^{-l-n-1} a_k \cdot 10^k \right) \cdot 10^{n+1}$$

Příklad 4.4.5: Příklady normalizace čísla

Rozvržení paměti počítače obsahující číslo v plovoucí čárce lze vyjádřit:

Formát čísla v plovoucí čárce:

$$a = (-1)^s \cdot 2^{exp-bias} \cdot m$$

kde:

s	samostatný znaménkový bit uloženého čísla
exp	počet bitů exponentu, je vždy kladné celé číslo
$bias$	posunutí exponentu, celé číslo
m	mantisa, zapisována jako *zlomková část čísla*

$$\underbrace{s \ \underbrace{eee\cdots eee}_{\text{bity exponentu}} \overbrace{mmm\cdots mmm}^{\text{bity mantisy}}}_{\text{celkový počet bitů čísla}}$$

Blok 4.4.2: Formát čísla v plovoucí čárce nad množinou $M(2,t)$

Pozorný čtenář nejspíše zjistil, že v popisu formátu plovoucí čárky (4.4.2) *chybí* určení znaménka exponentu. Absenci, respektive jeho nepotřebnost řeší právě parametr *bias*. Jeho hodnota je stanovena:

$$bias \approx \frac{\text{ExpMax} - \text{ExpMin}}{2}$$

kde: $ExpMax$: maximální hodnota exponentu
$ExpMin$: minimální hodnota exponentu

pak hodnota uloženého exponentu je:

$$exp = (n+1) + bias$$

kde $n+1$ je exponent normalizovaného čísla.

Tím se zajistí, že uložená hodnota exponentu *exp* bude *vždy kladná* a tak není nutné vyhradit zvláštní bit v paměti pro znaménko uloženého exponentu.

4.4.3 Implementace formátu plovoucí čárky

V tabulce 4.4.2 jsou uvedeny velikosti mantisy a exponentu pro různé celkové počty bitů vyhrazené pro uložení čísel v množině $M(2,t)$, které mohou být (jsou) implementovány v současných mikroprocesorových systémech.

Název	**bitů**	**exponent**	**mantisa**	**cifer** (t)	**poznámka**
binary16	16	5	10+1	3 až 4	poloviční přesnost
binary32	32	8	23+1	7 až 8	základní přesnost
binary64	64	11	52+1	15 až 16	dvojitá přesnost
extended	80	15	64+1		norma IEEE 754-1985
binary128	128	15	112+1	33 až 34	čtyřnásobná přesnost

Tabulka 4.4.2: Formáty definované nad množinou $M(2,t)$ dle normy IEEE-754

V tabulce 4.4.3 jsou pak uvedeny velikosti mantisy a exponentu pro různé celkové počty bitů vyhrazené pro ulžení čísle v množině $M(10,t)$. Tyto jsou součástí druhé revize normy IEEE-754 z roku 2008.

Název	**bitů**	**exponent**	**cifer** (t)	**poznámka**
decimal32	32	-95 až +96	7 až 8	základní přesnost
decimal64	64	-383 až +384	16	dvojitá přesnost
decimal128	128	-6 143 až +6 144	34	čtyřnásobná přesnost

Tabulka 4.4.3: Formáty definované nad množinou $M(10,t)$ dle normy IEEE-754

Poznámka: formáty nad množinou $M(10,t)$ se zatím v praxi příliš nepoužívají.

Díky zvolené interní struktuře čísla v plovoucí čárce, viz formulace (4.4.2), jsou možné některé, poněkud „podivné“ hodnoty, které lze do dané struktury uložit. Například (viz tabulka 4.4.4):

4.5 Aritmetické operace v množině M(q,t)

Jak již bylo konstatováno, množina $M(q,t)$ není z hlediska aritmetických operací *uzavřená*, tzn. výsledek nějaké aritmetické operace s čísly z $M(q,t)$ nemusí být v $M(q,t)$.

název	**popis, hodnoty**
„kladná“ nula	$s = 0$, $exp = 0$, $m = 0$
„záporná“ nula	$s = 1$, $exp = 0$, $m = 0$
„kladné“ nekonečno	$s = 0$, $exp = 11\cdots1$, $m = 0$
	(výsledek je příliš velký)
„záporné“ nekonečno	$s = 1$, $exp = \max$, $m = 0$
	(výsledek je příliš malý)
„NaN“, Not a Number	$exp = \max$, $m > 0$
	(výsledek operace typu $\frac{0}{0}$, nebo 0^0)

Tabulka 4.4.4: Některé zvláštní hodnoty čísel v plovoucí čárce

Například operace násobení dvou čísel s $t-místnou$ mantisou má obecně $2t$, případně $2t-1$ cifer.

Víme také, viz odst. 4.3, že norma IEEE-754 definuje mimo vlastní formát uložení i související výpočetní operace, tak, aby byly „dosažitelné“ matematické operace sčítání, odčítání, násobení a dělení.

Připomeňme jen, že operaci sčítání (odčítání) a násobení (dělení) prování *Aritmeticko-logická jednotka* (ALU) v mikroprocesoru, případně ji provádí specializovaná HW podpůrná součástka *Floating-point unit* (FPU), též nazývaná matematický koprocesor.

4.5.1 Operace sčítání a odčítání

Pro ilustraci předpokládejme operace definované normou IEEE-754, viz odstavec (4.3). Zvolíme operace sčítání (odčítání) dvou čísel v plovoucí čárce.

Operace sčítání probíhá tak příslušná výpočetní jednotka (ALU, FPU) postupuje dle kroků:

- zjistí, zda některý z operandů neobsahuje „podivné hodnoty“, viz tabulka 4.4.4
- pokud ne, pak porovná exponenty obou operandů
- nalezne větší z operandů porovnáním hodnot exponentů
- mantisa operandu s *menším* exponentem se posune o tolik bitů doleva, aby se hodnoty exponentů shodovaly (proces *denormalizace*).
- sečtou se (odečtou) mantisy operandů, dle hodnoty znaménkových bitů
- je-li výsledek součtu mantis > 1 provede se:
  - výsledek mantisy se posune o jedno místo doprava
  - hodnota exponentu se zvýší o 1
- doplní se výsledné znaménko
- výsledek se zaokrouhlí dle jednoho ze 4 určených postupů (rounding modes)

- kontrola, zda nedošlo během výpočtu k *přetečení*, tj. nebyla překročena maximální povolená hodnota.
- kontrola, zda nedošlo během výpočtu k *podtečení*, tj. výsledek není menší, než nejmenší povolená hodnota
- chybějící číslice vpravo se doplní nulami
- vrátí výsledek

Příklad sčítání dvou čísel v množině $M(10,6)$:

zadání	$0,256845 \cdot 10^1 + 0,327917 \cdot 10^{-3}$
denormalizace	$0,256845 \cdot 10^1 + 0,000327 \cdot 10^1$
součet	$0,257172 \cdot 10^1$

Všimněme si jednoho dosti nepříjemného jevu:
- u druhého operandu se výpočtu „účastní“ pouze cifry: ...327
- cifry ...917 jsou ztraceny!
- dochází ke *ztrátě platných cifer* během výpočtu

Příklad 4.5.6: Sčítání dvou čísel v $M(10,6)$

Poznámka: Jev zvaný *ztráta platných cifer* je způsoben nepřesnou aproximací čísla v paměti počítače. Ta je dána omezeným, konečným počtem bitů, které je možné vyhradit pro uložení čísla v paměti počítače včetně omezeného počtu bitů pro provedení požadované matematické operace.

Je-li tedy $\gamma : \mathbb{R} \to M$ a znamená-li symbol $\circledast$ aritmetickou operaci prováděnou výpočetním systémem a symbol $*$ tutéž operaci nad tělesem reálných čísel, dá se ověřit, že pro většinu výpočetních systémů platí:

$$x \circledast y = \gamma(x * y)\,; \quad x, y \in M$$

a

$$\left| \frac{x \circledast y - y * y}{x * y} \right| \leq kq^{1-t}$$

Aritmetickou operaci ve výpočetním systému (množině $M(q,t)$) můžeme napodobit tak, že provedeme obvyklou operaci a její výsledek zobrazíme na příslušné strojové číslo z $M(q,t)$.

Dá se také ukázat, že pro aritmetické operace v $M(q,t)$ obecně neplatí asociativní a distributivní zákony. Jedním z důsledků neplatnosti axiomů aritmetiky reálných čísel při strojním počítání je že stejný výpočet realizovaný rozdílnými algoritmy může dávat rozdílné výsledky.

Úloha:
Rozhodněte o výsledku výpočtu.

Mějme definovány dva částečné konečné součty následujícím předpisem:

1. $S_{n1} = \sum_{i=1}^{N} \frac{1}{i}$
2. $S_{n2} = \sum_{i=N}^{1} \frac{1}{i}$

Implementujeme dva přímočaré algoritmy, vycházející přímo z jednotlivých definic, které spočítají příslušné součty. Rozhodněte zda:

- $S_{n1} = S_{n2}$
- $S_{n1} > S_{n2}$
- $S_{n1} < S_{n2}$
- pokud jsou S_{n1} a S_{n2} různé, který výsledek je přesnější.

4.5.2 Operace násobení a dělení

Nyní zvolíme operace násobení (dělení) dvou čísel v plovoucí čárce.
Operace násobení pak probíhá tak příslušná výpočetní jednotka (ALU, FPU) postupuje dle:

- zjistí, zda některý z operandů neobsahuje „podivné hodnoty“, viz tabulka 4.4.4
- pokud ne, pak se sečtou hodnoty exponentů (s korekcí pro parametr *bias*)
- vynásobí se obě mantisy
- případně se normalizuje výsledek součinu
- doplní se výsledné znaménko (operace *nonekvivalence* - s_1 XOR s_2)
- výsledek se zaokrouhlí dle jednoho ze 4 určených postupů (rounding modes)
- kontrola, zda nedošlo během výpočtu k *přetečení*, tj. nebyla překročena maximální povolená hodnota.
- kontrola, zda nedošlo během výpočtu k *podtečení*, tj. výsledek není menší, než nejmenší povolená hodnota
- chybějící číslice vpravo se doplní nulami
- vrátí výsledek

Příklad násobení a dělení dvou čísel v množině $M(q,t)$:

1.číslo	$a_1 = (-1)^{s_1} \cdot 2^{e_1} \cdot m_1$
2.číslo	$a_2 = (-1)^{s_2} \cdot 2^{e_2} \cdot m_2$
zadání	$a = a_1 \cdot a_2$
součin	$a = (-1)^{s_1} \cdot (-1)^{s_2} \cdot 2^{(e_1+e_2)} \cdot (m_1 \cdot m_2)$
zadání	$b = \frac{a_1}{a_2}$
podíl	$b = (-1)^{s_1} \cdot (-1)^{s_2} \cdot 2^{(e_1-e_2)} \cdot \frac{m_1}{m_2}$

Slovně:

1. u součinu se sečtou exponenty základu a vynásobí mantisy
2. u podílu se odečtou exponenty základu a vydělí mantisy

Příklad 4.5.7: operace v $M(q,t)$

Úloha:
Pracujete v množině $M(10,5)$. Rozhodněte o výsledku výpočtu:

1. $1 - 10^{-7} = ?$
2. $1 + 10^{7} = ?$
3. $\frac{10^{10}}{10^{-15}} = ?$

Kapitola 5
Jazyk množin a formální logiky

Obsah kapitoly

Georg Cantor (1845-1918) vymyslel, vytvořil množiny, resp. teorii množin. Za zakladatele logiky je považován *Aristoteles (384–322 př. n. l.)*. Založil takzvanou sylogistickou logiku.

Princip sylogismu:

1. Premisa: Každý člověk je smrtelný.
2. Premisa: Sokrates je člověk.
3. Závěr: Sokrates je smrtelný.

Další jména Gottfried Wilhelm Leibniz (1646–1716), Bernard Bolzano (1781–1848), Gottlob Frege (1848–1925), Bertrand Russell (1872–1970).

Kurt Gödel (1906-1978) byl matematik rakouského původu, který se stal jedním z nejvýznamnějších logiků všech dob. Významné jsou i jeho příspěvky ve fyzice a ve filosofii matematiky.

George Boole (1815-1864) byl britský matematik a filosof, známý jako objevitel základů formální logiky, nazvané později *Booleovou algebrou*. Je považován za zakladatele informatiky, jakkoli v jeho době nebylo o počítačích ani uvažováno.

Motto (vyposlechnuto v tramvaji):
Babička ukazuje svému vnukovi - asi prvňákovi, velké parkoviště aut a ptá se: co to tam vidíš? Je tam hodně aut. Vnouček nic nechápe.
No, jak říkáte v matematice tomu, kde je hodně věcí pohromadě, pokračuje babička. Vnouček opět nic.
No, přeci množina aut, prozrazuje babička.
Vnouček vrtí hlavou. To není množina , auta nejsou v ohrádce a nejsou všechna stejná.

5.1 Abeceda množin

V předchozím textu jsme v konkrétních situacích množinu „definovali“ buď taxativně (výčtem prvků) - například množinu přirozených čísel N, nebo nějakou charakteristickou vlastností - například množinu racionálních čísel Q (viz také odst. 2.2).

Problém našeho stručného výkladu spočívá v tom, že jazyk množin a jazyk formální (matematické) logiky se tak prolínají, že i označování může být zdrojem nedorozumění.

Základní znaky: $x \in A$; x je prvkem množiny A;
$x \notin A$; x není prvkem množiny A;
$A \subset B$; množina A je podmnožinou množiny B;
každý prvek množiny A je prvkem množiny B;

Sjednocení	$A \cup B = \{x : x \in A \vee x \in B\}$, Sjednocení množin je množina těch prvků, které patří aspoň jedné množině, tj. patří jedné nebo druhé množině
Průnik	$A \cap B = \{x : x \in A \wedge x \in B\}$, Průnik množin je množina těch prvků, které patří zároveň obou množinám, tj. jsou přítomny v jedné i druhé množiny zároveň
Rozdíl	$A - B = \{x : x \in A \wedge x \notin B\}$.
Doplněk množiny	$A \subset E$ do množiny $E : A_E = E - A, x \in A_E \Longleftrightarrow x \in E \wedge x \notin A$.

Jestliže dvě množiny X a Y nemají žádné společné prvky říkáme, že tyto množiny jsou ***disjunktní*** a že jejich průnik je ***prázdná množina***:

$$X \cap Y = \emptyset \text{ (znak prázdné množiny).}$$

Vlastnosti množinových operací:

$$\begin{aligned} A \cup B &= B \cup A \\ A \cap B &= B \cap A \\ A \cup A &= A \\ A \cap A &= A \end{aligned}$$

Kartézský součin množin A, B

$$A \times B = \{(a,b) : a \in A \wedge b \in B\}$$

je *množina uspořádaných dvojic* prvků $a \in A, b \in B\}$.
Speciálně: $A \times A = A^2$; $A \times A \times A = A^3$, atd. $\Rightarrow$ $R^2 = R \times R$.

Blok 5.1.1: Slovník množin

Kartézský součin není komutativní:

$$A \times B \neq B \times A$$

pokud $A \neq \emptyset$, $B \neq \emptyset$ a $A \neq B$.

5.2 Abeceda výroků

Výrok V je sdělení, u něhož má smysl hovořit o pravdivosti či nepravdivosti, přičemž platí právě jedna možnost, tj. výrok nemůže být současně pravdivý i nepravdivý.

Negace výroku V značíme $\overline{V}$, (také non V,V', $\neg V$).

1. Výrok V: $x \in A$, čteme „x je prvkem množiny A“.
2. Výrok $\overline{V}$: $x \notin A$; čteme „x není prvkem množiny A“, není pravda, že x je prvkem množiny A.

Axiom dvouhodnotové logiky Výrok V je pravdivý právě tehdy, když $\overline{V}$ je nepravdivý („Tercium non datur“)

Složené výroky:

Konjunkce	$V_1 \wedge V_2$ je pravdivá právě tehdy, když jsou pravdivé oba výroky V_1, V_2; v ostatních případech je konjunkce nepravdivá.
Slovně:	V_1 a V_2.
Disjunkce	$V_1 \vee V_2$ je pravdivá, je-li pravdivý alespoň jeden z výroků V_1, V_2,
Slovně:	V_1 nebo V_2.
Implikace	$V_1 \Longrightarrow V_2$ je nepravdivá, je-li V_1 pravdivý a V_2 nepravdivý; v ostatních případech je implikace pravdivá.
Slovně:	z V_1 plyne V_2, V_1 je postačující pro V_2, V_2 je nutné pro V_1.
Ekvivalence	$V_1 \Longleftrightarrow V_2$ je pravdivá, jsou-li oba výroky pravdivé nebo oba nepravdivé.
Slovně:	V_1 právě tehdy, když V_2, V_1 je nutné a stačí pro V_2, V_2 je nutné a stačí pro V_1.

Blok 5.2.2: Složené výroky

Je zvykem značit pravdu jako 1 a nepravdu jako 0 a zachycovat situace pomocí pravdivostních tabulek. Pokud znáte pravdivostní hodnotu V_1 a V_2, najdete si je v prvních dvou sloupcích a pak vidíte hodnotu dalších složených výroků v příslušném řádku.

V_1	V_2	$\overline{V}_1$	$\overline{V}_2$	$V_1 \wedge V_2$	$V_1 \vee V_2$	$V_1 \Rightarrow V_2$	$V_2 \Rightarrow V_1$	$V_1 \Leftrightarrow V_2$
1	1	0	0	1	1	1	1	1
1	0	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0
0	0	1	1	0	0	1	1	1

Tabulka 5.2.1: Pravdivostní tabulka složených výroků

Například:

1. V_1: Daný trojúhelník je pravoúhlý.
2. V_2: V daném trojúhelníku platí rovnost $a^2+b^2=c^2$,
   kde: a, b jsou délky odvěsen, c je přepona.

Složený výrok $V_1 \Leftrightarrow V_2$ je známá *Pythagorova věta*:

Trojúhelník s délky stran a, b, c je pravoúhlý, právě když v tomto trojúhelníku platí pro délky stran rovnost $a^2+b^2=c^2$.

Logické zákony:

1.	$\overline{V_1 \wedge V_2}$	$\Leftrightarrow$	$\overline{V}_1 \vee \overline{V}_2$	negace konjunkce je ekvivalentní disjunkci negací
2.	$\overline{V_1 \vee V_2}$	$\Leftrightarrow$	$\overline{V}_1 \wedge \overline{V}_2$	negace disjunkce je ekvivalentní konjunkci negací
3.	$\overline{V_1 \Rightarrow V_2}$	$\Leftrightarrow$	$V_1 \wedge \overline{V}_2$	negace implikace - princip důkazu sporem
4.	$V_1 \Rightarrow V_2$	$\Leftrightarrow$	$\overline{V}_2 \Rightarrow \overline{V}_1$	princip důkazu sporem
5.	$V_1 \Rightarrow V_2$	$\Leftrightarrow$	$\overline{V}_1 \vee V_2$	princip nepřímého důkazu
6.	$V \Leftrightarrow \overline{\overline{V}}$			zákon dvojí negace („negace negace“)
7.	$V \vee \overline{V}$ $\overline{V \wedge \overline{V}}$, $V \Leftrightarrow \overline{\overline{V}}$			vždy pravdivé výroky; tautologicky pravdivé výroky
8.	$V_1 \Leftrightarrow V_2$	$\Leftrightarrow$		$(V_1 \Rightarrow V_2) \wedge (V_2 \Rightarrow V_1)$

Blok 5.2.3: Logické zákony

5.3 Kvantifikované výroky

1. Zápis: $\forall x \in M\colon \ V(x)$
   Slovně: pro každý prvek $x \in M$ platí $V(x)$ (každy prvek $x \in M$ má vlastnost$V(x)$).
2. Zápis: $\exists x \in M\colon V(x)$
   Slovně: existuje prvek $x \in M$ s vlastností $V(x)$.
3. Zápis: $\exists\, ! x \in M\colon V(x)$
   Slovně: existuje právě jeden prvek $x \in M$ s vlastností $V(x)$.

Blok 5.3.4: Kvantifikované výroky

1.	$\forall x \in \mathbb{N} : x > 0$	„každé přirozené číslo x je kladné“ - pravdivý výrok;
2.	$\forall x \in \mathbb{Q} : x < 5$	„každé racionální číslo je menší než 5“ - nepravdivý výrok;
3.	$\exists x \in \mathbb{N} : x > 0$	„existuje přirozené číslo, které je kladné“ - pravdivý výrok;
4.	$\exists x \in \mathbb{Q} : x < 5$	„existuje racionální číslo, které je menší než 5“ - pravdivý výrok;
5.	$\exists x \in \mathbb{R} : x^2 + 1 = 0$	„existuje reálné číslo x takové, že jeho druhá mocnina je -1“ - nepravdivý výrok;

Příklad 5.3.1: Příklady kvantifikovaných výroků - lekce čtení

Negace kvantifikovaných výroků:

$\overline{\forall x \in M : V(x)} \Leftrightarrow \exists x \in M : \overline{V(x)}$.

Slovně: Není pravda, že pro všechna $x \in M$ platí $V(x) \equiv$ existuje $x \in M$, pro které $V(x)$ neplatí.

$\overline{\exists x \in M : V(x)} \Leftrightarrow \forall x \in M : \overline{V(x)}$.

Slovně: Není pravda, že existuje $x \in M$ platí $V(x) \equiv$ pro všechna $x \in M$, je $V(x)$ nepravdivý $\equiv$ pro žádné $x \in M; V(x)$ neplatí.

Blok 5.3.5: Negace kvantifikovaných výroků

5.4 Komentář

Hned na začátku je třeba konstatovat, že logika není schopna rozhodovat, která tvrzení jsou pravdivá, čistě podle jejich obsahu. Takže pokud se proste zeptáte, zda mají trojúhelníky tři strany, tak nečekejte od logiky, že vám s tím pomůže. Logika dělá něco jiného. Vy ji dodáte vstupní data (ověření jejich pravdivosti je na vás), a podle jejich vzájemné provázanosti vám logika muže pomoci rozhodnout, zda platí (či neplatí) také nějaké jiné věci.

Pokud se rozhodnete začít předpokladem, že věta „Země má tvar krychle“ je pravdivá, tak na tom logika neuvidí nic špatného—jejím úkolem není soudit. Ona vám jen řekne, jaké důsledky by plynuly z tohoto předpokladu. V podstatě to znamená, že logiku zajímá jen struktura vytvořená jednotlivými daty, nikoliv data samotná.

Základním předmětem zkoumání formální logiky jsou výroky, což jsou věty, kterým lze přiřadit pravdivostní hodnotu (pravda/nepravda).

5.5 Zobrazení množin

5.5.1 Od obrazů zvířat k magnetické ezonanci

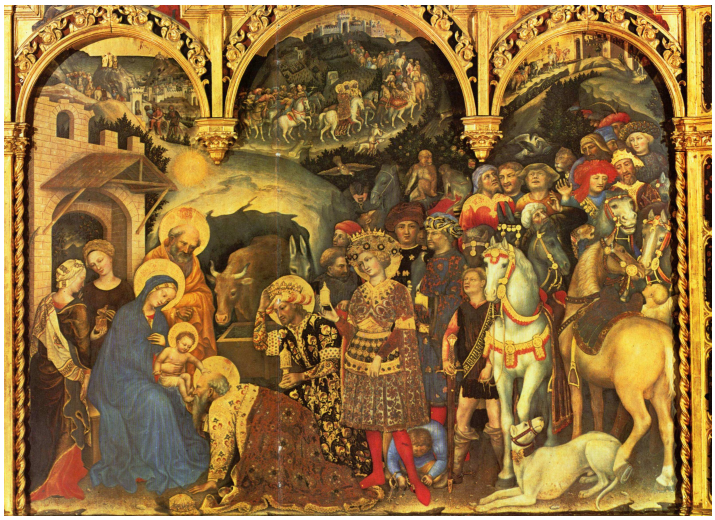
Starověcí umělci byli první, kteří začali zobrazovat reálné objekty. Postupně se rozvíjela geometrie jako nástroj zobrazování, tedy zobrazovací metody. Elektronika a počítače je zatím poslední nástroj zobrazování. Ruku v ruce s tím se rozvíjela matematická abstrakce základního zobrazovacího principu: objekt → obraz objektu.

Bez komentáře uveďme několik obrázků z historie výtvarného umění:

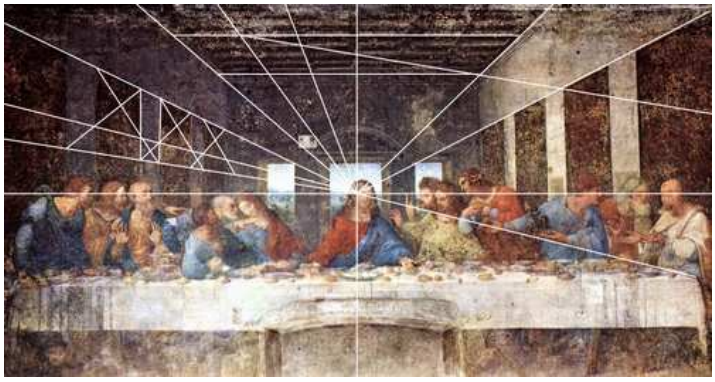
Obrázek 5.5.1: Mladší paleolit - jeskyně Altamira, Španělsko
(zdroj: https://cs.wikipedia.org/wiki/Altamira)

Obrázek 5.5.2: Egypt - 18.dynastie
(zdroj: https://cs.wikipedia.org/wiki/Egypt)

Obrázek 5.5.3: Starověké Řecko - Achiles zabíjí Penthesileiu
(zdroj: https://cs.wikipedia.org/wiki/Trojsk%C3%A1_v%C3%A1lka)

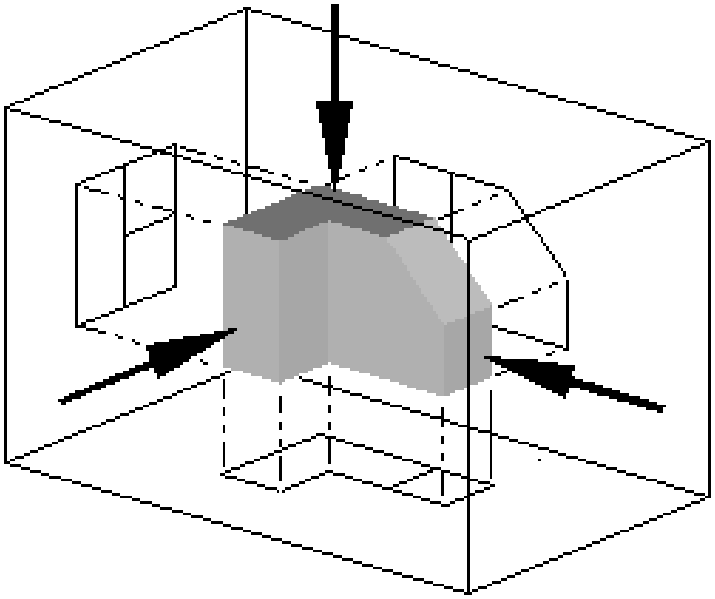


Obrázek 5.5.4: Gotika - Klanění tří králů (Gentile da Fabriano)
(zdroj: https://cs.wikipedia.org/wiki/Gotika)



Obrázek 5.5.5: Renesance - Leonardo Da Vicni - poslední večeře páně
(zdroj: https://cs.wikipedia.org/wiki/Line%C3%A1rn%C3%AD_perspektiva)

Principy zobrazování - současnost:

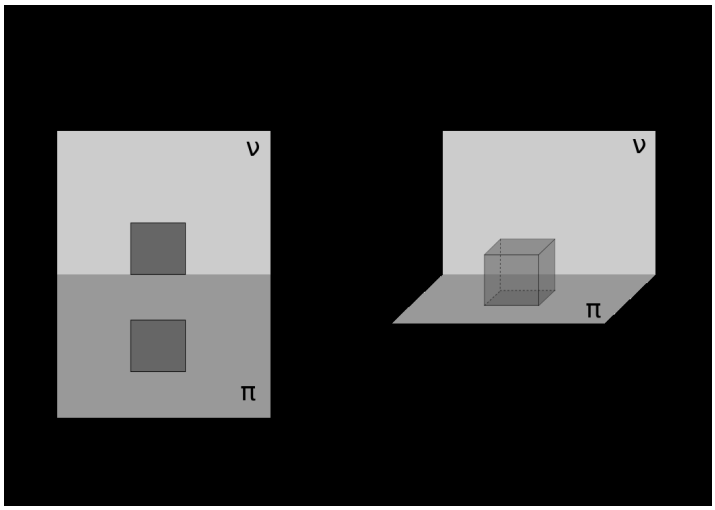


Obrázek 5.5.6: Desktiptivní geometrie - pravoúhlé promítání
(zdroj: https://cs.wikipedia.org/wiki/Technick%C3%BD_v%C3%BDkres)

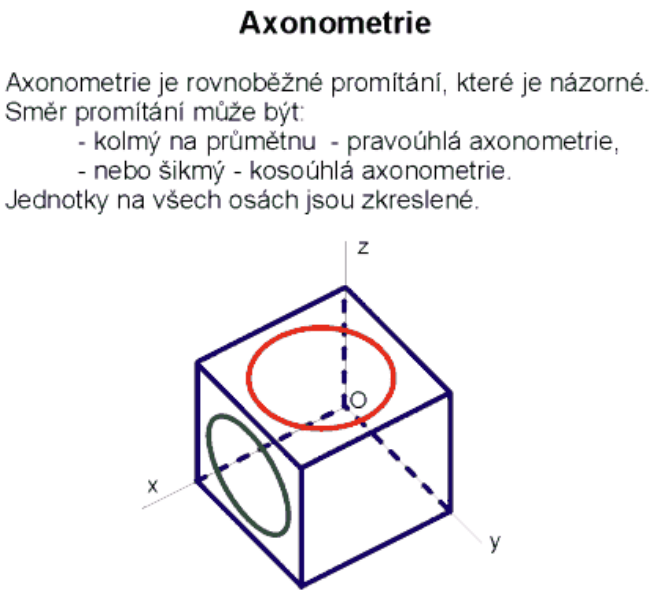
Rovnoběžné pravoúhlé promítání na dvě navzájem kolmé průmětny, *půdorysnu* a *nárysnu* se nazývá *Mongeovo promítání*.

Někdy je vhodné použít i třetí průmětnu. Většinou ji volíme kolmo k půdorysně i nárysně a nazývá se *bokorysna*.

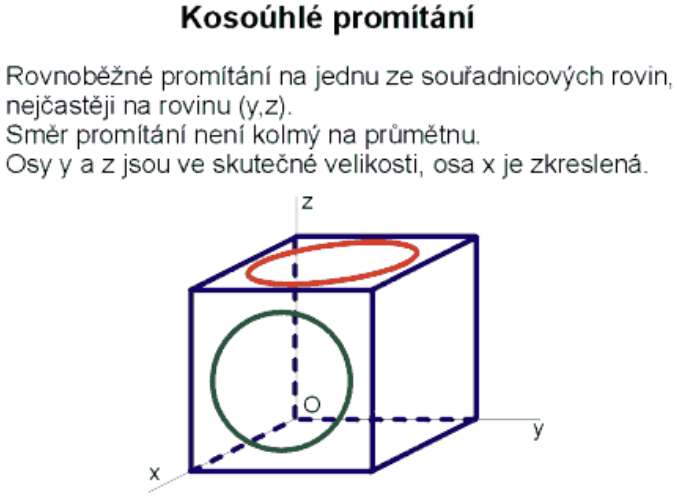
Všechny souřadnicové osy jsou ve skutečné velikosti. Zobrazení však není příliš názorné.



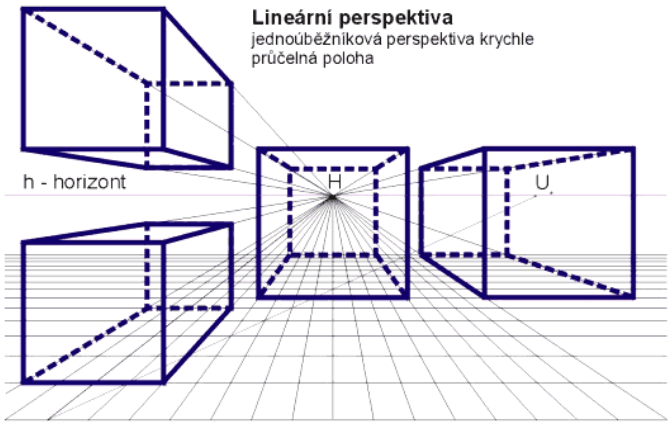
Obrázek 5.5.7: Desktiptivní geometrie - Mongeovo promítání
(zdroj: https://cs.wikipedia.org/wiki/Mongeovo_prom%C3%ADt%C3%A1n%C3%AD)



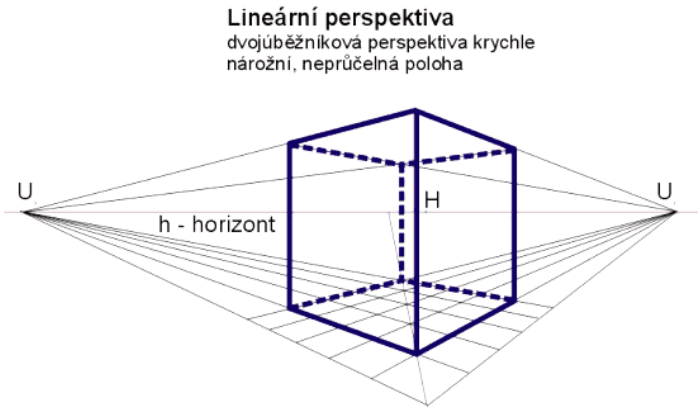
Obrázek 5.5.8: Axonometrie - izometrie
(zdroj: http://sisyfos.zcu.cz/matika/predm1/prednaska.htm)



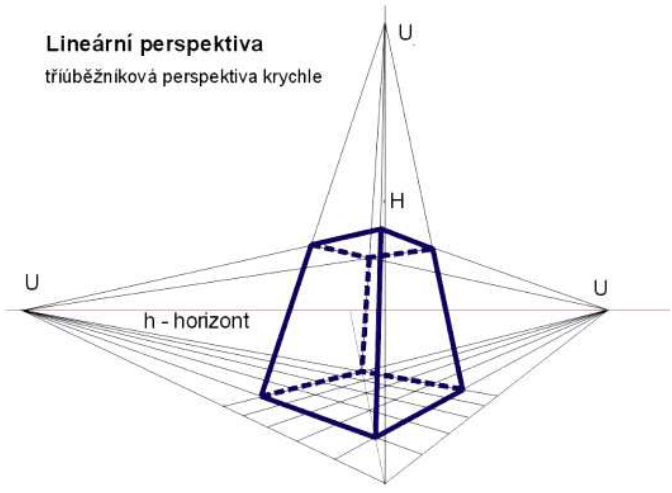
Obrázek 5.5.9: Axonometrie - kosoúhlé promítání
(zdroj: http://sisyfos.zcu.cz/matika/predm1/prednaska.htm)



Obrázek 5.5.10: Lineární perspektiva - jednoúběžníková perspektiva (zdroj: http://sisyfos.zcu.cz/matika/predm1/prednaska.htm)

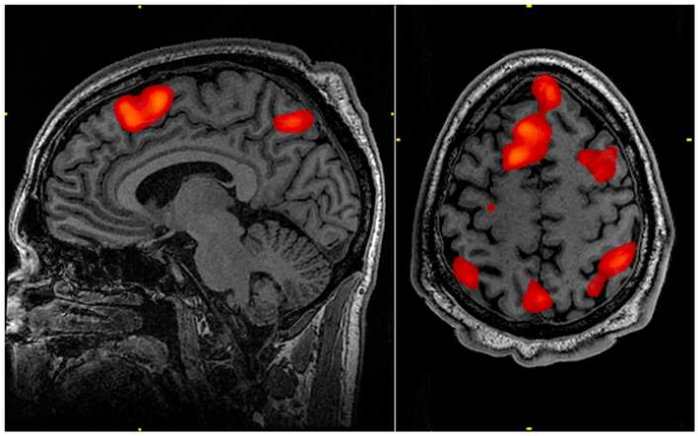


Obrázek 5.5.11: Lineární perspektiva - dvojúběžníková perspektiva (zdroj: http://sisyfos.zcu.cz/matika/predm1/prednaska.htm)



Obrázek 5.5.12: Lineární perspektiva - trojúběžníková perspektiva (zdroj: http://sisyfos.zcu.cz/matika/predm1/prednaska.htm)

Nukleární magnetická rezonance:



Obrázek 5.5.13: Nukleární magnetická rezonance - snímek zátěže paměti
(zdroj:
http://www.wikiskripta.eu/index.php/Nukle%C3%A1rn%C3%AD_magnetick%C3%A1_rezonance)

5.5.2 Abstrakce

V abstraktním pohledu na zobrazení a zobrazování se odpoutáme od toho, jestli zobrazovacím paprskem je proud fotonů (světlo), krátkovlnné elektromagnetické záření (paprsky X, CT), nebo elektromagnetické signály vyvolané magnetickým polem protonů.

Zavedeme označení a názvy:

1. $\mathcal{X}$ je neprázdná množina *vzorů, originálů, argumentů*;
   $x \in X$, x je vzor, nezávisle proměnná;

2. $\mathcal{Y}$ je neprázdná množina *obrazů, hodnot*;
   $y \in \mathcal{Y}$, y je obraz, závisle proměnná;

Uvědomíme si, že vzory a jejich obrazy jsou v nějakém *vztahu - relaci, závislosti přiřazení*. Označíme tento vztah nějakým symbolem, například: f, F; $\mathbf{F}, \mathbf{f}$; $\mathcal{A}, \mathcal{B}$; ...

V různých situacích používáme názvy: *funkce*, *posloupnost*, *transformace*, *operátor*, *funkcionál*.

Definiční obor zobrazení F označíme: $\mathcal{D}_F \subset \mathcal{X}$ ($\mathcal{D}_F$ je podmnožina $\mathcal{X}$), obor hodnot označíme: $\mathcal{H}_F \subset \mathcal{Y}$.

Používané zápisy:

$F : \mathcal{X} \to \mathcal{Y}$;	čteme:	F přiřazuje množině $\mathcal{X}$ množinu $\mathcal{Y}$; F přiřazuje každému prvku $x \in \mathcal{X}$ jediný prvek $y \in \mathcal{Y}$; F přiřazuje každému prvku $x \in \mathcal{X}$ jediný prvek $y = F(x) \in \mathcal{Y}$;
$F : \mathcal{D}_F \to \mathcal{Y}$, $\mathcal{D}_F \subset \mathcal{X}$; $F : x \to y$, $x \in \mathcal{D}_F$		
$y = F(x)$, $x \in \mathcal{D}_F$;	čteme:	prvek $y \in \mathcal{Y}$ je jediným obrazem prvku $x \in \mathcal{D}_F$;
$(x, y) \in F$, $x \in \mathcal{D}_F$, $y \in \mathcal{H}_F$	čteme:	F je množina všech uspořádaných dvojic vzorů a obrazů

Blok 5.5.6: Označení a názvy

Speciálně pokud v zobrazení $F : \mathcal{X} \to \mathcal{Y}$ je $\mathcal{Y} \subset \mathcal{X}$, resp. $H_F \subset \mathcal{X}$, říkáme, že jde o *transformaci množiny* $\mathcal{D}_F$ *v množině* $\mathcal{X}$. Termín *transformace* používáme v případě, kdy obrazy leží v množině vzorů.

Například: posunutí, otočení rovinného geometrického útvaru v $\mathbb{R}^2$.

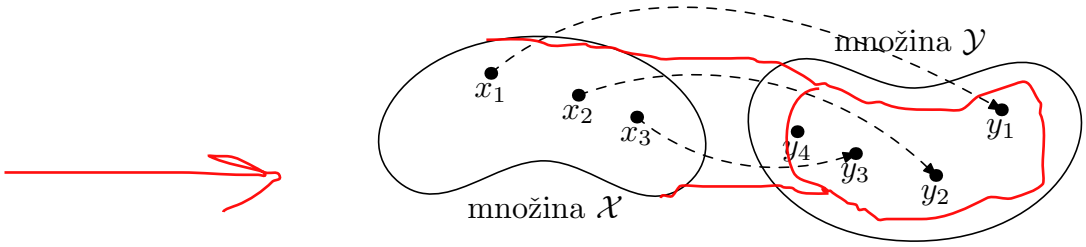
Přiřazení (relace) F prvků $x \in \mathcal{X}$, $y \in \mathcal{Y}$ nazveme *zobrazení množiny* $\mathcal{X}$ *do množiny* $\mathcal{Y}$, jestliže:

1. každému prvku $x \in \mathcal{X}$ je přiřazen jediný prvek $y \in \mathcal{Y}$, $y = F(x)$ předpisem, pravidlem f; tj. různým obrazům odpovídají různé vzory
2. $\forall y_1, y_2 \in \mathcal{H}_F$; $y_1 \neq y_2$; $\exists x_1, x_2 \in \mathcal{D}_F$, $(y_1 = F(x_1), y_2 = F(x_2)) \Rightarrow x_1 \neq x_2$.

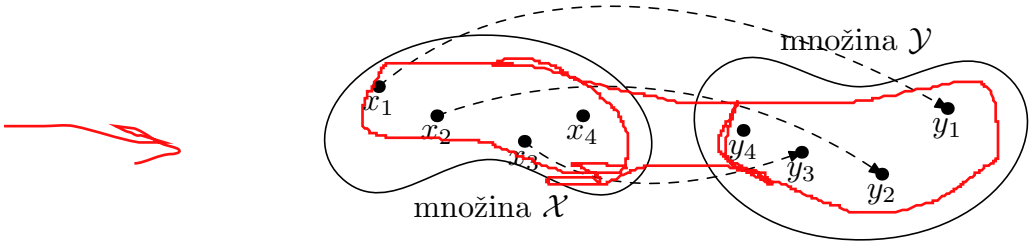
Blok 5.5.7: Zobrazení množiny $\mathcal{X}$ do množiny $\mathcal{Y}$

Máme čtyři názvoslovné formulace, které se matematickým významem dosti liší:

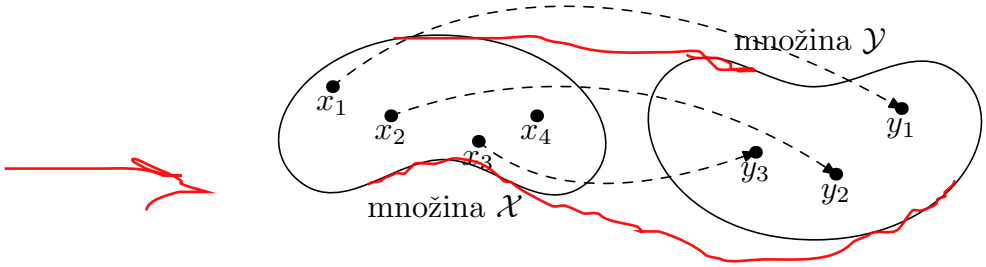
1. *zobrazení množiny* $\mathcal{X}$ *do množiny* $\mathcal{Y}$ - viz obrázek 5.5.14
(připouštíme, že v množině $\mathcal{Y}$ jsou prvky, které nemají svůj vzor);
2. *zobrazení z množiny* $\mathcal{X}$ *do množiny* $\mathcal{Y}$ - viz obrázek 5.5.15
(připouštíme, že v množině $\mathcal{X}$ jsou prvky, které se nezobrazují, tj. nepatří do definičního oboru zobrazení)
3. *zobrazení množiny* $\mathcal{X}$ *na množinu* $\mathcal{Y}$ - viz obrázek 5.5.16
(každý prvek z $\mathcal{Y}$ je obrazem)
4. *zobrazení množiny z* $\mathcal{X}$ *na množinu* $\mathcal{Y}$ - viz obrázek 5.5.17



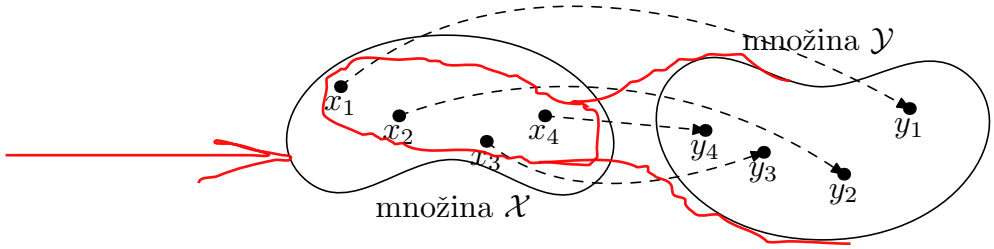
Obrázek 5.5.14: Zobrazení množiny $\mathcal{X}$ do množiny $\mathcal{Y}$



Obrázek 5.5.15: Zobrazení z množiny $\mathcal{X}$ do množiny $\mathcal{Y}$



Obrázek 5.5.16: Zobrazení množiny $\mathcal{X}$ na množinu $\mathcal{Y}$



Obrázek 5.5.17: Zobrazení z množiny $\mathcal{X}$ na množinu $\mathcal{Y}$

Surjekce je zobrazení „na množinu“:

$$\mathcal{H}_F \Rightarrow \mathcal{Y}\,;\ \text{resp. } \forall y \in \mathcal{Y}\,,\ \exists x \in \mathcal{X}:\ F(x) = y.$$

Injekce (prosté zobrazení) - platí-li implikace:

$$F(x_1) = F(x_2) \Rightarrow x_1 = x_2,$$

tj:

$$\forall x_1, x_2 \in \mathcal{D}_F:\ x_1 \neq x_2 \Rightarrow F(x_1) \neq F(x_2).$$

Inverzní zobrazení: $F^{-1}: \mathcal{H}_F \to \mathcal{D}_F$ k zobrazení $F: \mathcal{D}_F \to \mathcal{H}_F$ je právě takové zobrazení, které každému obrazu $y = F(x)$ přiřazuje jeho vzor x.
Postačující podmínkou existence inverzního zobrazení F^{-1} je, že zobrazení $F: \mathcal{D}_F \to \mathcal{H}_F$ je prosté.

Bijekce (vzájemně jednoznačné přiřazení množin $\mathcal{X}$ a $\mathcal{Y}$): *surjekce* a *injekce* (konjunkce vlastností), tj.:

$$x_1 \neq x_2 \Rightarrow F(x_1) \neq F(x_2) \wedge \forall y \in \mathcal{Y},\ \exists x \in \mathcal{X}:\ F(x) = y$$

Blok 5.5.8: Surjekce, injekce, bijekce

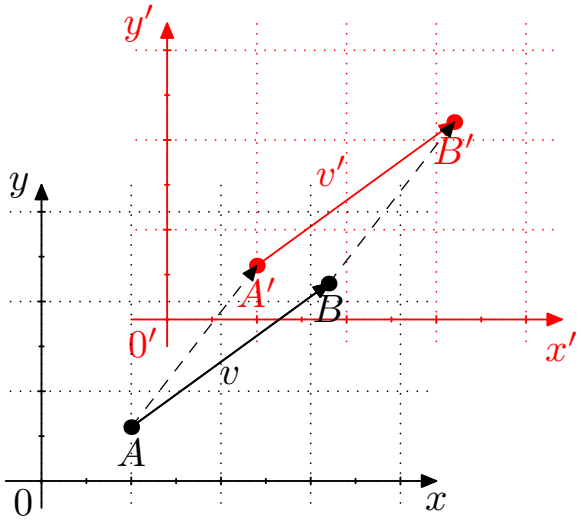
5.5.3 Konkretizace

Čtenář by z předchozího odstavce mohl nabýt dojmu, že jsme se zpronevěřili svým slibům a vydali se na cestu abstraktních matematických pojmů a „neužitečných“ poznatků. To zcela jistě není pravda. Samotná příroda nás však svoji složitostí k zobecňování nutí...

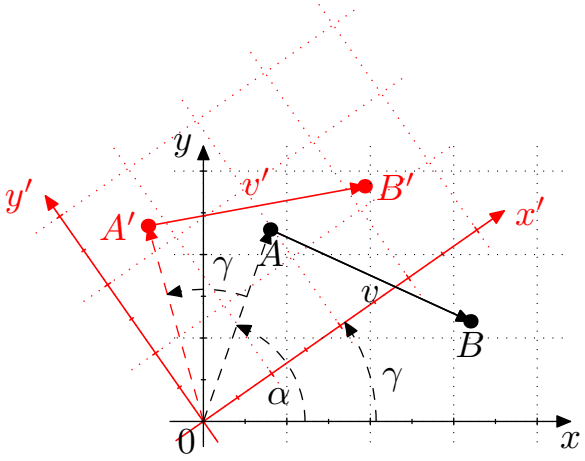
V kapitole 6 se chceme alespoň zmínit o vektorech a tenzorech, ale to bez pojmu *lineární zobrazení* neumíme a zdá se, že to ani nejde.

O číslech máme v sobě zakódovánu moudrost věků a jejich vlastnosti poznáváme od předškolního věku. S vektory se seznamujeme od puberty a tenzory poprvé potkáme až na přednášce z fyziky, případně diferenciální geometrie. Většinou si odnášíme poznatek, že fyzikové a geometři mluví naprosto jiným jazykem.

Věnujme pozornost obrázkům 5.5.18 a 5.5.19.



Obrázek 5.5.18: Posunuté systémy



Obrázek 5.5.19: Otočené systémy

Co pozorujeme? Vektor ~~(y)~~ se nezměnily při změně souřadnicového systému. Totéž se dá ukázat pro vektory typu $n-tice$ čísel, případně zcela obecně pro prvky lineárního prostoru.

Toto je ta ***potřebná vlastnost***, aby nějaký objekt byl *vektorem*, případně *tenzorem*!

Kapitola 6
Skaláry, vektory, tenzory

Obsah kapitoly

6.1 Jazyk přírody

Pojmy čísla a bodu jsou abstraktní nástroje pro vyjádření polohy, vzdálenosti, množství, atd. Bylo třeba měřit veličiny jako čas, teplotu, hmotnost, energii, sílu, rychlost, moment setrvačnosti, napětí, ... Mezi fyzikální veličiny řadíme:

- *skaláry*
  k jejich popisu stačí reálná čísla s doplněním jednotky (sekunda, stupeň), směrovost nepotřebujeme.

- *vektory*
  k vystižení směrovosti potřebujeme již dvojici, trojici, n-tici, reálných čísel, pokud se spokojíme s popisem jednoho směru. Působí-li na těleso (částici, bod) více sil, pak se pohyb odehrává v tom směru, ve kterém působí výslednice všech sil, vektory musíme umět sčítat, odčítat.

- *tenzory*
  jsou ovšem veličiny, které nemají jednu směrovost a navíc tyto směrovosti se nesmí skládat jako vektory. Například silové působení v pevných látkách a kapalinách se popisuje tenzorem napjatosti (napěťor). Potřebujeme více n-tic reálných čísel a k tomu specificky definované operace.

Vhodným matematickým modelem skaláru je reálné číslo, modelem vektoru je n-tice čísel, nebo orientovaná úsečka, modelem tenzoru jsou určité specifické soubory čísel. Hovoříme o složkách vektoru, tenzoru. Matematickou představu a interpretaci dáváme (dostáváme) pomocí souřadnicových systémů.

6.2 Prostory a souřadnicové systémy

Motto:

Nechápu, co matematici na těch sprostorech nebo prostorech milují. Asi jsou to nějaké sprosťárny. (odposlechnuto)

Prostory v matematice jsou jako byty v paneláku. Liší se výhledem a vybavením, ale jinak jsou na jedno brdo (S.Míka)

Vesmír. či kosmos je souhrnné označení veškeré hmoty, času a prostoru. Myslím, tudíž jsem (Descartes)

Zdravý rozum je patrně ta nejlépe rozdělená věc na světě, neboť každý si myslí že jej má dost a ani ti, kdo jsou v jiných nejvíce nenasytní, vůbec netouží mít jej víc, než mají. (Descartes)

Termínem *prostor* označujeme prostředí, ve kterém:

- žijeme (životní prostor, bytový prostor),
- pracujeme (komerční prostor),
- zkoumáme a popisujeme různé stavy, děje a procesy (fázový prostor, konfigurační prostor),
- zkoumáme polohu, vzdálenosti a rozměry (metrický prostor, normovaný prostor)

Poloha objektu v prostoru je dány vždy vzhledem k nějakým okolním objektům. tj. vzhledem k nějaké *vztažné soustavě*.

V matematice termínem *prostor* označujeme neprázdnou množinu prvků opatřenou určitou *strukturou*, tj. s definovanými relacemi (rovnost, vztah k číslům) a operacemi (sčítání, skládání, příp. určitý typ násobení). Prvkům takové množiny říkáme obecně *body*, speciálně pak $n-tice$ čísel, vektory, funkce, operátory, posloupnosti, ap.

V prostoru je nejdůležitější pojem *lineární nezávislost* a *lineární závislost* prvků prostoru. Ten umožňuje definovat pojmy *báze* a *dimenze* prostoru, tedy to, co určuje *souřadnicový systém*.

6.2.1 Bodový prostor $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R}$

Množinu všech uspořádaných $n-tic$ reálných čísel označujeme:

$$\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R} \equiv \{(x_1, x_2, \ldots, x_i, \ldots x_n) \,:\, x_i \in \mathbb{R} \quad \text{pro: } i = 1,2,3,\ldots,n\}.$$

Tj.:

$$\mathbb{R}^1 \equiv \mathbb{R} \equiv (-\infty, +\infty)$$

Tento prostor se dá ztotožnit s přímkou (tzv. reálná osa). Prvky tohoto prostoru nazveme body na přímce a příslušné reálné číslo nazveme souřadnicí tohoto bodu. Protože k určení polohy bodu na přímce stačí jedna souřadnice, říkáme, že prostor $\mathbb{R}^1$ je jednorozměrný; píšeme $\dim \mathbb{R}^1 = 1$.

$$\mathbb{R}^2 \equiv \mathbb{R} \times \mathbb{R} \equiv \{(x_1, x_2) : \ x_1 \in \mathbb{R} \ \wedge \ x_2 \in \mathbb{R}\}$$

Tento prostor se dá ztotožnit s rovinou. Prvky tohoto prostoru nazveme body v rovině a čísla x_1, x_2 jsou souřadnice tohoto bodu. Protože k určení polohy bodu v rovině jsou potřebné dvě souřadnice, říkáme, že prostor $\mathbb{R}^2$ je dvourozměrný a píšeme $\dim \mathbb{R}^2 = 2$.

Množinu všech *uspořádaných trojic* reálných čísel označujeme:

$$\mathbb{R}^3 \equiv \mathbb{R} \times \mathbb{R} \times \mathbb{R} \equiv \{(x_1, x_2, x_3) : \ x_1 \in \mathbb{R}, \ x_2 \in \mathbb{R}, \ x_3 \in \mathbb{R}\}.$$

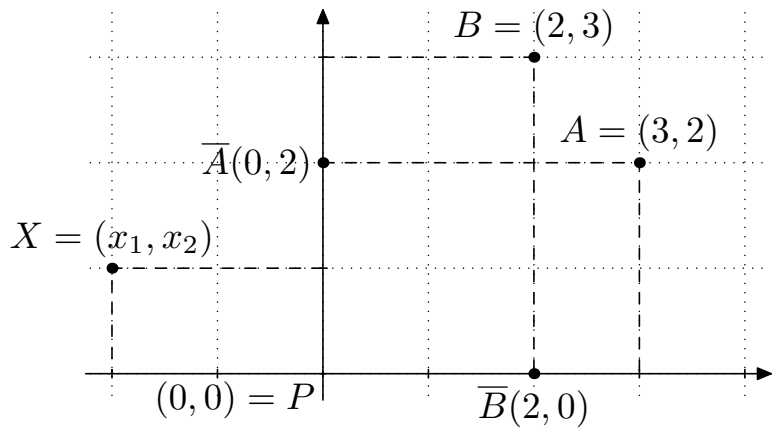
Čísla x_1, x_2, x_3 jsou *souřadnice* tohoto bodu ve zvoleném souřadnicovém systému.

René Descartes (lat. Renatus Cartesius, 1596-1650) přiřadil Eukleidovským bodům skupiny čísel a nazval je *souřadnice*. Vymyslel tak jeden z největších matematických výsledků tím, že vynalezl *analytickou geometrii*. Umožnil tím převádět geometrické konstruování na počítání, tj. řešení rovnic.

Bod $X = (x_1, x_2) \in \mathbb{R}^2$ interpretujeme (stanovíme jeho polohu v rovině) takto:

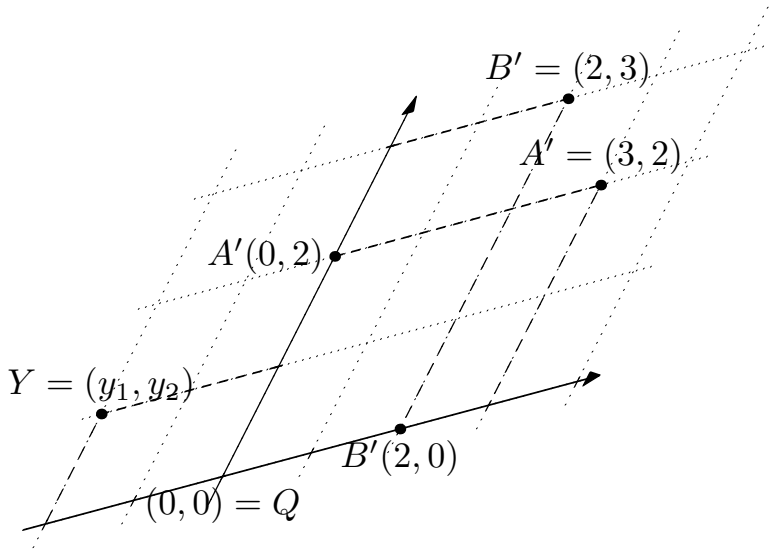
V $\mathbb{R}^2$ volíme dva navzájem kolmé systémy rovnoběžných přímek (tzv. přímková pravoúhlá síť). Dále zvolíme z tohoto systému dvě navzájem kolmé přímky jako vztažný systém. Jejich průsečíku P přiřadíme dvojici čísel $(0,0)$ a nazveme jej počátek, tj. $P = (0,0)$. Ještě zvolíme měřítko. Říkáme, že jsme v $\mathbb{R}^2$ zvolili *souřadnicový systém* nebo *soustavu souřadnic*.

Další informace jsou patrné z obrázku 6.2.1 Ještě si všimneme, že souřadnice všech bodů na „vodorovné“ vztažné přímce (souřadnicová osa x_1) mají tvar $(x_1, 0)$ a souřadnice všech bodů na „svislé“ vztažné přímce (souřadnicová osa x_2) mají tvar $(0, x_2)$.



Obrázek 6.2.1: Pravoúhlý (kartézský) souřadnicový systém v $\mathbb{R}^2$,$[P, x_1, x_2]$

V $\mathbb{R}^2$ můžeme zavést kosoúhlý souřadnicový systém (viz obrázek 6.2.2). Opět zvolíme síť (pouze) různoběžných přímek a vybereme vztažný systém, počátek $Q = (0,0)$ a měřítko.



Obrázek 6.2.2: Kosoúhlý souřadnicový systém v $\mathbb{R}^2$, $[Q, y_1, y_2]$

Pro určení polohy bodu potřebujeme:

1. volbu počátku;

2. orientaci souřadnicových čar (os);

3. jednotky (měřítko).

V *kartézském souřadnicovém systému* $\mathbb{R}^2$ definujeme (Eukleidovskou) vzdálenost bodů A a B jako nezáporné číslo:

$$d(A,B) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}; \quad A = (a_1, a_2),\ B = (b_1, b_2).$$

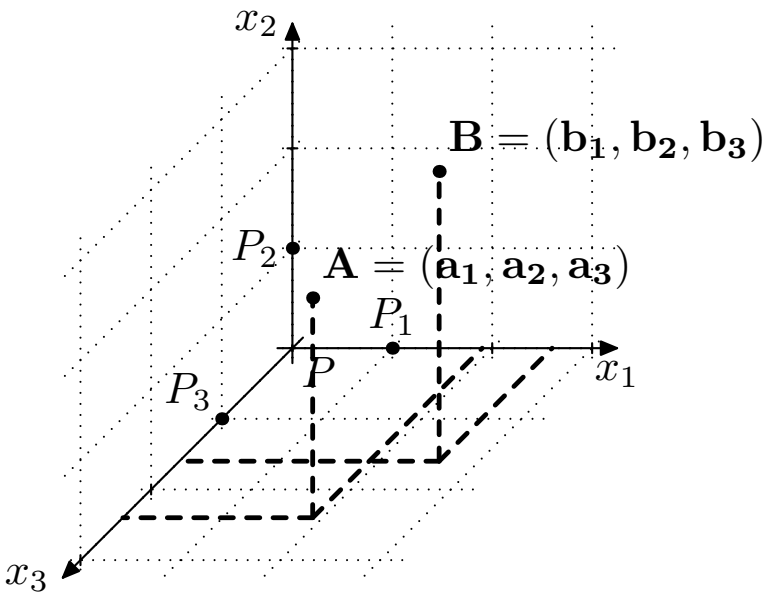
V $\mathbb{R}^3$ zvolíme tří navzájem různoběžné *souřadnicové čáry* se společným průsečíkem P, který nazýváme *počátek*. Na obrázku (6.2.3) je znázorněn pravoúhlý souřadnicový systém (*kartézský systém*). Alternativně je možné zvolit tři navzájem kolmé souřadnicové roviny (průmětny), viz obrázek 6.2.3.

V $\mathbb{R}^3$ zvolíme počátek $P = (0,0,0)$ a tři body $P_1 = (1,0,0)$, $P_2 = (0,1,0)$ a $P_3 = (0,0,1)$ (viz obrázek 6.2.3). Souřadnicové přímky jsou PP_1, PP_2 a PP_3. Zvolili jsme tak orientaci souřadnicových přímek i měřítko v $\mathbb{R}^3$.

V *kartézském souřadnicovém systému* $\mathbb{R}^3$ definujeme (Eukleidovskou) vzdálenost bodů A a B jako nezáporné číslo:

$$d(A,B) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2}; \quad A = (a_1, a_2, a_3),\ B = (b_1, b_2, b_3).$$

Poznámka: Obecně se vzdálenost dvou prvků nějaké množiny zavádí axiomaticky, tj. nepotřebujeme k tomu souřadnicový systém.



Obrázek 6.2.3: Pravoúhlý (kartézský) souřadnicový systém v $\mathbb{R}^3$

Transformace souřadnic:

Volba souřadnicového systému a převody souřadnic jsou velmi důležité matematicko-technologické nástroje pro modelování procesů a jevů různé povahy. Vyložíme elementární princip převodů na souřadnicových systémech $[P;x_1,x_2]$ a $[Q;y_1,y_2]$ z obrázků 6.2.1 a 6.2.2.

Převod: $[P;x_1,x_2] \to [Q;y_1,y_2]$:

Představíme si, že jsme obrázek 6.2.2 překopírovali do obrázku 6.2.1 tak, že počátek $Q=(0,0)_Y$ přesuneme do nějakého bodu $\overline{Q}=(q_1,q_2)_X$ v systému $[P;x_1,x_2]$.

Souřadnicová osa y_1 bude přímkou o rovnici:

$$a_1x_1+b_1x_2+c_1=0.$$

Souřadnicová osa y_2 bude přímkou o rovnici:

$$a_2x_1+b_2x_2+c_2=0.$$

kde koeficienty a_i,b_i,c_i $i=1,2$ jsou konkrétní čísla.

Řešením této soustavy je právě dvojice (q_1,q_2).

Zobrazení (převod) $[P;x_1,x_2] \to [Q;y_1,y_2]$ definujeme proto *transformačními vztahy* (kterým se dosti často říká transformační rovnice i když to žádné rovnice nejsou – ne všemu, kde se vyskytuje symbol „=“ se musí říkat „rovnice“!).

$$\begin{aligned} y_1 &= a_1x_1+b_1x_2+c_1,\\ y_2 &= a_2x_1+b_2x_2+c_2. \end{aligned}$$

Známe-li souřadnice x_1,x_2 nějakého bodu, můžeme jednoznačně určit souřadnice y_1,y_2 téhož bodu v druhém souřadnicovém systému.

Převod: $[Q;y_1,y_2] \to [P;x_1,x_2]$:

Představíme si nyní převrácenou (inverzní) situaci a obrázek 6.2.1 překopírujeme do obrázku 6.2.2 tak, že počátek $P = (0,0)_X$ přesuneme do nějakého bodu $\overline{P} = (p_1, p_2)_Y$ v systému $[Q; y_1, y_2]$. Souřadnicové osy x_1, x_2 budou přímkami o rovnicích:

$$c_1 y_1 + d_1 y_2 + e_1 = 0,$$
$$c_2 y_1 + d_2 y_2 + e_2 = 0.$$

Zobrazení (převod) $[Q; y_1, y_2] \to [P; x_1, x_2]$ definujeme transformačními vztahy:

$$x_1 = c_1 y_1 + d_1 y_2 + e_1,$$
$$x_2 = c_2 y_1 + d_2 y_2 + e_2.$$

Poznámka: jistě nás napadlo, zda bychom vztahy $[Q; y_1, y_2] \to [P; x_1, x_2]$ dokázali vypočítat ze vztahů $[P; x_1, x_2] \to [Q; y_1, y_2]$. Jde to? Kladná odpověď závisí na podmínce, zda matice koeficientů:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \text{ a } \begin{bmatrix} c_1 & d_1 \\ c_2 & d_2 \end{bmatrix}$$

jsou navzájem inverzní!

Doporučení: Formulujte sami otázky typu:

1. jak se zobrazí jednotlivé rovinné geometrické útvary (přímky, úsečky, trojúhelník, křivky),
2. jak vymyslet transformační vztahy, aby obrazem elipsy byla kružnice,
3. jak souvisí tyto elementy s tzv. *Lorentzovou transformací* v základech Einsteinovy teorie relativity.

Dvojici čísel z $\mathbb{R}^2$, nebo trojici čísel z $\mathbb{R}^3$ můžeme interpretovat jako souřadnice bodu i v jiných souřadnicových systémech. Uveďme:

Polární souřadnicový systém v $\mathbb{R}^2$:
Polární souřadnicovou síť tvoří soustava soustředných kružnic se středem ve zvoleném bodě P a systém různoběžných přímek se společným průsečíkem právě v bodě P. Dvojici souřadnic pak označujeme:

$$\varrho \in R_+ \; \alpha \in \langle 0, +2\pi \rangle$$

viz obrázek 6.2.4

Vztažný systém tvoří dvě vybrané orientované polopřímky vycházející ze zvoleného bodu P.

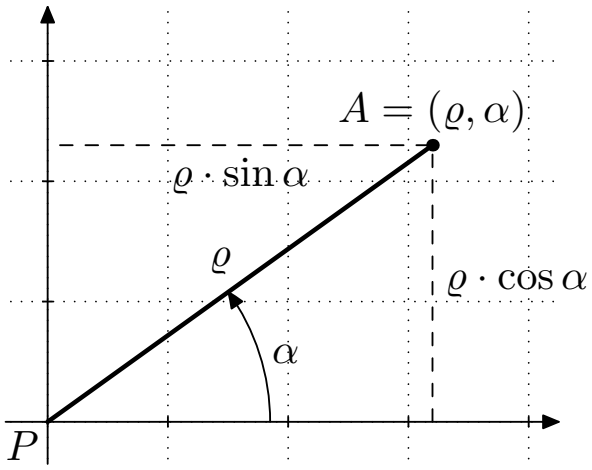
Transformaci *polárních* souřadnic ϱ, α na *kartézské* souřadnice x, y definujeme takto:

$$x = \varrho \cdot \cos\alpha,$$
$$y = \varrho \cdot \; sin\alpha.$$

Transformaci *kartézských* souřadnic na *polární* definujeme takto:

$$\varrho = \sqrt{x^2 + y^2}.$$
$$\alpha = inv(x, y)$$

kde funkce $inv(x, y)$ je definována takto: (v MA II, na straně 176 jsem nenašel nic, co by připomínalo inverzní funkci)



Obrázek 6.2.4: Polární souřadnicový systém v $\mathbb{R}^2$

6.2.2 Eukleidovský (vektorový) prostor $\mathbb{E}^n$

Prvky množiny $\mathbb{E}^n$ z tohoto odstavce budeme nazývat *vektory* a budeme je interpretovat, jako sloupce (sloupcové matice) reálných čísel , kterým říkáme složky nebo také souřadnice.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad = \quad [x_1, x_2, \ldots, x_n]^T \qquad \text{sloupcový vektor (sloupec),}$$

Což jsou dva různé zápisy prvku $\mathbf{x} \in \mathbb{E}^n$

$$\mathbf{x}^T = [x_1, x_2, \ldots, x_n] \quad \text{řádkový vektor,}$$

V množině $\mathbb{E}^n$ definujeme rovnost vektorů a operaci sčítání vektorů a násobení vektoru číslem s těmito vlastnostmi (axiomy) :

- Dva vektory $\mathbf{x}, \mathbf{y} \in \mathbb{E}^n$ jsou si rovny právě když jsou si rovny jejich odpovídající složky, tj.
$$\mathbf{x} = \mathbf{y} \iff x_i = y_i.$$
- Definujeme sčítání, tj. každým dvěma vektorům $\mathbf{x} \in \mathbb{E}^n$, $\mathbf{y} \in \mathbb{E}^n$ přiřadíme vektor $\mathbf{z} \in \mathbb{E}^n$ nazývaný *součtem* takto:
$$\mathbf{z} = \mathbf{x} + \mathbf{y} = [x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n]^T, \text{ tj. } z_i = x_i + y_i.$$
- Definujeme *násobení* vektoru číslem, tj. každému $\mathbf{x} \in \mathbb{E}^n$ a každému číslu $\alpha \in \mathbb{R}$ přiřadíme vektor $\alpha \cdot \mathbf{x} \in \mathbb{E}^n$ takto:
$$\alpha \cdot \mathbf{x} = [\alpha x_1, \alpha x_2, \ldots, \alpha x_n]^T \text{ (násobek vektoru číslem).}$$
- $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$, komutativita;

- $\mathbf{x}+(\mathbf{y}+\mathbf{z})=(\mathbf{x}+\mathbf{y})+\mathbf{z}$, asociativita;
- $\mathbf{0}_n \in \mathbb{R}^n : \mathbf{x}+\mathbf{0}_n=\mathbf{x}$, existence nulového vektoru: $\mathbf{0}_n=[0,0,0,\ldots,0]^T$ (index n označuje, že jde o nulový prvek prostoru $\mathbb{E}^n$);
- $\forall \mathbf{x} \in \mathbb{R}^n$, $\exists \overline{\mathbf{x}} \in \mathbb{R}^n : \mathbf{x}+\overline{\mathbf{x}}=\mathbf{0}$, existence opačného vektoru: $\overline{\mathbf{x}}=-\mathbf{x}$;
- $\alpha \cdot (\beta \mathbf{x})=(\alpha \cdot \beta)\mathbf{x}$;
- $\alpha(\mathbf{x}+\mathbf{y})=\alpha\mathbf{x}+\alpha\mathbf{y}$;
- $(\alpha+\beta)\mathbf{x}=\alpha\mathbf{x}+\beta\mathbf{x}$;
- $1 \cdot \mathbf{x}=\mathbf{x}$.

Důsledky axiomů:

1. Existuje jediný nulový vektor;
2. Ke každému vektoru existuje jediný opačný vektor;
3. $0 \cdot \mathbf{x}=\mathbf{0}$;
4. $-\mathbf{x}=(-1)\mathbf{x}$;
5. Rovnost: $\mathbf{x}=\mathbf{y}$ právě když $\mathbf{x}-\mathbf{y}=\mathbf{0}$, $\mathbf{x}-\mathbf{y}=\mathbf{x}+(-1)\mathbf{y}$;

Souřadnice v $\mathbb{E}^n$: Zvolíme n lineárně nezávislých *bázových vektorů*:

$$\begin{array}{lcl} e_1 & = & [1,0,0,\ldots,0]^T \\ e_2 & = & [0,1,0,\ldots,0]^T \\ \ldots & & \ldots \\ e_n & = & [0,0,0,\ldots,1]^T \end{array}$$

Každý vektor $\mathbf{x}=[x_1,x_2,\ldots,x_n]^T$ pak můžeme vyjádřit jako *lineární kombinaci bázových vektorů*, tj. ve tvaru:

$$\mathbf{x}=x_1 \cdot \mathbf{e}_1+x_2 \cdot \mathbf{e}_2+x_3 \cdot \mathbf{e}_3+\cdots+x_n \cdot \mathbf{e}_n;$$

a číslům: x_i, $i=1,2,\ldots,n$ říkáme souřadnice vektoru $\mathbf{x}$ v bázi$\{\mathbf{e}_i\}$.

Skalární součin:

Skalární součin dvou vektorů $\mathbf{x},\mathbf{y} \in \mathbb{E}^n$ je reálné číslo:

$$\sum_{i=1}^{n} x_i \cdot y_i = [\text{řádek } x_i] \cdot [\text{sloupec } y_i] \overset{ozn}{=} \mathbf{x}^T\mathbf{y} \overset{ozn}{=} x_i y_i \quad \text{(sumační konvence)}.$$

Má tyto vlastnosti:

1. $\mathbf{x}^T\mathbf{y}=\mathbf{y}^T\mathbf{x}$,
2. $(\mathbf{x}+\mathbf{y})^T\mathbf{z}=\mathbf{x}^T\mathbf{z}+\mathbf{y}^T\mathbf{z}$,
3. $(\alpha\mathbf{x})^T\mathbf{y}=\alpha\mathbf{x}^T\mathbf{y}$, pro libovolné reálné číslo $\alpha \in \mathbb{R}$

4. $\mathbf{x}^T\mathbf{x} > 0$ pro $\mathbf{x} \neq \mathbf{0}$.

Norma vektoru $\mathbf{x} \in \mathbb{E}^n$ je nezáporné číslo:

$$\sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}} \overset{ozn}{=} ||\mathbf{x}|| \overset{ozn}{=} (x_i^2)^{\frac{1}{2}} \text{ (sumační konvence).}$$

Má tyto vlastnosti:

1. $||\mathbf{x}|| > 0$ pro $\mathbf{x} \neq \mathbf{0}$ (rozlišujeme 0 a $\mathbf{0}$)
2. $||\alpha\mathbf{x}|| = |\alpha| \cdot ||\mathbf{x}||$ pro libovolné $\alpha \in \mathbb{R}$
3. $||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$ (trojúhelníková nerovnost)

Nezáporné číslo $d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||$ definuje *metriku* v $\mathbb{E}^n$.

Cauchyova-Schwartzova nerovnost:

1. $|\mathbf{x}^T\mathbf{y}| \leq ||\mathbf{x}|| \cdot ||\mathbf{y}||$ pro každé $\mathbf{x},\mathbf{y} \in \mathbb{E}^n$,
2. pro nenulové $\mathbf{x},\mathbf{y}$ existuje reálné číslo

$$\varphi = \arccos \frac{\mathbf{x}^T\mathbf{x}}{||\mathbf{x}|| \cdot ||\mathbf{y}||},\ \varphi \in (0,\pi),$$

tj. $\mathbf{x}^T\mathbf{y} = ||\mathbf{x}|| \cdot ||\mathbf{y}|| \cos\varphi$, ϕ je odchylka vektorů $\mathbf{x}$, $\mathbf{y}$.

Poznámky:

- Úplný název pro $\mathbb{E}^n$ je: Eukleidovský prostor nad tělesem reálných čísel. Pokud termín „reálné číslo“ nahradíme termínem „komplexní číslo“ a vlastnosti skalárního součinu modifikujeme, dostaneme prostor $\mathbb{C}^n$: Eukleidovský prostor nad tělesem komplexních čísel.
- V Eukleidovském prostoru není definován tzv. *vektorový součin orientovaných úseček*, známý z fyziky a z analytické geometrie.

  Jedním z důvodů je to, že neumíme v $\mathbb{E}^n$ matematicky definovat „pravidlo pravé ruky“. *Vektorové násobení* je operace, které se dá matematicky definovat až v rámci *tenzorového počtu* jako speciální případ součinu dvou tenzorů.

6.2.3 Afinní prostor $\mathbb{A}$

V geografii, kartografii a především v GIS (geografické informační systémy) a GPS potřebujeme určovat a zobrazovat *polohu*. Vystačíme proto s prostorem bodů (tj. $\mathbb{R}^m$ s jeho různými souřadnicovými systémy).

Ve fyzice, mechanice, inženýrství však potřebujeme určovat a popisovat *směry*, potřebujeme navíc prostor vektorů (tj. $\mathbb{E}^n$). K popisu polohy a směru potřebujeme prostředí (prostor), kde jsou jak body tak i vektory.

Takovým „hybridem“ je vlastně množina geometrických bodů a orientovaných úseček s níž se pracuje v analytické geometrii.

Body neumíme sčítat, neumíme je násobit (prostor bodů $\mathbb{R}^n$ není vybaven základní strukturou). Vektory umíme sčítat a umíme je násobit číslem (prostor vektorů je vybaven strukturou lineárního prostoru).

Nebudeme se pouštět do velkých abstrakcí a zůstaneme u obecnější analytické geometrie.

Vytvoříme prostor $\mathbb{A}$ *spojením (přiřazením) prostorů* $\mathbb{R}^m$ a $\mathbb{E}^n$ podle těchto pravidel: (znak „=“ užíváme ve smyslu přiřazení), značíme $\mathbf{A} = \{\mathbb{R}^m, \mathbb{E}^n\}$:

1. Prvky (body) prostoru $\mathbb{R}^m$ budeme označovat velkými písmeny $O, P, X, Y, \ldots$.
2. Prvky (vektory) prostoru $\mathbb{E}^n$ budeme označovat malými zvýrazněnými písmeny $\mathbf{o}, \mathbf{p}, \mathbf{x}, \mathbf{y}, \ldots$.
3. Každé dvojici bodů $X, Y \in \mathbb{R}^m$ je přiřazen jeden vektor $\mathbf{u} \overset{ozn}{=} Y - X \in \mathbb{E}^n$.
4. Pro každý bod $X \in \mathbb{R}^m$ a pro každý vektor $\mathbf{u} \in \mathbb{E}^n$ existuje právě jeden bod $Y \in \mathbb{R}^m$, že $Y = X + \mathbf{z}$ (všimneme si, že jsme právě zavedli sčítání bodů a vektorů).
5. je-li $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_n$ báze prostoru $\mathbb{E}^n$ a P je libovolný bod prostoru $\mathbb{R}^m$, potom skupinu $\{P, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_n\}$ nazýváme *afinní bází* prostoru $\mathbf{A}$ a bod P *počátek afinní báze*.
6. $\dim \mathbf{A} = \dim \mathbb{E}^n = n$.

Důsledky:

a) $X - Y = \mathbf{o}$,

b) $X - Y = -(Y - X)$,

c) $(X + \mathbf{u}) - Y = (X - Y) + \mathbf{u}$,

d) $X - (Y + \mathbf{u}) = (X - Y) - \mathbf{u}$,

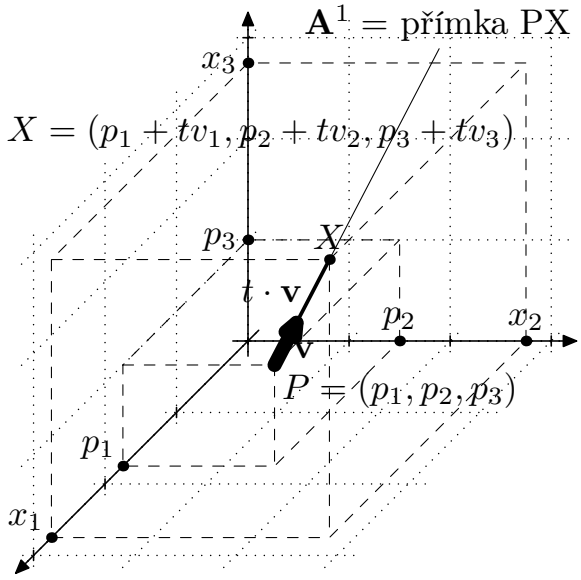
e) $(X + \mathbf{u}) + \mathbf{v} = X + (\mathbf{u} + \mathbf{v})$,

f) $(X - Y) + (Q - U) = (X - U) + (Q - Y)$,

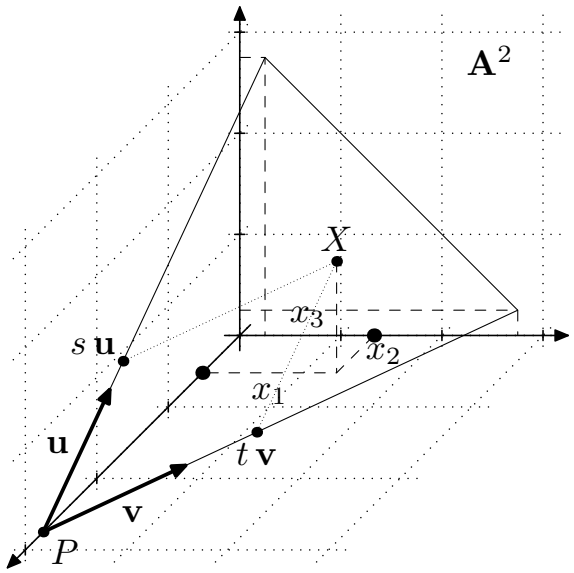
g) Každý bod $X \in \mathbb{R}^n$ lze vyjádřit ve tvaru $X = P + \mathbf{x} = P + \sum_{i=1}^{n} x_i \, \mathbf{e}_i$. Vektoru $\mathbf{x} \in \mathbb{E}^n$ se potom říká *průvodič* bodu X a číslům x_i *afinní souřadnice* bodu X.

Ilustrace:

1. Afinní prostor $\mathbf{A}^1 = \{\mathbb{R}^3, \mathbb{E}^1\}$ je množina bodů $X = P + t \cdot \mathbf{v}$, kde $X = (x_1, x_2, x_3)$, $P = (p_1, p_2, p_3)$, $\mathbf{v} \in \mathbb{E}^n$, tj. $x_1 = p_1 + t \cdot v_1$, $x_2 = p_2 + t \cdot v_2$ a $x_3 = p_3 + t \cdot v_3$. Jedná se tedy o přímku procházející bodem P ve směru $\mathbf{v}$, tzv. *parametrické vyjádření přímky* (viz obrázek 6.2.5 v $\mathbb{R}^3$).
2. Afinní prostor $\mathbf{A}^2 = \{\mathbb{R}^3, \mathbb{E}^2\}$ je množina bodů $X = P + t \cdot \mathbf{v} + s \cdot \mathbf{u}$, kde $X = (x_1, x_2, x_3)$, $P = (p_1, p_2, p_3)$, tj. $x_1 = p_1 + t \cdot v_1 + s \cdot u_1$, $x_2 = p_2 + t \cdot v_2 + s \cdot u_2$ a $x_3 = p_3 + t \cdot v_3 + s \cdot u_3$. Jedná se tedy o rovinu procházející bodem P určenou dvěma nezávislými bázovými vektory $\mathbf{u}$ a $\mathbf{v}$; tzv. *parametrické vyjádření roviny* (viz obrázek 6.2.6 v $\mathbb{R}^3$).



Obrázek 6.2.5: Afinní prostor $\mathbf{A}^1$



Obrázek 6.2.6: Afinní prostor $\mathbf{A}^2$

objekt	výklad	
bod,	Bod jest, co nemá dílu	Eukleides z Alexandrie
($\sigma\tau\iota\gamma\mu\alpha$,	Bod je to, co nemá délku (obsah, objem)	Překlady Eukleida
signum,	Prvek množiny $n-tic$ čísel (např $\mathbb{R}^n$)	Analytická geometrie
punctus,	Obraz komplexního čísla v Gaussově rovině	Teorie komplexních čísel
značka,	Obraz reálného čísla na souřadnicové ose	Souřadnicové systémy
puntík)	Alternativní název pro prvek libovolného prostoru	
	Body fázového prostoru určující všechny možné stavy soustavy hmotných bodů, do kterých se systém může dostat	
vektor	Prvek lineárního (vektorového) prostoru (např. $\mathbb{E}^n$)	
	Tenzor prvního řádu	
	Dvojice, trojice, $n-tice$ čísel	Aritmetický vektor
	Orientovaná úsečka, resp. množina rovnoběžných ,stejně dlouhých, orientovaných úseček	Analytický geometrie
	Veličina, která má velikost (počet jednotek) a směr	Fyzika
Přímka	Čára je délka bez šířky. Hranicemi čáry jsou body.	Eukleides z Alexandrie
	Úsečka je čára, která se svými body táhne rovně	
	Projektivní prostor dimenze nejméně 1 je takový, který obsahuje alespoň 2 body (a tedy přímku)	
	Prvek množiny všech řešení $n-1$ lineárních rovnic pro n neznámých.	

Tabulka 6.2.1: Názvoslovná tabulka

Kapitola 7
Bitové operace v jazyce C

Obsah kapitoly

Stále širší pronikání výpočetní techniky do všech oborů lidské činnosti přináší potřebu „mít někoho, kdo tomu rozumí“. Někoho, kdo poradí, pomůže, když něco přestane fungovat, něco se rozbije. Někoho, kdo třeba „jen“ poradí, když si s něčím nevíme rady.

Ten někdo by měl být technicky vzdělaný odborník v oboru počítačových, obecněji asi informačních, technologiích. Předpokládá se, že bude mít příslušné znalosti, větší, či menší hloubky, z oborů počítačových věd, matematiky, programování.

Programování je považováno za jednu z klíčových dovedností, mají-li být technické prostředky efektivně využívány. S tím souvisí velmi intenzivní rozvoj a výzkum a implementace funkcí operačních systémů, programovacích jazyků a kompilátorů a různé programovací techniky a postupy.

Dnes se programování standardně vyučuje na středních školách s tímto zaměřením. Často se používá k výuce programování[1] jazyk C, případně jeho „nástupci“. Didaktické postupy výuky programování jsou převážně zaměřeny na zvládnutí základních *syntaktických struktur*, případně *datových struktur* jazyka. Méně pak už bývá věnován prostor používaným *algoritmům*, jejich vlastnostem, principům funkce, či jejich tvorby, nebo úpravy.

[1] bohužel z didaktického pohledu výuky programování nepříliš vhodně

Což vede k mechanickému učení kterak poskládat jednotlivé příkazy jazyka, tvořit syntaktické struktury, pouze tak, aby kompilátor takový kód úspěšně přeložil. Bohužel, často bez pochopení jejich významu a tak i funkce programu.

Podobný přístup je možné spatřit i při používání jednotlivých typů proměnných, které jsou v jazyce dostupné. Mechanický přístup k návrhu datové struktury, nesprávné, nebo neúplné pochopení jednotlivých vlastností datových typů končí jejich nevhodným, či nesprávným použitím.

Tak jako v matematice pracujeme s objekty typu čísla, vektory, matice, funkce, body, množiny, výroky, tak, že s nimi z různých užitečných důvodů manipulujeme (sčítáme, skládáme, srovnáváme , atd.), tak potřebujeme manipulovat s informačními, datovými objekty jako jsou bity, bajty, slova .... Tyto objekty jsou závislé na konstrukci použitého technického vybavení počítače.

Manipulace musíme definovat tak, aby byly technicky realizovatelné. Poměrně častá je potřeba manipulací s jednotlivými signály (sadou signálů). Vyšší programovací jazyky obvykle nemají ve svém seznamu dostupných instrukcí a povelů („syntaktické výbavě“) takové možnosti manipulace.

Využitím vhodných matematických operací, jejich dobrému porozumění a vhodné aplikaci je možné tyto potřeby velmi efektivně naplnit.

Předložený článek se snaží ukázat užitečnost spojení znalostí matematických a technologických, jejich vzájemně výhodnou symbiózu. Předpokládá, že čtenář, žák střední školy, má přiměřené matematické znalosti a chápe pojmy „součet čísel“, případně „součin čísel“.

Také předpokládá, že získal základní informace ze světa programování a chápe a rozumí pojmům „příkaz“, „aritmetická operace“, „logická operace“. Zná pojmy „strukturovaná proměnná“, „skalární proměnná“, „logická proměnná“ a rozdíly mezi nimi. Nepředpokládá se znalost složitějších algoritmických struktur, případně datových struktur.

Je výhoda, bude-li čtenář mít podrobnější informace o způsobu uložení základních typů dat v paměti počítače, reprezentaci základních typů dat v paměti počítače a některé metody, způsoby manipulace s nimi, nicméně pro pochopení dalšího textu toto není nutné.

Snaží se ukázat, že matematika je schopna poskytnou univerzální metodu kterak pronikat do složitých problémů. Že je dobré, praktické, užitečné ovládnout „základní matematický výcvik“ a získat tak schopnost práce se symboly. Např. existence „prostorové představivosti“ (druh matematického myšlení) dovoluje znalci číst technické výkresy, nám dovolí lépe chápat taje programování.

7.1 Základní pojmy

Výuka programování se na vybraných středních školách[2] dnes ubírá směrem ke stále intenzivnějšímu používání jazyka *C* a jeho mnoha různých následovníků. Počínaje verzí jazyka *ANSI C* a konče dnes populárním jazykem *C#*.

[2] někdy i vybraných základních školách

Pro výuku algoritmizace a datových struktur se však zdá být vhodnější jazyk *Pascal* a jeho varianty (*Delphi, Lazarus, ...*), který byl právě k tomuto účelu „sestrojen“. Současný vývoj je však „tlačený“ poněkud jiným směrem, asi hlavně díky požadavkům požadavkům vysokých škol a komerční praxe.

Ukazuje se, že pokud daný (používaný) jazyk neobsahuje přímo ve své definici „build-in“ prostředky pro manipulaci s hodnotami jednotlivých bitů v bajtu (slovu, dvojslovu, ...) je pro středoškolské studenty poměrně obtížné se samostatnými bitovými hodnotami programově „manipulovat“. Následující text se pokusí tuto mezeru poněkud vyplnit.

Programových technik, kterými lze dosáhnout požadovaných efektů je samozřejmě celá řada. V textu uvedeme pravděpodobně nejpoužívanější z nich, *techniku maskování*. Ta je považována za obecnou techniku založenou na aplikaci zákonů nazývaných *Booleova algebra* a nezávisí tak na konkrétním programovacím jazyce.

Stručně vysvětleme základní pojmy.

7.1.1 Bit, bajt, slovo, ...

V souladu s obecně zažitou a používanou terminologií velmi stručně připomeňme uvedené významy, tak, jak se ustálilo jejich používání.

bit (bit): je označení *základní jednotky informace*. Termín *bit* vznikl z anglického označení ***b***inary dig***it*** (binární číslo) ve významu „drobet“, „kousek“. Je to tedy binární hodnota, hodnota ve dvojkové číselné soustavě a obsahuje tak pouze dvě možné hodnoty – hodnotu 0 a hodnotu 1.

Během rozvoje nepřeberného počtu elektrotechnických (a poté elektronických) oborů byla označována různými názvy, podle účelu použití tím kterým (elektrotechnickým, či elektronickým) oborem.

Hodnota „0“ (nula) bývá také označována termíny:

- vypnuto - logické (releové, kontaktní) řízení
- false - teoretická informatika, matematika
- „L“ - logické (číslicové) polovodičové obvody

Hodnota „1“ (jedna) bývá také označována termíny:

- zapnuto - logické (releové, kontaktní) řízení
- true - teoretická informatika, matematika
- „H“ - logické (číslicové) polovodičové obvody

Svět programování kupodivu používá většinu zmíněných označení. Není tak problém nalézt v jednom odborném textu současně použité symboly „0 (1)“, „L (H)“, i „true (false)“. Je to pravděpodobně proto, že výpočetní systémy jsou založeny na (logických) číslicových obvodech (symboly L,H) a většina významných teoretických informatiků (symboly true,false) se rekrutovala z významných matematiků (symboly 0,1).

bajt (byte): je dnes označení skupiny obvykle 8 bitů. Nebylo tomu tak ale „odjakživa“. V roce 1956 zavedl tento termín *Werner Buchholz*, když pracoval s počítačem *IBM Stretch*. Zpočátku tento termín popisoval skupinu 1 až 6 bitů. Přechod na osmibitový bajt nastal o něco později. Časem se osmibitový bajt stal standardem pro počítač *System/360*. Ukázalo se, že osmice bitů *bajt* je optimální volba mezi počtem bitů a (tehdy) užitečné velikosti čísla, které se „do něj vejde“. Jeho popularita pak vedla k tomu, že osmibitový bajt je dnes standardem.

Slovo *byte* pochází z anglického slova *bite*, značící „sousto“, tzn. nejmenší objem dat, který počítač dokáže „přechroustat“. Původní slovo *bite* pak bylo upraveno na tvar *byte*, aby se předešlo záměně se slovem *bit*. Bajt je nedělitelnou informační jednotkou ve vyšších programovacích jazycích. V těch je deklarován datový typ ***byte***, případně jiný „kompatibilní“ datový typ (datový typ ***char*** v jazyce C[3]). Tyto jazyky pak „nabízejí“ (umožňují) s bajtem provádět jak aritmetické operace, tak operace logické. Rozsah hodnot 0 až 255 ($255 = 2^8 - 1$), které lze uložit do proměnné typu ***byte*** je poměrně omezený, byť pro řadu úloh ze školní praxe[4] může být dostatečný.

slovo (word): je skupina obvykle 2 bajtů. Pokud je takto označena (deklarována), je pak považována za dále nedělitelnou entitu. Vyšší programovací jazyky ji obvykle označují termínem *integer*[5]. Pracuje se s ní podobně jako s bajtem s tím, že je možné do takové proměnné uložit mnohem větší rozsah hodnot 0 až 65535 ($65535 = 2^{16} - 1$).

dvojslovo (doubleword): je obvykle skupina 4 bajtů (32 bitů: 4×8). Jde v podstatě o „extenzivní rozšíření“ pojmu *integer*. Je zřejmé, že do proměnné délky 32 bitů (4 bajty) je možné uložit podstatně větší rozsah hodnot 0 až 4294967295 ($4294967295 = 2^{32} - 1$).

čtyřslovo (quadword): je skupina 8 bajtů (64 bitů). Tento termín se příliš nepoužívá, spíše je ze světa programátorů rozšířen pojem *long int*[6]. Je zřejmé, že jde o pouhé extenzivní rozšíření předchozích úvah...

Poznámky:

1. Konkrétní označení *byte*, *int*, případně *long int* může být implementačně závislé na použité verzi zvoleného operačního systému (Windows, MS-DOS, Linux,...), případně na druhu a typu zvoleného jazyka (Ansi C, C#, ...). Stejně tak i počet bitů pro označení *int*, případně *long int* může být závislý na zvolené technické platformě (16 bitové versus 64 bitové mikroprocesory).

2. V literatuře lze nalézt alternativní význam termínu *slovo*. Často bývá odvozené od hardware, technických vlastností používaného (mikro)procesoru a označuje *paměťovou jednotku*.

[3] ANSI C norma jazyka
[4] komerční praxe také
[5] znaménkový, bezznaménkový, podle potřeb a možností dané implementace jazyka
[6] skoro všechny...

Podle typu použitého mikroprocesoru tak pracujeme s paměťovými jednotkami, *slovy* o délce 16 bitů, je-li použitý systém 16bitový, případně se slovy o délce 32 bitů, je-li použitý HW[7] 32bitový.

Dnes jsou nejčastěji používané systémy 64bitové, takže pracujeme s paměťovými jednotkami, *slovy* o délce 64 bitů. Tím termíny *dvojslovo (doubleword)* a *čtyřslovo (quadword)* pozbývají svůj původní význam.

7.1.2 Aritmetické operace

V matematice jsou „známy“ *aritmetické operace*[8] sčítání, odčítání, násobení, dělení, umocňování. Veškeré programové jazyky, nižší i vyšší, implementují tyto operace definitoricky[9].

Téměř všechny programovací jazyky pracují s bajtem, případně skupinou bajtů (slovo, dvojslovo, ...) jako s elementární entitou, dále nedělitelnou.

Jinými slovy, skupina číslic je považována za číslo jehož hodnota je určena právě jednotlivými číslicemi (ciframi) a jejich pozicí v čísle. Je lhostejné, zda to jsou číslice 0 až 9 v dekadické číselné soustavě, nebo číslice 0 a 1 ve dvojkové číselné soustavě.

13 + 15 = 28
13 * 15 = 195

⇔

dekadická hodnota		**binární hodnota**							
1	3	0	0	0	0	1	1	0	1
+ 1	5	0	0	0	0	1	1	1	1
= 2	8	0	0	0	1	1	1	0	0
1	3	0	0	0	0	1	1	0	1
* 1	5	0	0	0	0	1	1	1	1
= 1 9	5	1	1	0	0	0	0	1	1

Tabulka 7.1.1: Aritmetické operace v dekadické a binární číselné soustavě

Aritmetické operace se v počítači vykonávají podle stejných principů, jako v „normální aritmetice“, stejně v desítkové i dvojkové číselné soustavě. Až na to, že výsledek operací je „upraven“ na menší počet cifer. V počítači nemáme tolik paměti pro uložení přesného výsledku, viz ***** odkaz na sešit o počítačových číslech

Připomeňme jen, že pracujeme s pozičními číselnými soustavami, kde význam, vliv, číslice (cifry) v čísle je určen jeho pozicí v něm. Platí, že čím více je číslice v čísle „umístěna“ více vlevo, tím větší vliv na výslednou hodnotu má.

Operace s čísly ve dvojkové (binární) soustavě nejsou příliš „user friendly“. Přeci jenom, jsme spíše zvyklí používat desítkovou číselnou soustavu.

7.1.3 Logické operace

Logické operace jsou definovány na základě pravidel formální (matematické) logiky. Binární operace operují se dvěma výroky a jsou to:

[7] asi celkem zažitá zkratka pro „hardware“, technické vybavení. Podobně označení SW je asi zažitá zkratka pro „software“, programové vybavení...

[8] také množinové operace a logické operace, případně jiné.

[9] v rámci množiny „počítačových čísel“

operace:	konjunkce	disjunkce	implikace	ekvivalence
označení:	$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$

Unární operace *negace* operuje s jedním výrokem a je to:

$$\text{negace: } \neg$$

Operace *Booleovy logiky* jsou definovány na množině hodnot $\{0,1\}$. Zabývají se (pracují) s operacemi $\wedge$ (logický součin), $\vee$ (logický součet) a $\neg$ (negace). Jejím rozšířením je pak *Booleova algebra*, která zobecňuje *množinové* a *logické* operace.

Například:

- konjunkci výroků V_1 a V_2 interpretujeme jako logický (Booleovský) součin,
- disjunkci výroků V_1 a V_2 interpretujeme jako logický (Booleovský) součet.

Připomeňme jen, že jeden obor elektrotechniky, číslicová technika, si „zvolil“[10] formální označení *H - High* označující vyšší napětí a *L - Low* označující nižší napětí v číslicových elektronických obvodech. Tyto symboly označují vnitřní stavy číslicových elektronických systému vyjádřené určitou úrovní napětí.

Z těchto důvodů se v technické odborné literatuře číslicové techniky, programování, teoretické informatiky, ... často považují za ekvivalentní označení:

$$\text{pravda} \Leftrightarrow \text{true} \Leftrightarrow 1 \Leftrightarrow \text{H}\,,$$

případně:

$$\text{nepravda} \Leftrightarrow \text{false} \Leftrightarrow 0 \Leftrightarrow \text{L}\,.$$

Z praktických, úsporných důvodů se velmi často (nejčastěji) k zápisu logických operací i mimo oblast číslicové techniky používají symboly:

$$\text{L, (0)}\,, \quad \text{příp. H (1)}\,.$$

Připomeňme základní logické operace, tak jak jsou definovány *Booleovou algebrou* pro dvouprvkovou množinu $\{0,1\}$. Jejich znázornění lze nalézt v tabulce (7.1.3).

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

(a) Konjunkce

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

(b) Disjunkce

A	B	Q
0	0	1
0	1	1
1	0	0
1	1	1

(c) Implikace

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

(d) Ekvivalence

Tabulka 7.1.3: Binární logické operace

Doplňme i unární logický operátor, *negaci*. Znázornění je tabulce 7.1.4.

[10] ze svých interních, ryze praktických důvodů

A	Q
0	1
1	0

Tabulka 7.1.4: Unární logická operace - negace

7.1.4 Bitové operace

Termíny *bitové operace*, případně *bitové operátory* jsou ze světa programátorů, výpočetní a číslicové techniky. Elektronické obvody (mikro)procesorů jsou sestrojeny tak, že „umí“ tyto operace provést. Přivedeme-li na jejich vstup(y) odpovídající napětí (úrovně L, příp. H), získáme výstupní napětí z těchto obvodů, které odpovídá implementované logické funkci.

Tak, jako jejich matematické vzory, *logické operace*, pracují s *bitovými hodnotami*, viz odst. 7.1.1.

Téměř všechny programovací jazyky implementují bitové operátory *logický součin* (konjunkce), *logický součet* (disjunkce) a *negace* (negace). Z ryze praktických důvodů implementují ještě další logickou operaci, *nonekvivalence* (exkluzivní disjunkce).

Za povšimnutí stojí fakt, že počítače (jejich mikroprocesory) neimplementují operace *implikace* a *ekvivalence*. Na základě *De Morganových zákonů* formální logiky lze:

$$\text{implikaci } A \Rightarrow B \text{ ekvivalentně nahradit disjunkcí } \neg A \vee B$$

nebo konjunkcí:

$$\neg(A \wedge \neg B)$$

případně:

$$\text{ekvivalenci } A \Leftrightarrow B \text{ ekvivalentně nahradit konjunkcí } (A \vee \neg B) \wedge (\neg A \vee B)$$

Připomeňme obvyklé značky (symboly) používané k zápisu logických operací v programátorské praxi, případně v praxi konstruktéra číslicových obvodů. Je užitečné uvést také symboly používané (implementované) v jazyce C (C++, případně C#).

	matematika	číslicová technika	jazyk C	
konjunkce	&, $\wedge$, AND	AND	&	&&
disjunkce	$\vee$, OR	OR	|	||
negace	$\neg$, NON	NOT	$\sim$	!
vylučovací disjunkce		XOR, $\oplus$	$\widehat{}$	

Tabulka 7.1.5: Symboly a značky pro bitové operace

Poznámka:
Jazyk C „rozlišuje“ termíny *bitový operátor* a *logický operátor*. Logické operátory jsou aplikované na logické hodnoty „pravda“, příp. „nepravda“. Obvykle se používají ke zřetězení logických operací jejich výsledkem je logická hodnota. Například:

$$((a == b)\&\&(c >= b))||(d\,!=\,e).$$

Lze pozorovat, že uvedený příklad je fragment kódu jazyka C zapsaný programátorem. Matematik by spíše zapsal:

$$(a = b) \wedge (c \geq b) \vee (d \neq e).$$

Věnujme se nyní bitové operaci *bitový součin*. Nyní budeme proměnné A a B považovat za *bitové proměnné*, které mohou nabývat hodnot pouze 0 a 1. Operace bitový součin:

$$Q = A \text{ AND } B \quad \text{případně} \quad Q = A \,\&\, B$$

bude mít výsledek:

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

$\Leftrightarrow$

A	B	Q
L	L	L
L	H	L
H	L	L
H	H	H

Tabulka 7.1.6: Operace - bitový součin

Pozorujeme následující vlastnosti:

Je-li logická proměnná A nastavena na hodnotu „L“ (0), je patrné, že „výstupní“ hodnota Q, *není* hodnotou proměnné B (signálu B) nijak ovlivněna. Signál B se ve výstupu neuplatní.
Říkáme, že je **zamaskován** hodnotou signálu A – *maskou*. Ta „překryje“, *zamaskuje* vliv signálu B na výstupní signál, který tak vždy zůstane nastaven na úroveň „L“ (0).

Podobnou úvahu nyní provedeme i pro bitovou operaci *bitový součet*. Stále budeme proměnné A a B považovat za bitové proměnné, které mohou nabývat hodnot pouze 0 a 1. Operace bitový součet:

$$Q = A \text{ OR } B \quad \text{případně} \quad Q = A \mid B$$

bude mít výsledek:

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

$\Leftrightarrow$

A	B	Q
L	L	L
L	H	H
H	L	H
H	H	H

Tabulka 7.1.7: Operace - bitový součet

Také nyní pozorujeme relevantní vlastnosti:

Je-li logická proměnná A nastavena na hodnotu „H“ (1), je patrné, že ve „výstupní“ hodnota Q *není* hodnotou proměnné B (signálu B) nijak ovlivněna. Signál B se ve výsledku neuplatní.
Říkáme, že je i nyní ***zamaskován*** hodnotou signálu A – *maskou*. Vždy zůstane nastaven na úroveň „H“ (1).

V odstavci 7.1.1 jsme nastínili, že bajt (slovo, ...) je skupina bitů označená tímto termínem. Můžeme tedy na bajt (slovo, ...) také nahlížet jako na určitou skupinu jednotlivých bitů. Podle okolností[11] to může být skupina *jednobitových signálů*, které mohou být mezi sebou v podstatě nezávislé.

Máme tedy skupinu *nezávislých signálů*. Označme je $B = \{B_0, B_1, \ldots, B_7\}$. Můžeme najít (sestavit, vypočítat) druhou skupinu signálů. Ty označme $A = \{A_0, A_1, \ldots, A_7\}$.

Využijeme-li analogie s tabulkou 7.1.6, případně tabulkou 7.1.7, je patrné, že můžeme na příslušné dvojice signálů aplikovat stejné závěry. A to vždy na *stejnolehlé dvojice* bitů (signálů) $A_0 \leftrightarrow B_0, A_1 \leftrightarrow B_1, \ldots, A_7 \leftrightarrow B_7$, Viz příklad v tabulce 7.1.9:

155	1	0	0	1	1	0	1	1
∨, OR	↕	↕	↕	↕	↕	↕	↕	↕
233	1	1	1	0	1	0	0	1
	↓	↓	↓	↓	↓	↓	↓	↓
251	1	1	1	1	1	0	1	1

(a) Bitový součet stejnolehlých bitů

155	1	0	0	1	1	0	1	1
∧, AND	↕	↕	↕	↕	↕	↕	↕	↕
233	1	1	1	0	1	0	0	1
	↓	↓	↓	↓	↓	↓	↓	↓
137	1	0	0	0	1	0	0	1

(b) Bitový součin stejnolehlých bitů

Tabulka 7.1.9: Bitové operace pro vícebitové operandy

Z tabulky 7.1.9 je patrné, že stejnolehlé bitové operace se provádějí **nezávisle**. Bitová operace prováděná s dvojicí bitů (operandů) není nijak ovlivněna výsledkem logické operace „pravé dvojice“ operandů a nijak neovlivňují průběh zpracování „levé dvojice“ operandů.

Těchto vlastností využijeme v technice *maskování*, která je velmi užitečná v každodenní programátorské praxi. Bližší v kapitole 7.2.

7.2 Technika maskování

Technika maskování je určitý postup založený na aplikaci logických operacích dostupných i pro vyšší programovací jazyky. Je to vhodná technika v případě, že daný (zvolený) programovací jazyk nepodporuje práci s jednotlivými bity, nemá pro tyto operace instrukce, v rámci vícebitových operandů (proměnných).

Je to programový postup, založený na *matematické logice* jak poměrně efektivně zjistit stav libovolného vstupního bitu (čteného z „okolního světa“) ve vícebitovém operandu typu *byte, int, long int, ...*

Dovolí také nastavit libovolný výstupní bit na požadovanou hodnotu (zapisovaný do „okolního světa“) ve vícebitovém operandu bez nežádoucího ovlivňování hodnot ostatních bitů operandu.

Je proto považována za poměrně obecnou programovou techniku.

[11] obvykle určených technologickými okolnostmi

7.3 Zjištění vstupní hodnoty

Jako aplikační programátoři často máme za úkol zjistit stav nějaké bitové hodnoty „ukryté“ ve vícebitovém operandu. Poměrně přímočarý postup, který by nás pravděpodobně napadl jako první, je definovat požadované činnosti pro všechny možné kombinace vstupních signálů.

Všimneme si, že vlastně považujeme přečtenou hodnotu daného vstupu za *číslo* ve smyslu odstavce 7.1.2.

Zdá se, že pro relativně malý počet (vstupních) bitů by tento přístup mohl poskytnout poměrně dobře fungující metodu. Bohužel, již první jednoduchá analýza tohoto přístupu ale odhalí jeho závažné nedostatky. Osmibitový operand (bajt) „dovolí“, obsahuje 256 různých hodnot. Má-li být náš program schopen reagovat na libovolnou kombinaci vstupních signálů, je zřejmé, že bychom museli naprogramovat právě 256 různých variant reakcí, různých programových situací.

Budeme-li považovat jednotlivé bity operandu (bajtu) za *nezávislé* signály, viz odstavec 7.1.4, zredukujeme potřebný počet programových reakcí na 8!

Kolik různých „situací“ lze získat, bude-li počet vstupních (jednobitových) signálů například 24 lze velmi snadno spočítat ($2^{24} \approx 16777216$). Získaný počet možných situací je v podstatě programově nezvládnutelný.

Prakticky je téměř nezvládnutelný i mnohem menší počet vstupních signálů (bajt). Je totiž nutné si uvědomit, že řada dílčích programových konstrukcí se bude velmi pravděpodobně vícenásobně opakovat v jednotlivých variantách „reakcí“ na určitou kombinaci vstupních signálů. Což je sice nežádoucí, nicméně ideální zdroj chyb a překlepů. Tato „vlastnost“ je tak noční můra všech programátorů.

Technika maskování, založená na bitových operacích, (viz odstavec 7.1.3) poskytuje onu efektivní, snadno „uchopitelnou[12]“ techniku, která nás „ochrání“ před zjištěnými nežádoucími vlastnostmi.

Lze použít dva navzájem komplementární postupy, jak získat požadovaný výsledek. Oba postupy jsou ekvivalentní, nelze říci, který je lepší, či horší. Volba dané metody, postupu, je tak spíše určena konkrétní konstrukcí použitých periferních zařízení („zbytku světa“).

7.3.1 Aktivní vstupní úroveň „L“

Mějme řízenou periferii, která je konstruována tak, že aktivní logické úrovně, které „produkuje“ jsou „L“. Je patrné, že to je *výstupní signál* z pohledu periferie. Jestliže z periferie tento signál „vystupuje“ musí zákonitě „vstupovat“ do počítače. Z pohledu našeho řídícího programu jde tedy o *vstupní signál*.

Budeme tedy používat (domluvenou) konvenci, že vstupní signály budeme posuzovat z pohledu počítače. Signály *vstupující* do našeho procesu zpracování nazveme *vstupní signály*. Jeví se mírně výhodnější použít postup pracovně nazývaný *detekce „L“*.

Předpokládejme tedy, že vhodnou instrukcí „čteme“ bajtovou vstupní hodnotu ze vstupních obvodů počítače. Ta obsahuje 8 nezávislých jednobitových signálů. Jejich

[12] čti poměrně srozumitelnou a snadno naučitelnou techniku...

význam je určen konkrétním schematem zapojení (konstrukcí systému). Dále předpokládejme, že nás bude zajímat jednobitový signál číslo d_4, viz tabulka (7.3.10). Na hodnotách ostatních bitových signálů teď nezáleží, nejsou důležité, nezajímají nás.

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	1	1	0	0	1	1	1

Tabulka 7.3.10: Bitové signály

Při bližším pohledu shledáme, že „odstínění“ vlivu, *zamaskování*, ostatních bitových hodnot v bajtu lze provést dvěma způsoby.

- nastavením ostatních (nesledovaných) bitů na „H“
- nastavením ostatních (nesledovaných) bitů na „L“

Vyzkoušejme nejprve jaký bude výsledek v případě, že se rozhodneme bity, jejichž hodnota nás nyní nezajímá, nastavit, *zamaskovat* na hodnotu „H“. Jak toho dosáhneme? Využijeme závěry odstavce (7.1.3).

Klíč k úspěchu spočívá ve vhodné volbě (a aplikaci) „druhého operandu“ v bitové operaci. Budeme ho nazývat termínem *maska*. A bude sloužit právě k odstínění, *zamaskování* těch signálů uložených v přečteném bajtu, jejich hodnota nás nyní nezajímá.

Je zřejmé, že hodnota masky, která zamaskuje všechny ostatní (nesledované) signály bude mít hodnotu, která je uvedena v tabulce (7.3.11). Stále chceme zjistit stav signálu d_4.

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
1	1	1	0	1	1	1	1

Tabulka 7.3.11: Maska pro signál d_4

Je také patrné, že má-li technika maskování správně pracovat, je nutné použít (aplikovat) „správnou“ bitovou operaci. V tomto případě aplikujeme operaci *bitový součet*, označovaný symbolem ($\vee$). Budeme-li aplikovat masku z tabulky (7.3.11) na bitové signály z tabulky (7.3.10), je zřejmé, že získáme výsledek uvedený v tabulce (7.3.13) pro aktuální hodnotu sledovaného signálu d_4.

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	1	0	1	1	1
$\vee$	↕	↕	↕	↕	↕	↕	↕	↕
	1	1	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
	1	1	1	1	1	1	1	1

(a) výsledek maskování pro $d_4 = 1$

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	0	0	1	1	1
$\vee$	↕	↕	↕	↕	↕	↕	↕	↕
	1	1	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
	1	1	1	0	1	1	1	1

(b) výsledek maskování pro $d_4 = 0$

Tabulka 7.3.13: Maskování na „H“

Z tabulky (7.3.13) je tak patrné, že bude-li mít signál d_4 **neaktivní** úroveň („H“), bude výsledek *techniky maskování* hodnota $0xFF$. V opačném případě bude výsledek *techniky maskování* jiný.

Lze také nahlédnout, že hodnotu $0xFF$ získáme pro jakýkoliv jiný sledovaný **neaktivní** signál. Což je poměrně důležitý a významný závěr.

Plyne z něj, že sestrojíme-li pro libovolný bitový signál vhodnou masku, vždy jednoznačně získáme informaci o hledané úrovni sledovaného signálu. A to vždy stejnou výpočetní operací.

V tabulce (7.3.14) je příklad programu (fragment kódu), který by mohl popisované činnosti provést. Význam a použití jednotlivých proměnných plyne z jejich mnemotechnických názvů.

```
...
unsigned char inPort; // bajtová hodnota
const int adrPortIn = 0x301;
const unsigned char maskD4H = 0xEF;
const unsigned char nonActiveSignal = 0xFF;
...
inPort = inportbyte(adrPortIn);
sigD4 = inPort | maskD4H;
if (sigD4 == nonActiveSignal)
  { akce pro neaktivní signál} // test is TRUE
else
  { akce pro aktivní signál} // test is FALSE
...
```

Tabulka 7.3.14: Fragment kódu pro maskování na „H“

Nyní prozkoumejme variantu, kdy se rozhodneme nesledované signály (bity) nastavit, *zamaskovat* na hodnotu „L“. Zvolíme takovou masku, abychom tohoto dosáhli, viz tabulka (7.3.15).

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	0	0	1	0	0	0	0

Tabulka 7.3.15: Maska pro signál d_4

Mělo by již být zřejmé, že pro správné fungování našich úvah musíme použít logickou operaci *logický součin* označovaný symbolem $\wedge$. Budeme-li tedy aplikovat masku z tabulky (7.3.15), získáme výsledky uvedené v tabulce (7.3.17).

Z tabulky (7.3.17) je patrné, že bude-li mít signál d_4 **aktivní** úroveň („L“), bude výsledek *techniky maskování* hodnota $0x00$. V opačném případě bude hodnota jiná. Lze také nahlédnout, že hodnotu $0x00$ získáme jakýkoliv sledovaný **aktivní** signál v kombinaci s vhodnou hodnotou masky.

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	1	0	1	1	1
$\wedge$	↕	↕	↕	↕	↕	↕	↕	↕
	0	0	0	1	0	0	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	0	1	0	0	0	0

(a) výsledek maskování pro $d_4 = 1$

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	0	0	1	1	1
$\wedge$	↕	↕	↕	↕	↕	↕	↕	↕
	0	0	0	1	0	0	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	0	0	0	0	0	0

(b) výsledek maskování pro $d_4 = 0$

Tabulka 7.3.17: Maskování na „L“

Fragment programového kódu je uveden v tabulce (7.3.18). Lze všimnou změn příslušných hodnot použitých masek a logických operací v jeho struktuře ve srovnání s fragmentem kódu uvedeným v tabulce 7.3.14.

```
...
unsigned char inPort; // bajtová hodnota
const int adrPortIn = 0x301;
const unsigned char maskD4H = 0x10;
const unsigned char activeSignal = 0x00;
...
inPort = inportbyte(adrPortIn);
sigD4 = inPort & maskD4H;
if (sigD4 == activeSignal)
    { akce pro aktivní signál} // test is TRUE
else
    { akce pro neaktivní signál} // test is FALSE
...
```

Tabulka 7.3.18: Fragment kódu pro maskování na „L“

Závěr:
Porovnáme-li výsledky získané v případě volby techniky maskování na úroveň „H“ s právě získanými výsledky techniky maskování na úroveň „L“, zjistíme, že výsledky jsou vlastně ekvivalentní. Záleží tedy na naší volbě, který přístup zvolíme. Oba dva přístupy jsou si rovnocenné.

Zdá se, že z historických důvodů se u programátorské veřejnosti poněkud více „ujal“ přístup maskování na hodnotu „L“. Lze se pouze domýšlet proč tomu tak je. Pravděpodobně je to způsobeno formální podobností aktivity sledovaného signálu (aktivní úroveň „L“) s výsledkem techniky maskování na hodnotu „L“ (výsledek maskování je hodnota $0x00$).

Je proto doporučováno preferovat tento způsob aplikace *techniky maskování* i když z formálního hlediska jsou oba způsoby ekvivalentní.

7.3.2 Aktivní vstupní úroveň „H“

Mějme nyní řízenou periferii, která je konstruována tak, že aktivní logické úrovně, které „produkuje“ jsou „H“. Není to sice častým zvykem, nicméně existují určité, opodstatněné situace, kdy je vhodné (nutné) aplikovat právě toto konstrukční řešení.

Předpokládejme tedy, že opět „čteme“ bajtovou vstupní hodnotu. Ta zase obsahuje 8 nezávislých jednobitových signálů. Jejich význam je stále určen konkrétním schematem zapojení.

Dále předpokládejme, že nás stále bude zajímat signál číslo d_4, viz tabulka (7.3.19).

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	1	1	0	0	1	1	1

Tabulka 7.3.19: Bitové signály

Vidíme, že hodnota vstupního bajtu (8 nezávislých signálů) je stejná, jako v případě odstavce (7.3.1). Zkušenost, kterou již máme z odstavce (7.3.1) nám praví, že postupy asi budou podobné. Také v tomto případě je možné „odstínit“ ostatní hodnoty v bajtu dvěma způsoby.

- nastavením ostatních bitů na „H“
- nastavením ostatních bitů na „L“

Využijeme-li výsledky již získané v odstavci (7.3.1), můžeme dojít k praktickým závěrům rychleji.

Prostudujme nyní variantu, kdy se rozhodneme maskovat nesledované bity na hodnotu „H“. Z tabulky (7.3.20) je patrné, že volíme takovou masku, kdy aplikace „správné“ bitové operace povede k žádanému cíli, podobně jako v odstavci (7.3.1)

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
1	1	1	0	1	1	1	1

Tabulka 7.3.20: Maska pro signál d_4

Nyní to je operace *bitový součet*, označovaná symbolem ($\vee$). Budeme-li aplikovat masku z tabulky (7.3.20), získáme výsledek uvedený v tabulce(7.3.22).

Z tabulky (7.3.22) je také patrné, že bude-li mít signál d_4 **aktivní** úroveň („H“), bude výsledek *techniky maskování* hodnota $0xFF$. V opačném případě bude výsledek *techniky maskování* jiný.

Lze také nahlédnout, že hodnotu $0xFF$ získáme pro jakýkoliv jiný sledovaný **aktivní** signál. Což je opět poměrně důležitý a významný závěr.

Porovnáme-li toto zjištění s obdobným zjištěním příslušné části odstavce (7.3.1), lze konstatovat, ze formální vzhled i použitý postup jsou **totožné**. Odlišnost spočívá pouze ve **významu**, interpretaci výsledku získaného *operace maskování*. Ten je logicky navzájem komplementární (opačný).

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	1	0	1	1	1
$\vee$	↕	↕	↕	↕	↕	↕	↕	↕
	1	1	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
	1	1	1	1	1	1	1	1

(a) výsledek maskování pro $d_4 = 1$

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	0	0	1	1	1
$\vee$	↕	↕	↕	↕	↕	↕	↕	↕
	1	1	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
	1	1	1	0	1	1	1	1

(b) výsledek maskování pro $d_4 = 0$

Tabulka 7.3.22: Maskování na „H“

Nyní prozkoumejme variantu, kdy se rozhodneme nesledované signály (bity) nastavit, zamaskovat na hodnotu „L“. Zvolíme tedy takovou masku tak, abychom tohoto dosáhli, viz tabulka (7.3.23).

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	0	0	1	0	0	0	0

Tabulka 7.3.23: Maska pro signál d_4

Mělo by již být zřejmé, že správné fungování našich úvah musíme použít logickou operaci *bitový součin* označovaný symbolem $\wedge$. Budeme-li tedy aplikovat masku z tabulky (7.3.23), získáme výsledky uvedené v tabulce (7.3.25).

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	1	0	1	1	1
$\wedge$	↕	↕	↕	↕	↕	↕	↕	↕
	0	0	0	1	0	0	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	0	1	0	0	0	0

(a) výsledek maskování pro $d_4 = 1$

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	0	0	1	1	1
$\wedge$	↕	↕	↕	↕	↕	↕	↕	↕
	0	0	0	1	0	0	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	0	0	0	0	0	0

(b) výsledek maskování pro $d_4 = 0$

Tabulka 7.3.25: Maskování na „L“

Z tabulky (7.3.25) je patrné, že bude-li mít signál d_4 **neaktivní** úroveň („L“), bude výsledek *techniky maskování* hodnota $0x00$. V opačném případě bude hodnota jiná. Lze také nahlédnout, že hodnotu $0x00$ získáme jakýkoliv sledovaný **neaktivní** signál.

Všímavý čtenář patrně zaregistroval absenci příkladů programového řešení v odstavci (7.3.2). Je to proto, že případné fragmenty programového kódu by byly v podstatě identické s fragmenty kódu uvedenými v odstavci (7.3.2).

Mnohem důležitější pro pochopení a následné použití je správné porozumění rozdílům porovnávaných postupů. Špatné „uchopení“, interpretace významu jednotlivých výsledků vede na dosti obtížně odhalitelné sémantické chyby.

Porovnáme-li výsledky získané v případě volby techniky maskování na úroveň „H“ s právě získanými výsledky techniky maskování na úroveň „L“, zjistíme, že výsledky jsou vlastně ekvivalentní. Záleží tedy na naší volbě, který přístup zvolíme. Oba dva přístupy jsou ekvivalentní.

Zopakujme ještě jednou, zdá se, že z historických důvodů se více „ujal“ přístup maskování na hodnotu „L“. Je to pravděpodobně způsobeno formální podobností aktivity sledovaného signálu (aktivní úroveň „L“) s výsledkem techniky maskování na hodnotu „L“ (výsledek maskován je hodnota $0x00$). Je proto doporučeno preferovat tento způsob aplikace *techniky maskování* i když z logického hlediska jsou oba způsoby ekvivalentní.

7.3.3 Závěr

Všímavý čtenář patrně zaregistroval absenci příkladů programového řešení v odstavci (7.3.2). Je to proto, že případné fragmenty programového kódu by byly v podstatě identické s fragmenty kódu uvedenými v odstavci (7.3.1).

Mnohem důležitější pro pochopení a následné použití je správné porozumění rozdílům porovnávaných postupů. Špatné „uchopení“, špatná interpretace významu jednotlivých výsledků vede k dosti obtížně odhalitelným sémantickým chybám.

Porovnáme-li výsledky získané v případě volby techniky maskování na úroveň „H“ s právě získanými výsledky techniky maskování na úroveň „L“, zjistíme, že výsledky jsou vlastně ekvivalentní. Záleží tedy na naší volbě, který přístup zvolíme. Oba dva přístupy jsou si rovnocenné.

Zopakujme ještě jednou, zdá se, že z historických důvodů se více „ujal“ přístup maskování na hodnotu „L“. Je to pravděpodobně způsobeno formální podobností aktivity sledovaného signálu (aktivní úroveň „L“) s výsledkem techniky maskování na hodnotu „L“ (výsledek maskován je hodnota $0x00$). Je proto doporučeno preferovat tento způsob aplikace *techniky maskování* i když z formálního hlediska jsou oba způsoby ekvivalentní.

7.4 Nastavení výstupní hodnoty

Máme za úkol zajistit nastavení daného jednobitového signálu „ukrytého“ ve vícebitovém operandu. Jednoduchý, přímočarý postup, který by nás pravděpodobně napadl jako první, je všelijak, nejspíše dosti složitě, kombinovat stávající hodnotu výstupní kombinace bitů s požadovanou hodnotou. Což je složité, nepřehledné, neefektivní.

Nicméně, je zřejmé, že pro relativně malý počet výstupních bitů by tento přístup mohl poskytnout poměrně dobře fungující metodu. Bohužel, již první jednoduchá analýza tohoto přístupu ale odhalí jeho závažné nedostatky. Výsledky této analýzy budou totožné se zjištěními z odstavce (7.3).

Také v tomto případě poskytuje technika maskování, založená na logických operacích, (viz odstavec 7.1.3) efektivní, snadno „uchopitelnou“ techniku. Nastavení různých hodnot („L“, případně „H“) je sice poněkud složitější, ale stále velmi užitečné a přehledné.

7.4.1 Nastavení výstupní hodnoty „L“

Mějme řízenou periferii, která je konstruována tak, že aktivní logické úrovně, které „spotřebovává“ jsou „L“. Je patrné, že to je *vstupní signál* z pohledu periferie. Jestliže

do periferie tento signál „vstupuje“ musí zákonitě „vystupovat“ do počítače. Z pohledu našeho řídícího programu jde tedy o *výstupní signál*.

Předpokládejme nyní, že „zapisujeme“ bajtovou výstupní hodnotu „do periferie“. Přesněji, do výstupních obvodů počítače spojených soustavou 8 vodičů s periferií. Ta obsahuje 8 nezávislých jednobitových signálů. Jejich význam je určen konkrétním schematem zapojením.

Dále předpokládejme, že nás bude zajímat signál číslo d_4, viz tabulka (7.4.26), který chceme nastavit na úroveň „L“ tak, aby ostatní výstupní signály zůstaly **nezměněny**.

Tento požadavek je velmi důležitý, přímo kritický. Nežádoucí změna jiného výstupního signálu může způsobit velmi nežádoucí reakci řízené technologie. Například nevhodným uvedením do pohybu některé strojní části technologie můžeme někoho i zabít!

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	1	1	1	0	1	1	1

Tabulka 7.4.26: Bitové signály

Již máme určité zkušenosti s *technikou maskování* pro zjištění stavu sledovaného vstupního signálu z odstavců (7.3.1) a (7.3.2). Jak ji využijeme v případě potřeby nastavit jednobitový výstupní signál? Opět využijeme závěry odstavce (7.1.3).

Klíč k úspěchu spočívá ve vhodné volbě (a aplikaci) „druhého operandu“ v bitové operaci. Opět ho budeme nazývat termínem *maska*. A bude sloužit právě k odstínění, *zamaskování* těch signálů, které nechceme ve výstupním bajtu měnit. Tentokrát však budeme manipulovat s výstupním bajtem, výstupní hodnotou.

Je zřejmé, že hodnota masky pro signál d_4 bude mít hodnotu, která je uvedena v tabulce (7.4.27). Malé připomenutí, požadavek zní: nastavit signál d_4 na hodnotu „L“.

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
1	1	1	0	1	1	1	1

Tabulka 7.4.27: Maska pro signál d_4

Je také patrné, že má-li technika maskování správně pracovat, je nutné použít (aplikovat) bitovou operaci *bitový součin*, označovaný symbolem ($\wedge$). Budeme-li aplikovat masku z tabulky (7.4.27), získáme výsledek uvedený v tabulce(7.4.28).

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	1	0	1	0	1
$\wedge$	↕	↕	↕	↕	↕	↕	↕	↕
	1	1	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
	0	1	1	0	0	1	0	1

Tabulka 7.4.28: Nastavení výstupu d_4 na „L“

V tabulce (7.4.28) lze nahlédnout, že se opravdu změnila pouze hodnota signálu d_4, ostatní zůstaly nezměněny. Nyní „stačí“ odeslat vhodnou instrukcí takto získanou hodnotu na příslušný port.

V tabulce (7.4.29) je příklad programu (fragment kódu), který by mohl popisované činnosti provést. Význam a použití jednotlivých proměnných plyne z jejich mnemotechnických názvů.

```
...
unsigned char outPort; // bajtová hodnota
const int adrPortOut = 0x301;
const unsigned char maskD4L = 0xEF;
...
// nastav signal d4 na "L"
outPort = OutPort & maskD4L;
...
// odesli "ven"
outportbyte(adrPortOut,outPort);
...
```

Tabulka 7.4.29: Fragment kódu pro nastavení výstupu d_4 na „L“

7.4.2 Nastavení výstupní hodnoty „H“

Předpokládejme. že stále máme naši řízenou periferii, která je konstruována tak, že aktivní logické úrovně, které „spotřebovává“ jsou tentokrát „H“. Není to sice běžné řešení, ale čas od času se používá.

Opět jen pro úplnost doplňme, že to je *vstupní signál* z pohledu periferie. Jestliže do periferie tento signál „vstupuje“ musí zákonitě „vystupovat“ do počítače. Z pohledu našeho řídícího programu jde tedy o *výstupní signál*.

Také nyní „zapisujeme“ bajtovou výstupní hodnotu. Ta obsahuje 8 nezávislých jednobitových signálů. Jejich význam je určen konkrétním schematem zapojením.

Dále předpokládejme, že nás bude zajímat signál číslo d_4, viz tabulka (7.4.26), který chceme nastavit na úroveň „H“ tak, aby ostatní signály zůstaly **nezměněny**. Co by se jinak mohlo stát jsme již zmínili.

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	1	1	0	0	1	1	1

Tabulka 7.4.30: Bitové signály

Je zřejmé, že hodnota masky pro signál d_4 bude mít hodnotu, která je uvedena v tabulce (7.4.31). Malé připomenutí, požadavek zní: nastavit signál d_4 na hodnotu „H“.

d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
0	0	0	1	0	0	0	0

Tabulka 7.4.31: Maska pro signál d_4

Je také patrné, že má-li technika maskování správně pracovat, je nutné použít (aplikovat) bitovou operaci *bitový součet*, označovaný symbolem ($\vee$). Budeme-li aplikovat masku z tabulky (7.4.31), získáme výsledek uvedený v tabulce(7.4.32).

	d_7	d_6	d_5	$\boxed{d_4}$	d_3	d_2	d_1	d_0
	0	1	1	0	0	1	0	1
$\vee$	↕	↕	↕	↕	↕	↕	↕	↕
	0	0	0	1	0	0	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
	0	1	1	1	0	1	0	1

Tabulka 7.4.32: Nastavení výstupu d_4 na „H“

V tabulce (7.4.32) lze pozorovat, že se opravdu změnila pouze hodnota signálu d_4, ostatní zůstaly nezměněny. I nyní „stačí“ odeslat vhodnou instrukcí takto získanou hodnotu na příslušný port.

V tabulce (7.4.33) je vhodný příklad programu (fragment kódu), který by mohl popisované činnosti provést.

```
...
unsigned char outPort; // bajtová hodnota
const int adrPortOut = 0x301;
const unsigned char maskD4H = 0x10;
...
// nastav signal d4 na "H"
outPort = OutPort | maskD4H;
...
// odesli "ven"
outportbyte(adrPortOut,outPort);
...
```

Tabulka 7.4.33: Fragment kódu pro nastavení výstupu d_4 na „H“

7.4.3 Závěr

Podrobnější průzkum techniky nastavení výstupu na hodnotu „L“ a techniky nastavení výstupu na hodnotu „H“ nás dovede k zajímavým závěrům.

Porovnáním obou použitých masek s hodnotou masky pro nastavení požadovaného výstupu na úroveň „L“, viz tabulka (7.4.27) a nastavení požadovaného výstupu na úroveň „H“, viz tabulka (7.4.31) lze zjistit jistý vzájemný vztah.

- potřebujeme definovat *dvě různé masky* a použít *dvě různé logické operace*.
- jedna hodnota masky je *bitovou negací* té druhé.

Nabízí se tak možnosti použít pouze jednu masku (pro manipulaci s jedním signálem) a „opačnou hodnotu“ získat její bitovou negací.

Zda je možné použít instrukci pro negaci jednotlivých bitů použité datové struktury je závislé na implementaci. A to jak operačního systému, tak i použitého jazyka. Pokud použitá verze jazyka nemá potřebnou instrukci pro negaci jednotlivých bitových hodnot operandu, je nutné použít jiný postup.

Vhodný, možná obecnější postup je využití *instrukce XOR*, viz příslušná tabulka (7.1.3) v odstavci (7.1.3). Poznamenejme jen, že symbol ($\oplus$) je obvykle používán pro zápis operace XOR. Lze si všimnout, že výstupy bitových operací jsou navzájem komplementární. Platí:

- `0xFF` $\oplus$ `0xEF = 0x10` `(11111111` $\oplus$ `11101111 = 00010000)`
- `0xFF` $\oplus$ `0x10 = 0xEF` `(11111111` $\oplus$ `00010000 = 11101111)`

Je na uvážení uživatele - programátora, který postup považuje výhodnější, zda použití vždy dvojice masek, či kombinaci jedné masky a instrukce XOR.

Kapitola 8
Poučení z historického a didaktického vývoje

Obsah kapitoly

Motto: (konference JČMF, listopad 2014: „Moderní technologie ve výuce fyziky“)
Moderními technologiemi se zde rozumí tablet, iphone, snad i-mobily. Co nevidět bude jistě taková konference i pro výuku matematiky. Takže fyzika i matematika jsou tak nemoderní, že je nutné je zmodernizovat? Nepotřebujeme spíše „zmodernizování“ učitelů?

8.1 Počty, Měřičství, Rýsování

nebo-li Aritmetika, Algebra, Geometrie včera a dnes.

8.1.1 Počítání v éře počítačů

Má smysl učit se počítat a rýsovat (v širším smyslu tím myslíme matematiku všeobecné středoškolské úrovně), když máme dokonalé kalkulačky? Odpovědi mohou být následující:

- Nemá, jedině do výše svého platu či podpory.
- Nemá, nechápal jsem proč jsme se to ve škole učili, nikdy jsem to nepotřeboval.
- Nemá, Je to pouze šance učitelů vybít si svůj intelektuální sadismus na nevinných duších žáků a adolescentů.
- Nemá, na matematice není nic zajímavého; je to pouze memorování fantasmagorických vzorečků.
- Nemá, ve všeobecném referendu by to občané (lid) jasně potvrdili; citát: „Zavrženíhodné umění matematické jest zakázáno především“.

Ale

MÁ!

matematika jako poznání je neoddělitelným základem evropské vzdělanosti.

Vyjádřeno mluvou spíše lidovou:

Používej a trénuj si svůj mozek, jinak o něj přijdeš.

Poznámka:
V Byzanci, východní části římského císařství, vydával císař Justinián I. (527-565) výnosy proti vyznavačům různých kacířství a v roce 529 vydal zákaz vyučovat v Athénách jakékoli filozofii. V právním kodexu, který Justinián uveřejnil čteme odstavec:

„De maleficiis mathematicis et caeteris similibus“

(O zločincích, matematicích a jiných jim podobných)

a v něm se nachází tento bod:

„Ars autem mathematica damnabilis interdicta est omnino“

(Zavrženíhodné umění matematické jest zakázáno především).

V té době byly v hlubokém úpadku všechny světské nauky a s nimi i matematika. Což je celkem charakteristické pro pozdější celkový tragický úpadek byzantinské společnosti a její následný pád (dobytí osmanskými Turky v roce 1453).

Mnohem dalekosáhlejší dopad měla jím podnícená kodifikace římského práva, která vešla ve známost jako „Corpus iuris civilis“. Pro šíření tohoto traktátu byla v Itálii zřízena právnická škola Alma Mater Studiorum Universita di Bolognà a považuje se za nejstarší evropskou univerzitu. Historicky je především známa pro výuku světského a kanonického práva.

Možná, že právě odtud pramení animozita matematiky a politiky, neboť většina politiků jsou právníci, resp. lidé preferující právní vzdělání před matematickým.

8.1.2 Rozvoj počítání

Na první pohled by se mohlo zdát, že úloha vynásobit dvě přirozená čísla je nezajímavá. Známe to přece už ze základní školy. Stačí umět malou násobilku a sčítat, a úloha je vyřešená. Úloha však zajímavá je, protože výpočetních postupů existuje celá řada:

- Egyptské a ruské násobení je založeno na binárním rozvoji násobence.
- Cauchyovo komplementární násobení využívá zápis čísel pomocí záporných cifer.
- Čínské násobení je grafické, a tedy pro žáky s vrozeným odporem k matematice možná nejpřívětivější.
- Ve středověké Evropě bylo běžné převádění násobení na sčítání, někdy velmi zvláštním způsobem. Například se využívalo funkce kosinus ve vzorci:

$$\cos\alpha \times \cos\beta = \frac{1}{2}\left(\cos(\alpha+\beta)+\cos(\alpha-\beta)\right)$$

Dokladem, že se tímto způsobem násobení provádělo, jsou obsáhlé, patnáctimístné tabulky goniometrických funkcí z roku 1613. Dalším převodem násobení na sčítání je převod pomocí logaritmů. Tento postup se ještě před nedávnem učil i na školách (s pomocí logaritmického pravítka), dnes upadá v zapomnění.

Násobení na prstech učitelky na základní škole nevidí rády. Chtějí nás totiž naučit násobit do 10 krát 10 zpaměti „jako když bičem mrská“.

Přesto je použití prstů ruky přirozeným zjednodušením, kterým si lidé pomáhají při výpočtech odedávna. Středověcí obchodníci s oblibou využívali tzv. cikánskou násobilku, která umožňuje násobit pomocí prstů do 9 krát 9 (se znalostí pouhé násobilky do 5 krát 5).

Počítačové násobení v binární soustavě není pro člověka běžné. Lidé totiž díky deseti prstům provádějí výpočty v soustavě desítkové, tj. čísla zapisují pomocí mocnin desítky a cifer od 0 do 9. Se soustavou binární ovšem pracuje valná většina počítačů.

Při násobení na papíře tvoříme částečné součiny, které příslušně posunuté vlevo sepisujeme pod sebe a na závěr všechny sečteme. Na počítači je výhodnější provádět násobení a sčítání současně (viz tzv. algoritmus M[1]). Rychlé násobení je násobení se složitostí menší než počítačové násobení klasické.

Snaha o zrychlení algoritmů se zesiluje ruku v ruce s rozvojem počítačů a speciálně snaha o maximální zrychlení násobení je dána potřebami kryptografie, kde je nutné násobit v přijatelném čase ohromná čísla (řádově 10^{100}).

Dvě rychlá násobení - Karacubovo násobení a Schönhageovo násobení v modulární aritmetice jsou založená na důvtipných myšlenkách a poměrně jednoduše popsatelná. Mezi rychlými algoritmy patří k těm nejstarším, ovšem algoritmy ještě rychlejší už jsou příliš technické.

Zájemce o násobení přirozených čísel s binárním rozvojem délky n blížící se svou složitostí libovolně blízko až k nejnižší hranici $\mathcal{O}(n)$ si dohledá podrobnosti v D. E. Knuth, The Art of Computer Programming volume 2: Seminumerical Algorithms, 3rd

[1] bližší informace - viz http://kmlinux.fjfi.cvut.cz/~balkolub/nasobeni_pocitac.html

ed, Addison-Wesley Boston etc., 1998. Poznamenejme, že jde o jednu z nejrespektovanějších publikací v oboru programování. Donald E. Knuth (na obrázku 8.1.1) je průkopníkem oboru matematické algoritmické analýzy a významnou osobností v mnoha dalších oborech teoretické počítačové vědy.

Obrázek 8.1.1: Donald Erwin Knuth
(zdroj: http://cs.wikipedia.org/wiki/Donald_Ervin_Knuth)

Ani náš nejpoužívanější způsob dělení na papíře nebyl první a zřejmě nebude poslední. V průběhu stovek až tisíců let, vznikaly a zase zanikaly nejrůznější civilizace a s nimi taktéž vznikaly a zanikaly často téměř nepochopitelně složité způsoby pro nás jednoduchých matematických úkonů, jako je například dělení.

Jako otec (S.M.) dítka školou povinného jsem se za komunistické civilizace v Československu seznamoval s „originálním“ způsobem dělení, který byl prý učitelům nařízen Státním Pedagogickým ústavem. Dále jsem se seznamoval s převody číselných soustav a dalšími výdobytky tzv. množinového šílenství. Bylo to tuším v šesté nebo sedmé třídě. Kladu si otázku, zda právě toto není ten důvod, proč žáci i jejich rodiče považují matematiku za tyranii. Žáci mají sice Matematiku v kostce, v kapse, atd. (Jaroslav Eisler – podnikatel), ale nikoliv v hlavě a v srdci.

Silně se přimlouváme za počty ve vzdělávací matematice, včetně dělení na papíře způsobem, který se osvědčil u našich předků.

8.1.3 Významná jména spojená s rozvojem počítání

Diofantos z Alexandrie (asi 200-284 n. l.) – Jeho nejvýznamnější spis, kterým vešel do dějin matematiky a vysloužil si tak označení „otec algebry“, se jmenoval *Aritmetika*. Stejně jako Eukleidovy *Základy* měla i Aritmetika 13 knih, v nichž bylo shromážděno vše, co bylo v době Diofantova života známo o řešení lineárních rovnic a kvadratických rovnic.

Vlivem křesťanských čistek v knihovnách a nešetrnému zacházení se nedochovaly všechny knihy. I tak se staly *Diofantova Aritmetika* a *Eukleidovy Základy* základem studia matematiky po celý středověk a ještě v 17. století byly téměř nenahraditelné.

Diofantos neznal nulu ani záporná čísla a tak např. rovnici:

$$4 = 4y + 20$$

nazývá absurdní rovnice, protože vede na nesmyslné řešení (záporné řešení tehdy neznal).

Motto: *Matematika učí: nepřehlížejte nuly !* (Gabriel Laub)

Všiml si, že některé kvadratické rovnice mohou mít dva kořeny a pro ten účel kvadratické rovnice rozdělil na tři typy podle toho, jaký člen psal na pravou stranu. Vyhnul se tím počítání se zápornými koeficienty.

Při řešení rovnic obecně nepožadoval celočíselné řešení, jak bylo v té době v řecké matematice zvykem, ale spokojil se s kladnými racionálními řešeními. *Diofantova Aritmetika* sehrála velmi důležitou roli pro formulaci a zejména pro důkaz tzv. *Velké Fermatovy věty*. Tu vyslovil francouzský matematik *Pierre de Fermat (1601-1665)*.

Al-Chwárizmí (asi 780-850 n.l.) – První dílo, které popisovalo desítkovou soustavu a operace v ní, byl -*Chwárizmího Aritmetický traktát*. Ten se bohužel dochoval pouze v neúplném latinském překladu z poloviny 12. století s názvem *Algorithmi de numero indorum*. Česky bychom řekli *Al-Chwárizmího Kniha o sčítání a odečítání podle indického způsobu*. Dvojkovou soustavu již 3 000 let př. n. l. objevil čínský císař Fou-Hi. Za novodobého průkopníka dvojkové soustavy považujeme *Gottfrieda Wilhelma Leibnitze* (1646-1716).

John Napier (1550–1617) – byl skotský matematik, fyzik, astronom a astrolog. Do paměti se zapsal jako vynálezce logaritmu, proto se na jeho počest přirozenému logaritmu (tedy logaritmu o základu e) říká *Napierův logaritmus*.

Jakub Filip Kulik (1793-1863) – V roce 1826 byl jmenován profesorem vyšší matematiky na univerzitě v Praze. Vedle podrobných učebnic vyšší analýzy a mechaniky jsou nejvýznamnější jeho díla tabulková (tabulky násobení, druhých a třetích mocnin, logaritmické, trigonometrických a hyperbolických funkcí a jejich logaritmů, tabulky k výpočtu obsahu válcových a kuželových nádob, dělitelů čísel, primitivních kořenů).

Kromě toho Kulik sestavil známé *Kulikovy tabulky* dělitelů čísel od 3 do 100 milionů, které byly uloženy do knihovny Vídeňské císařské akademie věd a na kterých Kulik pracoval dvacet let. Poznamenejme, že při ručním sestavování tak obsáhlých tabulek se Kulik samozřejmě nevyhnul chybám. Navíc v dnešní době počítačů jeho dílo přirozeně upadlo v zapomnění.

Antonín Svoboda (1907–1980) – byl jeden z nejvýznamějších novodobých českých matematiků a průkopníků výpočetní techniky. Narodil se v Praze jako syn profesora

českého jazyka. Vystudoval dvě vysoké školy, nejdříve elektrotechnické a strojní inženýrství na ČVUT, poté fyziku na Karlově Univerzitě.

Po obsazení Československa nacisty utíká i se svou manželkou a kolegou Vladimírem Vandem ze země nejdříve do Francie, odkud se jim později po téměř neuvěřitelných cestách podaří vycestovat do USA. V Americe spolupracoval na vývoji samočinných počítačů pro vojenský projekt protiletadlového zaměřovacího systému MARK 46, který se používal na letadlových lodích dalších 50 let.

V roce 1946 se vrací zpátky do Československa a vydává knihu o mechanických počítacích zařízeních, která o dva roky později vychází i v USA, kde ve stejný rok obdrží ocenění Naval Ordance Development Award. I proto měl v následujících letech problematičtější pozici v komunistickém režimu.

Roku 1950 založil Oddělení matematických strojů. Pod záštitou této instituce začal od roku 1951 pracovat na prvním československém samočinném počítači, zkráceně *SAPO*, jehož základní součástkou byla elektromagnetická relé. *SAPO* byl poprvé slavnostně spuštěn v roce 1958, bohužel byl již v té době opotřeben a roku 1960 vyhořel.

Dalším Svobodovým projektem byl počítač *EPOS*, *Elektronkový POčítač Stroj.* Zajímavostí je, že EPOS pracoval v desítkové soustavě a hardwarově sdílel čas až pro pět programů. Byl spuštěn v roce 1960, měl paměť 40 tisíc slov a rychlost přibližně 35 tisíc operací za sekundu. Následující upravená tranzistorová verze byla spuštěna roku 1962. Roku 1964 Svoboda opět emigroval do Ameriky, kde zůstal až do své smrti, která ho zastihla při pozorování výbuchu sopky Mount St Helen roku 1980.

Richard Dedekind 1831-1916) – byl německý matematik, který proslul v oboru algebry a teorie čísel. K jeho nejznámějším počinům patří konstrukce množiny reálných čísel. Během svého působení v Curychu přišel s myšlenkou konstrukce množiny reálných čísel, které dnes říkáme Dedekindův řez. Její základní myšlenka je tato:

Rozdělíme-li všechny body přímky do dvou tříd tak, aby každý bod první třídy ležel vlevo od každého bodu druhé třídy, pak existuje právě jediný bod, který toto rozdělení všech bodů do dvou tříd, resp. rozříznutí přímky na dva kusy vytváří.

Řečeno populárnějším jazykem, číselná osa není „„děravá“, každý její bod je buď racionální anebo iracionální číslo. Je zajímavé, že iracionálních čísel je „víc“ než racionálních, což dokázal velmi elegantně *Georg Cantor*, se kterým se Dedekind osobně znal a podporoval jej v jeho odborném zápolení s Kroneckerem.

Georg Cantor (1845-1918) – nejdříve zavedl nástroj (termín) pro porovnávání konečných množin. Například množiny:

$$\{1,2,3\} \text{ a } \{2,3,4\}$$

si nejsou rovny, ale mají *stejnou kardinalitu.* Je to zobecnění termínu „počet prvků“. Cantora zajímalo, zda každá nekonečná množina má vlastnost kardinality. Kardinalitě množiny přirozených čísel dal název *spočetnost* a označil ji symbolem:

$$\aleph_0 \quad \text{(alef nula).}$$

Dokázal, že množina reálnách čísel není spočetná. Cantor naučil matematiku počítat s nekonečnem (aktuální nekonečno!).

8.2 Neúplná čísla a úplné číselné množiny

V roce 1889 vyšel velmi hezky napsaný článek:

O relativních chybách čísel neúplných

Sepsal:

F.Hoza, Ředitel vyšších reálných škol v Hradci Králové

Citace:

> *„V učebných knihách mathematických pohřešujeme téměř napořád nauku o relativních chybách čísel neúplných. Kterak nauka ta důležitá jest pro počítání praktické, vysvitá odtud, že vede k jednoduchým větám o počtu spolehlivých míst ve výpočtu. . . “*
>
> (Časopis pro pěstování mathematiky a fysiky, vol. 18 (1889), issue 1, pp. 5-9).

Domnívali jsme se, že termín „neúplné číslo“ už opravdu patří do éry Rakouska-Uherska. Termín „neúplné číslo“ je nevhodný! Obvykle se jím (pouze na škole) rozumí číslo získané měřením nebo aproximací jiného čísla. Nevhodnost spočívá v tom, že neúplným číslem může být jakékoliv reálné(racionální, celé, přirozené číslo). Například číslo π je aproximací čísla $\frac{22}{7}$, a tudíž je „neúplným číslem“!?

K našemu údivu, téměř šoku, zjišťujeme, že je stálou součástí učebnic matematiky (v kostce, na dlani, v kapse, pro nižší gymnázia, atd.).

Klademe si otázku, zda se to stále učí na školách, protože se to učí na pedagogických fakultách, nebo se to učí na pedagogických fakultách, protože se to stále učí na školách? Vřele doporučujeme, abychom tento termín nechali v klidu odpočívat v Rakousku-Uhersku.

Máme dvě vhodné náhrady:

- *Aproximace reálného čísla*, například: čísla 5; 3; 3,1; 3,14; jsou různé aproximace čísla π,
- *Obraz reálného čísla do množiny* $M(q,t)$, zobrazení: $\gamma : \mathbb{R} \to M(q,t)$, viz odstavec 4.2 na straně 57.

V matematice je velmi frekventovaný a důležitý termín *úplná množina*. Je to například množina reálných čísel $\mathbb{R}$, ale nikoliv množina racionálních čísel $\mathbb{Q}$. Už jsme se zmínili, že v $\mathbb{Q}$ jsou „díry“, kdežto v $\mathbb{R}$ nikoliv. O úplných a neúplných množinách ještě hovořit budeme na různých místech této série sešitů.

Důležitým procesem je tzv. zúplnění. Například víme, že když k množině racionálních čísel $\mathbb{Q}$ přidáme všechna iracionální čísla, dostaneme množinu reálných čísel $\mathbb{R}$. Dále víme, že každé iracionální číslo lze libovolně přesně aproximovat nějakým racionálním číslem. Posloupnost takových aproximací nám tedy reprezentuje iracionální číslo (viz např. číslo π). Třída všech těchto posloupností je tedy tím „zaplněním“ příslušné „díry“ v množině $\mathbb{Q}$.

8.3 Dějepis komplexních čísel

8.3.1 Realita a mystika

K tomu, aby mohla být v matematice uznána existence komplexních čísel a došlo k implementaci těchto čísel do matematiky, musela být uznána existence jednak záporných veličin a jednak druhých odmocnin. V babylonské matematice došlo k přibližnému vyčíslování druhých odmocnin, chyběly zde však záporné veličiny. Záporná celá čísla se objevují v čínské matematice. Až indická matematika (především do 5.století n.l.) při řešení kvadratických rovnic se setkává s problémem druhých odmocnin ze záporných veličin. Problém však indická matematika obchází tím, že prohlašuje tyto druhé odmocniny za neexistující. Příslušné kvadratické rovnice vedoucí k druhým odmocninám ze záporných veličin jsou prohlášeny za neřešitelné.

V dějinách matematiky lze pozorovat dva odlišné filozoficko-metodologické přístupy ke komplexním číslům. Pracovně je označíme jako *mysticko-operacionalistický* a *realistický*.

Mysticko-operacionalistický přístup je většinový a zaujímala jej řada předních matematiků 16.-18. století. Než se budeme zabývat postojem jednotlivých matematiků, pár předběžných poznámek. Komplexní čísla stojí v protikladu k předchozím číslům, která vznikala v postupném vývoji matematiky. Byla zde čísla přirozená, která byla kvantitativními charakteristikami konečných předmětných souborů či událostí diskrétní povahy. Byla zde záporná čísla celá, která popisovala v ekonomické interpretaci dluhy. Byly zde zlomky, které popisovaly části celku, či vyjadřovaly pravděpodobnost. Byla zde čísla reálná, která popisovala spojitě se měnící kvantitativní procesy v objektivním stavu bytí.

Poté se objevují čísla komplexní jako něco, co je zbaveno objektivního obsahu. Jsou řešením kvadratických rovnic, které de facto nemají řešení, suplují neexistující průsečíky. Předchozí číselné obory byly stíny existujících objektů, komplexní čísla se jevila jako stíny těchto stínů, nebylo možné dohlédnout skutečné objekty, které jsou příčinou těchto stínů. Komplexní čísla se jevila jako abstrakta nemající své pravzory, zdála se být výplodem čistého myšlení, měla jakousi duální existenci, nacházela se někde mezi bytím a nebytím.

Většina matematiků 17. a 18. století byla přesvědčena o mystickém charakteru komplexních čísel, o nemožnosti reálného zdůvodnění jejich existence, o objektivnosti jejich smyslu. Jsou nazývaná *pomyslnými, fiktivními* čísly, která nemají

žádný objektivní obsah a nemají žádnou interpretaci. Dostávají atribut mystičnosti a jsou někdy též označována jako božská čísla. Na druhé straně je jim přiznána užitečnost, efektivnost při výpočtech a jejich nezastupitelnost pro zobecňování, završování a elegantnost matematické teorie. (Mají obdobnou funkci jako svorník v gotické klenbě.)

Uveďme názory některých matematiků.

Newton Při řešení konkrétní úlohy jsou možné dva případy: buď je tato úloha řešitelná a potom její řešení se nalézá v oblasti reálných čísel, nebo je tato úloha neřešitelná, což je zdůvodněno tím, že její řešení leží v množině komplexních čísel. Komplexní čísla jsou zdůvodněním neřešitelnosti úlohy.

Euler Euler si je vědom nedostatku komplexních čísel, kterým je nemožnost jejich uspořádání. Na druhé straně definuje matematiku jako vědu o veličinách, kdy pod veličinou se rozumí něco, co může být zvětšeno nebo zmenšeno. Pod kladnou veličinou se rozumí veličina větší než nula, pod zápornou veličinou veličina menší než nula. Komplexní čísla jsou tak Eulerem postavena vně matematiky. Jedná se o fiktivní, nemožná čísla, nemající reálný obsah. Existují pouze v naší fantazii.

D'Alembert poukazuje na principiální rozdíl mezi iracionálními a komplexními čísly. Iracionální čísla nelze reprezentovat pomocí racionálních čísel, ale jsou reprezentovatelná pomocí geometrických objektů (např.jako úhlopříčka čtverce o jednotkové straně). Komplexní čísla tuto vlastnost nemají, jsou nevyjádřitelná a nereprezentovatelná. Komplexní čísla považuje za pouhé hieroglyfy absurdních kvantit.

Navíc některé vlastnosti komplexních čísel jsou kontroverzní k uctívaným vlastnostem reálných (např. vzpomenutá uspořádanost množiny reálných čísel, nebo pravidlo $\sqrt{a \cdot b} = \sqrt{a} \cdot \sqrt{b}$). Mechanické převádění vlastnosti reálných čísel do oblasti komplexních čísel na základě nezdůvodněné analogie často ústilo v paradoxech. To vše vedlo k odmítnutí komplexních čísel jako objektivně existujících objektů.

Existuje ovšem kardinální problém. Jak je možné, že fiktivní, nic neodrážející komplexní čísla, jsou spolehlivým nástrojem matematického zkoumání a vedou ke správným výsledkům. Dochází k paradoxnímu spojení: odmítnutí objektivnosti a pragmatické uznání prospěšnosti a užitečnosti.

Řada matematiků (J. Bernoulli, D'Alembert, Carnot, Laplace) chápou komplexní čísla jako formy, v nichž lze vyjádřit reálné veličiny. Pregnantně to vyjadřuje Carnot:

> „V algebře často používáme ryze imaginární pojmy, fiktivní podstaty (etres), které neexistují, jsou nepochopitelné, neztrácejí však svou užitnou hodnotu. Mají pouze pomocnou funkci, umožňující nalézt správné vztahy mezi skutečnými kvantitami. Tyto vztahy jsou získávány pomocí určitých transformací majících ryze mechanický charakter.“

Cauchy podtrhuje užitečnost komplexních čísel pro zjednodušení výpočtů, pro elegantní a zkrácený zápis složitých výrazů. Mystický přístup ke komplexním číslům dostává novou dimenzi, kterou můžeme označit jako *operacionalistický přístup.*

Realistický přístup byl přístupem, ve kterém je dán komplexním číslům reálný výklad, jsou dány jejich obsahové interpretace, je jím propůjčen atribut objektivnosti.

První matematik, který se snažil o reálný výklad komplexních čísel byl *J. Wallis.* V odmítnutí reálnosti imaginárních veličin spatřuje současné odmítnutí reálnosti záporných veličin. Žádná veličina nemůže být menší než nic je výrok ve stejné rovině jako výrok, že neexistuje veličina, jejíž čtverec by byl zápornou veličinou.

Wallis ve své Algebře (1695) vyslovil obdivuhodnou myšlenku, která se později ukázala jako plodná:

> „Záporné veličiny vyjadřují protikladné stavy. To, co připouštíme na přímce vzhledem k záporným veličinám, musíme připustit v rovině ohledně komplexních čísel. Komplexní čísla připouští jak geometrický, tak algebraický výklad.“

Za povšimnutí stojí idea o dvourozměrnosti při interpretaci komplexních čísel. Wallis interpretuje imaginární veličiny jako délky stran čtvercových pozemků, které představují dlužné pozemky. Znamená to, že dlužný čtvercový pozemek o rozloze $-a$ arů má velikost strany $\sqrt{-a}$. Odtud již plyne interpretace oné záhadné formule:

$$\sqrt{-a} \cdot \sqrt{-a} = -a.$$

Jedná se o první dosti spekulativní interpretaci komplexních čísel. Se zcela zdařilou interpretací se setkáváme u dánského matematika *K. Wessela.* V roce 1799 publikuje práci, ve které komplexní čísla interpretuje jako orientované veličiny, tj. jako vektory. Jestliže aritmetika reálných čísel popisuje veličiny působící v přímém či opačném směru, tak komplexní čísla jsou popisem orientovaných veličin působících v rovině. Ty již nelze popsat reálnými čísly. Jde o velmi moderní interpretaci, která na čas zanikla, protože Wesselův spis byl psán dánsky a přední evropští matematici tento jazyk neovládali.

Komplexní číslo $a + \imath b$ je interpretováno jako vektor, jehož počáteční bod je v počátku pravoúhlého souřadnicového systému a koncový bod je bod mající souřadnice $[a, b]$. Sčítání komplexních čísel je interpretováno jako sčítání geometrických vektorů, viz odstavec 1.2.

Problémem bylo násobení komplexních čísel ve vektorové interpretaci. Součin komplexních čísel přes vektorové reprezentanty se provedl tak, že velikost příslušné úsečky reprezentující součin komplexních čísel se dostal grafickým součinem úseček odpovídajících komplexním číslům, jejichž součin se hledal. Amplituda součinu (úhel, který svíral vektor s kladnou poloosou) se dostala součtem amplitud násobených komplexních čísel.

Wessel dostává zcela přesvědčivou interpretaci záhadné formule $\imath^2 = -1$. Imaginární jednotka $\imath$ je interpretována jako jednotkový vektor v kladné části poloosy y a tedy vektor reprezentující čtverec imaginární jednotky leží v záporné části poloosy x a reprezentuje tedy číslo -1. Imaginární jednotka dostává jednoduchou geometrickou interpretaci - násobení touto jednotkou představuje rotaci o

90 stupňů. Řada matematiků (R. Argand, J. Francis), kteří vykládali imaginární jednotku jako symbol kolmosti v podstatě nepřekročili horizont Wesselovy interpretace komplexních čísel.

8.3.2 Algebra

V roce 1545 *G.Cardano* publikuje práci „Artis magnae sive de regulis algebraicis“, v níž se zabývá řešením kubické rovnice pomocí odmocnin. Objevují se jeho proslulé vzorce pro řešení kubické rovnice, která je ve tvaru:

$$\xi^3 + p\xi + q = 0.$$

Uvažujme z Cardanových vzorců ten vzorec, který vyjadřuje kořen ξ_1 bez použití primitivních třetích odmocnin z jedné (tento vzorec též Cardano využíval při řešení kubické rovnice). Kořen ξ_1 je vyjádřen ve tvaru:

$$\xi_1 = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

Z klasické algebry víme, že je-li $D > 0$ kde:

$$D = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3,$$

potom vzorec pro ξ_1 vyjadřuje reálný kořen kubické rovnice, zbývající dva kořeny jsou komplexní a to komplexně sdružené (ty se ovšem za kořeny nepovažovaly).

S nepříjemnou situací se Cardano setkává v případě, kdy $D < 0$ (tzv. případ *casus ireducibilis*). Tak tomu jest u kubické rovnice:

$$\xi^3 - 21\xi + 20 = 0.$$

V tomto případě je $D = -243$ a pro kořen ξ_1 dostáváme:

$$\xi_1 = \sqrt[3]{-10 + \sqrt{-243}} + \sqrt[3]{-10 - \sqrt{-243}}.$$

Vyjdeme-li z tehdy přijaté teze, že jsou-li kořeny algebraické rovnice vyjádřeny pomocí druhých odmocnin ze záporných veličin, potom tato rovnice není řešitelná, tak daná kubická rovnice nemá řešení. *Hornerovým algoritmem* se ovšem snadno přesvědčíme, že daná kubická rovnice má dokonce tři reálné kořeny $1, 4, -5$.

Dochází k radikálnímu zvratu. Jestliže se dříve komplexní čísla vyskytovala až v závěru výpočtu, což se považovalo za prokázání neřešitelnosti příslušné rovnice, tak nyní se komplexní čísla stávají průběžným mezičlánkem ve výpočtech, které posléze vedou k reálným kořenům dané rovnice. Této skutečnosti si všimla další významná osobnost italské algebraické školy - *R. Bombelli*. Ve své Algebře (1572) na konkrétních příkladech ukazuje, že v případě je kořen vyjádřen jako součet dvou čísel komplexně sdružených a je tedy číslem reálným.

Výpočet tedy není ukončen komplexními čísly, která vypovídají o neřešitelnosti rovnice, ale stávají se průběžným článkem výpočtu. S těmi je třeba provádět záhadné mystické operace, podle specifických pravidel, končící reálnými čísly.

Reálné kořeny kubické rovnice jsou maskovány pomocí komplexních čísel. O jedné takové záhadné operaci se již přesvědčil Cardano, když řešil úlohu:

Vyjádři číslo 10 jako součet dvou sčítanců, jejichž součin je roven 40.

Příklad 8.3.1: Cardano - kvadratická rovnice

Úloha vede, jak se snadno přesvědčíme, ke kvadratické rovnici:

$$\xi^2 - 10\xi + 40 = 0$$

Tato rovnice má kořeny:

$$\xi_1 = 5 + \sqrt{-15}\,,\ xi_2 = 5 - \sqrt{-15}.$$

Co když upravíme původní tezi o neřešitelnosti dané úlohy a prohlásíme tato „podivná“ čísla za řešení této úlohy. Není problém vidět, že sečtením těchto čísel dostaneme číslo 10. Horší je to s jejich součinem. Abychom je mohli prohlásit za řešení naší úlohy, musíme přijmout pravidlo:

$$\sqrt{-15} \cdot \sqrt{-15} = -15.$$

Toto pravidlo je však v kontroverzi s uznávaným pravidlem platícím v oblasti reálných čísel:

$$\sqrt{a} \cdot \sqrt{b} = \sqrt{a \cdot b},\ \text{ kde } a > 0 \wedge b > 0.$$

Uznání komplexních čísel (ve tvaru druhých odmocnin ze záporných veličin) jakožto řešení algebraických úloh vedlo italskou algebraickou školu k poznání, že pro tato čísla začínají platit jiná pravidla, než ve světě reálných veličin. Jde například o pravidlo:

$$\sqrt{-a} \cdot \sqrt{-a} = -a\,,\ \text{kde } a > 0.$$

Cardano vzhledem k podivným vlastnostem komplexních čísel tato čísla prohlašuje za „sofistická“. Cítí, že je však nelze ignorovat, praxe řešení algebraických rovnic je vyžaduje. Jejich smysl však zůstává zahalen (proto též někdy jejich označení jako mystických čísel).

Některé jejich vlastnosti, ke kterým se dospělo pragmatickou cestou (aby byla řešením daných problémů), se jevily v té době jako paradoxní a těžko pochopitelné. Bombelli jako první ustálil zápis komplexních čísel a formuloval pravidla pro jejich sčítání a násobení. Komplexní čísla zapisuje ve tvaru:

$$a + b\sqrt{-1}.$$

Definuje rovnost komplexních čísel:

$$a + b\sqrt{-1} = a' + b'\sqrt{-1} \ \Leftrightarrow\ a = a' \ \wedge\ b = b'.$$

Definuje sčítání a násobení komplexních čísel:

$$(a + b\sqrt{-1}) + (a' + b'\sqrt{-1}) = (a + a') + (b + b')\sqrt{-1}.$$

$$(a + b\sqrt{-1}) \cdot (a' + b'\sqrt{-1}) = (aa' - bb') + (ab' + a'b)\sqrt{-1}.$$

Pro pravidlo o násobení musí však formulovat již výše vzpomenuté kontroverzní pravidlo pro $a = 1$:

$$\sqrt{-1} \cdot \sqrt{-1} = -1$$

Toto pravidlo již představuje proslulou známou rovnost:

$$\imath^2 = -1.$$

V roce 1629 *Girard* formuluje tvrzení, že algebraická rovnice $n-tého$ stupně má právě n kořenů. Tím, ještě před Gaussem, formuluje základní větu algebry. K takovému tvrzení je však nutné zrovnoprávnit komplexní čísla s reálnými čísly a přiznat jim objektivní status kořenů algebraických rovnic.

Girard poukazuje na to, že pro vyvození *Vietových vzorců*, které vyjadřují vztahy mezi kořeny a koeficienty algebraických rovnic, je nutné neomezené používání komplexních čísel. Girard jako první odpovídá na otázku: K čemu jsou komplexní čísla?

„Pro správnost obecných vzorců a kvůli prospěšnosti.“

Gauss 11.12.1811 v dopise *Besselovi* zdůrazňuje:

> „Komplexní čísla jsou nepostradatelná pro uspořádanost a završenost matematické teorie. V případě jejich použití není nutné obecné pravdy znásilňovat nepotřebnými omezeními.“

8.3.3 Analýza

G. W. Leibniz (1646 - 1716), *Abraham de Moivre* (1667 - 1754) a *L. Euler* (1707 - 1783), zdůrazňovali význam a použití komplexních čísel v různých matematických oborech.

Euler zavedl označení pro imaginární jednotku. Leibniz některé integrály reálných funkcí zašifroval v podobě přirozených logaritmů komplexních čísel. Učinil již tak při integraci funkce $\frac{1}{1+x^2}$. Tuto funkci lze vyjádřit jako součet parciálních zlomků ve tvaru:

$$\frac{1}{1+x^2} = \frac{1}{2}\left(\frac{1}{1+\imath x} + \frac{1}{1-\imath x}\right).$$

Odtud dostaneme:

$$\int \frac{1}{1+x^2}\,\mathrm{d}x = \frac{1}{2\imath}\ln\frac{1+\imath x}{1-\imath x} + C.$$

Na druhé straně však víme, že platí:

$$\int \frac{1}{1+x^2}\,\mathrm{d}x = \operatorname{arctg} x$$

Tím Leibniz dokázal proslulou identitu:

$$\operatorname{arctg} x = \frac{1}{2\imath}\ln\frac{1+\imath x}{1-\imath x}.$$

Vzniklá situace je analogická k situaci v algebře, kdy reálné kořeny kubické rovnice byly zašifrovány pomocí komplexních čísel. V infinitesimálním počtu analogicky byly reálné funkce zašifrovány v nic neříkajících logaritmech komplexních čísel.

V tomto trendu pokračuje L. Euler. V roce 1740 v dopise *J. Bernoullimu* sděluje, že objevuje formule:

$$\cos x = \frac{e^{\imath x} + e^{-\imath x}}{2},$$

$$\sin x = \frac{e^{\imath x} - e^{-\imath x}}{2}.$$

K těmto formulím dochází na základě teorie diferenciálních rovnic. Např. uvažujeme-li počáteční úlohu:

$$y'' + y\,, \quad y(0) = 1\,,$$

tak řešením této rovnice, za dané počáteční podmínky, jsou právě uvažované Eulerovy funkce stojící v rovnosti. Stejně se dokáže, že $y = \sin x$ je řešením úlohy:

$$y'' + y = 0\,, \quad y(0) = 0.$$

Není problémem z uvedených formulí vyvodit výsledek:

$$e^{\imath x} = \cos x + \imath \sin x.$$

Což je proslulá Eulerova formule, kterou Euler dokazuje v roce 1743. Imaginární jednotka se tak ukazuje jako most spojující dříve od sebe ostře oddělené goniometrické funkce a funkce exponenciální.

V Eulerově době byla v matematice považována za nejdůležitější čísla:

$$0, 1, \pi, e, \imath.$$

Matematici byli přesvědčeni, že tato čísla jsou na sobě nezávislá. Dosadíme-li do poslední formule $x = \pi$, dostaneme zajímavou rovnost $e^{\imath\pi} = -1$. Odtud již dostáváme:

$$e^{\imath\pi} + 1 = 0.$$

Euler tuto formuli považuje za tak excelentní, že ji považuje za důkaz Boha. Ta umožnila Eulerovi vybudovat teorii logaritmů se záporným argumentem a obecně teorii logaritmů s komplexním argumentem.

Nejprve se ptejme, jak vypadá logaritmus záporného čísla. Dosadíme-li do příslušných formulí $x = \pi + 2k\pi$, dostaneme:

$$e^{\imath(\pi + 2k\pi)} = -1.$$

Po vynásobení kladným číslem a, dostaneme po úpravě:

$$e^{\ln a + \imath(\pi + 2k\pi)} = -a$$

Z čehož už dostáváme rovnost:

$$\ln(-a) = \ln a + \imath(\pi + 2k\pi)\,,\ k \in \mathbb{C}.$$

Závěr 1: Každé záporné číslo má v množině komplexních čísel nekonečně mnoho přirozených logaritmů, které jsou všechny komplexními čísly.

Obdobným způsobem můžeme odvodit rovnost:

$$\ln a = \ln a + 2\imath k\pi\,,\ k \in \mathbb{C}.$$

pro kladné číslo a.

Závěr 2: Každé kladné číslo má v množině komplexních čísel nekonečně mnoho přirozených logaritmů, z nichž právě jeden je reálným číslem (pro $k = 0$).

Euler odvodil celou řadu překvapivých rovností svazujících komplexní čísla s čísly reálnými. Na ukázku aspoň jednu z nich. Položíme-li v rovnosti $y'' + y = 0\,, \quad y(0) = 1$, dostaneme:

$$0 = e^{\imath\frac{\pi}{2}} + e^{-\imath\frac{\pi}{2}}.$$

Zavedením substituce $z = e^{\imath\frac{x}{2}}$ získáme rovnost:

$$0 = z + \frac{1}{z},$$

ze které plyne $z = \imath$. Tak dostáváme rovnost:

$$\imath = e^{\imath\frac{x}{2}}.$$

Umocněním na $\imath$ posléze dostáváme exkluzivní rovnost:

$$\imath^{\imath} = e^{-\frac{\pi}{2}}.$$

Nejdůležitějším metodologickým výsledkem matematiky 17. a 18. století byl důkaz uzavřenosti množiny komplexních čísel vůči všem algebraickým a elementárním transcendentním operacím.

To znamenalo např. dokázat, že mocniny komplexních čísel s komplexním exponentem, odmocniny komplexních čísel, logaritmy komplexních čísel, goniometrické funkce komplexních čísel, exponenciální funkce s komplexním exponentem jsou vždy komplexní čísla.

Ukažme to na příkladě exponenciální funkce $e^{a+\imath b}$ a logaritmické funkce $\ln(a+\imath b)$. Pro exponenciální funkci $e^{a+\imath b}$ bude postupně platit (použitím Eulerovy formule):

$$e^{a+\imath b} = e^a \cdot e^{\imath b} = e^a(\cos b + \imath \sin b).$$

Pro logaritmickou $\ln(a+\imath b)$ funkci bude postupně platit:

$$\ln(a+\imath b) = \ln(\varrho(\cos\alpha + \imath\sin\alpha)) = \ln\varrho + \ln(\cos\alpha + \imath\sin\alpha) = \ln\varrho + \imath\alpha.$$

Přešli jsme ke goniometrickému tvaru komplexního čísla a opět použili Eulerovu formuli.

Tím byla matematika pojištěna před tím, že v důsledku neuzavřenosti komplexních čísel vůči nějaké algebraické či transcendentní operaci by bylo nutné budovat nějaký nový číselný obor vzniklý rozšířením množiny komplexních čísel.

8.3.4 Geometrie

Od konce 18. století, kdy se mnozí matematici zabývali také znázorňováním komplexních čísel v rovině, užívala se komplexní čísla velmi často. Přispěly k tomu i práce o komplexních číslech německého matematika *K. F. Gausse* (1777 - 1855) a anglického matematika *W. R. Hamiltona* (1805 - 1865).

V 17. století v důsledku vzniku analytické geometrie se komplexní čísla dostávají do této nové matematické disciplíny, u jejíž zrodu stál *Descartes*. Při zkoumání vzájemné polohy rovinných křivek Descartes prohlašuje:

> „Jestliže kružnice neprotíná a ani se nedotýká paraboly v žádném jejím bodě, znamená to že příslušné rovnice odpovídající těmto křivkám mají pomyslné kořeny (pomyslné - čteme imaginární).“
> (René Descartes (lat. Renatus Cartesius) (1596-1650) byl francouzský filosof, matematik a fyzik).

Descartes jasně dává najevo, že v analytické geometrii komplexní čísla nemají reálný význam, body s komplexními souřadnicemi neexistují, komplexní čísla nazývá pomyslnými čísly.

Descartes navazuje na prvotní výklad významu komplexních čísel v algebře (vypovídají o neřešitelnosti úlohy), komplexní průsečíky daných křivek vypovídají o neexistenci těchto průsečíků, křivky se neprotínají. Descartes však dává komplexním číslům atribut užitečnosti.

Umožňují totální klasifikaci vzájemné polohy dvou křivek. Přímka je nesečna kružnice právě tehdy, když příslušná soustava lineární a kvadratické rovnice má řešení v množině komplexních čísel.

Situace se radikálně mění se vznikem projektivní geometrie a se zavedením homogenních souřadnic, které umožňují popsat nevlastní body roviny. Provedeme několik demonstrací.

V rovině opatřené pravoúhlým souřadnicovým systémem hledejme geometrické místo bodů, které mají od počátku souřadnicového systému vzdálenost rovnou nule. Úloha vede k rovnici:

$$x^2 + y^2 = 0.$$

Uvažujeme-li komplexní čísla, můžeme rovnici přepsat do tvaru:

$$(x + \imath y) \cdot (x - \imath y) = 0.$$

Řešením je dvojice přímek:

$$x + \imath y = 0 \ , \ x - \imath y = 0,$$

která je dvojicí komplexních přímek (komplexně sdružených) majících reálný průsečík v počátku souřadnicového systému.

Tyto přímky se nazývají *izotropickými přímkami* eukleidovské roviny a mají tu zvláštní vlastnost, že dva různé body nacházející se na jedné z těchto přímek mají nulovou vzdálenost. Přestože tyto přímky jsou v Descartově terminologii naprosto pomyslné, mohou charakterizovat naprosto reálné vlastnosti.

S rozvojem projektivní geometrie vznikly alternativní geometrie 19. století:

- hyperbolická (Lobačevského geometrie),
- eliptická (Riemannova geometrie),
- parabolická (Eukleidovská geometrie).

Sférická geometrie – geometrie na povrchu koule je zvláštní případ eliptické (Riemannovy) geometrie.

Představa, že imaginární kružnice je řešením rovnice kružnice $x^2 + y^2 = -1 = \imath^2$ s imaginárním poloměrem patří k těm, které jsme ochotni akceptovat.

Charakteristickou vlastností tzv. hyperbolické geometrie je, že součet vnitřních úhlů každého trojúhelníku v této geometrii je menší než 180°. Součet vnitřních úhlů trojúhelníka může být libovolně malý. Lokálně se dá hyperbolická geometrie modelovat na

části plochy, která má zápornou Gaussovu křivost, např. jednodílného hyperboloidu nebo hyperbolického paraboloidu.

Lobačevského hyperbolická geometrie se realizuje na sféře s imaginárním poloměrem.

Posledním matematikem, který uzavírá etapu geometrické interpretace komplexích čísel, je *C. F. Gauss*. V roce 1811 ve svém dopise Besselovi píše:

> „Stejně tak jako lze reálná čísla zobrazit pomocí nekonečné přímky, totální oblast všech veličin reálných a imaginárních lze zobrazit pomocí nekonečné roviny. Každý bod o souřadnicích $[a, b]$ současně představuje veličinu $a + \imath b$“

To je již idea Gaussovy roviny jakožto geometrické interpretace komplexních čísel. Zároveň je to ochuzená interpretace *Wesselova*, kdy modelování komplexního vektorem je zúženo na modelování bodem, představujícího koncový bod vektoru. Wesselova interpretace je totálnější a obsahově bohatší.

Geometrická interpretace komplexních čísel je jeden z vrcholů matematického myšlení 19. století.

> „Odhalení geometrické interpretace komplexních čísel představuje jeden z nejdůležitějších momentů dějin matematiky.“
>
> (N. Bourbaki)

> *Nicolas Bourbaki* byl pseudonym, pod kterým od roku 1935 publikovala svoje práce skupina převážně francouzských matematiků. Členové této skupiny jsou souhrnně označováni jako *bourbakisté*. Záměrem Bourbakistů bylo vybudovat celou matematiku na základě teorie množin. Tím měla být matematika plně axiomatizována a postavena na pevný základ.

F. Klein poukazuje na důležitost komplexních čísel pro totální popis podobných zobrazení v rovině. Sčítání komplexních čísel slouží pro popis posunutí v rovině, násobení komplexních čísel pro rotaci (např. $\varrho \cdot e^{\imath\varphi}$ je analytickým vyjádřením rotace roviny kolem počátku o úhel φ při změně velikosti úseček v poměru $1 : \varrho$).

Na závěr odstavce uveďme příklad úloh, kterými se zabývali již v 16. století italští matematikové *Hieronýmo Cardano* a *Rafael Bombelli*:

Rozložte číslo 10 v součet takových čísel, aby jejich součin byl 40.

Označíme-li hledaná čísla u, v, má platit:

$$u + v = 10\ ,\ u \cdot v = 40$$

Po dosazení:

$$(10 \breve{} v) \cdot v = 40, \text{ neboli } v^2 - 10v + 40 = 0.$$

Snadno se přesvědčíte, že řešení této kvadratické rovnice vede ke kořenům:

$$v_1 = 5 + \sqrt{-15}\ ,\ v_2 = 5 - \sqrt{-15}.$$

Připomeňme, že $\sqrt[n]{a}$ byla definována jen pro přirozená čísla n a nezáporná reálná čísla a.

Naše kvadratická rovnice je v tzv. normovaném tvaru. To pro její kořeny znamená, že:

$$v_1 + v_2 = 10\ ,\ v_1 \cdot v_2 = 40$$

První rovnost zřejmě platí, neboť je zřejmé, že, že $\sqrt{-15}$ a $-\sqrt{-15}$ jsou navzájem opačné prvky (jejich součet se rovná nule). Pak:

$$(5+\sqrt{-15})+(5-\sqrt{-15}) = 10.$$

Levá strana druhé rovnosti má tvar rozdílu čtverců. Lze pak zapsat:

$$(5+\sqrt{-15})\cdot(5-\sqrt{-15}) = 5^2-(\sqrt{-15})^2 = 25-(-15) = 40.$$

S odmocninou ze záporného čísla jsme zde však pracovali podle pravidel zavedených dosud jen pro počítání s odmocninami z nezáporných čísel.

Odmocniny ze záporných čísel byly nazývány imaginárními (neskutečnými) čísly. Výrazy tvaru $a+\imath b$, kde a, b jsou reálná čísla se začaly nazývat komplexní čísla (složená čísla).

8.3.5 „Komplexní“ perly ze současnosti

Když ve slovníku cizích slov hledáte význam slova imaginární,najdete, že se tím rozumí vymyšlené, pomyslné, neskutečné, zdánlivé.

Když si zvídavý matematický analfabet chce zjistit význam termínu imaginární číslo, slovník mu nic nenabídne a pokud jeho zvídavost není hlubší, spojí si tento termín se slovem vymyšlené číslo. Neuvědomí si, že vlastně všechna čísla jsou vymyšlená.

Na otázku, co je komplexní číslo, dostaneme kliknutím odpověď:

- Od běžných čísel se komplexní čísla liší především v tom, že obsahují dvě části – reálnou a imaginární.

Do života si odneseme poznatek, že to nejsou běžná čísla. Víme vůbec, co to jsou běžná čísla? Kliknutím zdroj informací raději zavřeme. Zjistili bychom také, že:

- např. číslo $2,565656\ldots$ je iracionální

(VŠTE CB - „Iracionální čísla jsou čísla s nekonečným desetinným rozvojem“). A co ty tečky?

Perla z wikipedie:

- množinu $\mathbb{C}$ komplexních čísel získáme z množiny reálných čísel $\mathbb{R}$ tak, že k ní přidáme číslo $\imath$ ($\mathbb{C} = \mathbb{R} + \imath$??).

To byl napsáno ve zprávě o vyřešení jednoho projektu spolufinancováném Evropským sociálním fondem a státním rozpočtem České republiky (INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ).

Závěr? No comment!

8.4 Co je matematika

8.4.1 „Definice matematiky“

Matematika je věda zabývající se z formálního hlediska kvantitou, strukturou, prostorem a změnou. Matematika je též popisována jako disciplína, jež se zabývá vytvářením abstraktních entit a vyhledáváním zákonitých vztahů mezi nimi.

> „Matematika nám dává oči, kterými můžeme spatřit to, co by našemu zraku zůstalo jinak skryto. V tomto smyslu lze říci, že matematika je způsob, jak zviditelnit neviditelné.“
>
> (Keith Devlin)

Krásný obraz, hudební skladba i román musí být překvapující i samozřejmý, harmonický i šokující, všezahrnující i rafinovaně jednoduchý. Taková musí být i každá matematická teorie.

Co je to matematika? Obvyklá odpověď, kterou uslyšíme, bude: „Matematika – to jsou počty“. Možná se nám dostane upřesněni, že jde o vědu o číslech. Ale takto chápaný popis matematiky přestal platit již před 2 500 lety!

Pro starověké Řeky, kteří kladli důraz na geometrii, byla již matematika naukou o číslech a tvarech. Newton a Leibniz zase přetvořili matematiku ve studium čísel, tvarů, ale také pohybu, změny a prostoru.

Koncem 19. století se matematika stala i naukou o matematických postupech, které jsou při studiu těchto pojmů používány. Bouřlivý rozvoj matematiky ve 20. století pak přinesl novou definici matematiky jako vědy o strukturách. Matematik zkoumá abstraktní číselné struktury, struktury tvarů, zákony pohybu, principy chování a rozhodovaní (řízení), podstatu pravděpodobnosti, atd.

Všechny struktury mohou byt reálné nebo uměle sestavené, zjevné nebo skryté, statické nebo dynamické, kvalitativní nebo kvantitativní, ryze účelové nebo vymyšlené jen tak pro zábavu. Jejich podstata vychází ze světa, který nás obklopuje, z hlubin prostoru a času i z labyrintu lidské mysli.

Slovo matematika vzniklo z řeckého $\mu\alpha\theta\eta\mu\alpha\tau\iota\kappa o\zeta$ (mathematikós) = milující poznání a $\mu\alpha\theta\eta\alpha$ (máthema) = věda, vědění, poznání. Takže je to věda pro všechny, kteří milují poznání .

Fakta matematiky jsou právě tak užitečná, jako fakta kterékoli jiné vědy. Bez ohledu na to, jak nesrozumitelná se zprvu mohou zdát, dříve či později si naleznou cestu zpět k aplikacím.

Tak například fakta teorie grup se mohou zdát být abstraktní a odtažitá, ale praktické aplikace teorie grup byly početné a vyskytovaly se takovými způsoby, které nikdo nemohl očekávat.

Fakta dnešní matematiky jsou odrazovým můstkem pro vědu zítřka. Matematická teorie začíná definicemi a odvozuje své výsledky na základě všeobecně uznaných odvozovacích pravidel. Každý fakt matematiky, má-li být uznán za pravdivý, musí být začleněn do axiomatické teorie a formálně dokázán.

Axiomatický výklad je v matematice nezbytný, protože fakta matematiky, na rozdíl od fakt fyziky, nemohou být podrobena experimentálnímu ověřování. Axiomatická metoda matematiky je jedním z největších úspěchů naší kultury. Je to však jen metoda.

Zatímco se jednou objevená fakta matematiky nikdy nemění, metoda, jíž byla tato fakta ověřena, se v minulosti mnohokrát změnila a bylo by šílenstvím očekávat, že k takovým změnám nebude v budoucnu opět docházet.

8.4.2 Názory

Teprve koncem devatenáctého století byla dokončena přesná teorie reálných čísel. Ještě i dnes, navzdory jednoduché představě o reálných číslech jakožto o bodech na přímce, je jejich formální (a vysoce abstraktní) definice postrachem pro vysokoškolské studenty.

Podobně se obtížně zaváděla záporná čísla, která si dnes představujeme jako body ležící nalevo od nuly. Jejich zavedení ale matematici odolávali až do poloviny 19. století.

Mnozí lidé dnes mají problém s představou komplexních čísel, která obsahují odmocninu ze záporného čísla –přestože je k dispozici jednoduchý intuitivní obraz komplexních čísel jakožto bodů v rovině.

V dnešní době pracuje s reálnými, komplexními a zápornými čísly bez problémů dokonce i mnoho nematematiků. A to přesto, že tato čísla představují vysoce abstraktní koncepty a s počítáním mají jen velmi málo společného. A to navzdory tomu, že se v našem každodenním životě s žádnými konkrétními příklady iracionálních reálných čísel nebo s číslem obsahujícím odmocninu z čísla -1 nesetkáváme.

V 18. století způsobil v geometrii odborníkům i nematematikům podobné obrovské koncepční potíže objev existence jiných geometrií než té, kterou popsal Eukleides ve své slavné knize Základy. Myšlenka „neeukleidovských geometrií“ se dočkala obecného přijetí až v devatenáctém století. Ke smíru došlo bez ohledu na to, že náš každodenní svět je výhradně eukleidovský.

S každým novým koncepčním skokem si museli i matematici zvykat na nové myšlenky a přijmout je za součást celkového obecného rámce, na jehož základech dělají svou práci.

Ještě nedávno byla rychlost pokroku v matematice taková, že zaujatý pozorovatel stačil zažít jednu změnu dříve, než přišla další. To se však drasticky změnilo.

Abychom porozuměli Riemannově hypotéze (tedy prvnímu problému na seznamu), musíme bezpečně ovládat nejen komplexní čísla (a jejich aritmetiku) a pokročilou matematickou analýzu, ale musíme navíc ještě pochopit, jak sečíst nekonečně mnoho (komplexních) čísel a jak mezi sebou nekonečně mnoho (komplexních) čísel vynásobit.

Takové znalosti jsou dnes již téměř výhradně výsadou těch, kdo vystudovali matematiku na univerzitě. Jen oni mohou vnímat Riemannovu hypotézu jako jednoduché tvrzení, asi tak jako průměrný člověk chápe Pythagorovu větu[2].

- Kdo kárá vznešenou moudrost matematiky, živí se bludem. (Leonardo da Vinci)
- Tvrdím však, že v každé jednotlivé nauce o přírodě lze najít jen tolik skutečné vědy, kolik je v ní matematiky. (Immanuel Kant)
- Matematika učí: nepřehlížejte nuly! (Gabriel Laub)

[2]Zdroj: http://fyzmatik.pise.cz/14-matematicke-problemy-tisicileti.html

- Vždy se směji, když někdo říká, že není chytrý na matematiku, ale že je chytrý na dějepis nebo na cokoliv jiného. Smutná pravda je, že kdo není chytrý na matematiku, není chytrý vůbec. (Miloš Čermák)

Výuka matematiky[3]:

V 50. letech : Dřevorubec prodá náklad dřeva za 100 dolarů. Náklady na produkci jsou 4/5 ceny. Jaký je jeho zisk?

V 70. letech : Dřevorubec prodá náklad dřeva za 100 dolarů. Náklady na produkci jsou 4/5 ceny neboli 80 dolarů. Jaký je jeho zisk?

V 80. letech : Dřevorubec prodá náklad dřeva za 100 dolarů. Náklady na produkci jsou 80 dolarů. Docílil zisku? Ano nebo Ne?

V 90. letech : Dřevorubec prodá náklad dřeva za 100 dolarů. Náklady na produkci jsou 80 dolarů a jeho zisk je 20 dolarů. Vaše zadání: Podtrhněte číslo 20.

Po roce 2000 : Dřevorubec porazí krásný les, protože je sobecký, bezohledný a nezajímá ho ochrana životního prostředí zvířat ani ochrana našich lesů. Činí tak proto, aby dosáhl zisku 20 dolarů. Co si myslíte o tomto způsobu obživy? Námět na celotřídní diskusi po odpovědi na tuto otázku: Jak se cítí ptáci a veverky, když dřevorubec podřezává jejich domovy? (Žádná odpověď není špatná, neobávejte se ale svobodně vyjádřit své pocity, tedy hněv, rozhořčení, pocit méněcennosti, bezmoci atd.) Pokud budete po skončení zkoušky potřebovat akční poradu, jsou vám k dispozici poradci, kteří vám pomohou se přizpůsobit zpět do reálného světa.

v roce 2050 :

■ پرُد ن د زَ ١٠٠ دُلَر نَ پرُدُكِ جسُا ٤/٥ غِني. لَك جِ جِهُ زِسك؟

Česká varianta vtipu[4]:
Přesto však nejen mnozí maturanti, ale i mnozí absolventi základní školy dosud zvládají trojčlenku a někteří i znají pojem procenta. Géniové mezi maturanty pak jsou schopni rozlišovat i mezi procentem a procentním bodem.

1960 : Státní podnik prodává náklad dřeva za 10000 korun. Výrobní náklady jsou čtyři pětiny ceny. Jaký je přínos pro naše socialistické hospodářství?

1980 : Státní podnik vymění množinu $\mathcal{D}$ dřeva za množinu $\mathcal{P}$ peněz. Udělej 10000 teček představujících prvky množiny $\mathcal{P}$. Množina $\mathcal{N}$ výrobních nákladů obsahuje o 2000 bodů méně než množina $\mathcal{P}$. Znázorni množinu $\mathcal{N}$ jako podmnožinu množiny $\mathcal{P}$ a ukaž množinu zisku.

2015 : Dřevorubec vykácí část lesa a na půlku nasází řepku. Náklady na práci mu zaplatí grant ministerstva zemědělství. Řepku prodá Babišovi za 1000 euro. Z Bruselu dostane 1500 euro za neobdělání druhé půlky. Co udělá v příštím roce?

[3] Zdroj: http://neviditelnypes.lidovky.cz/chtip-vyuka-matematiky-0v1-/p_zabava.aspx?c=A150618_222500_p_zabava_wag

[4] ... dtto

Některé další citace a naše komentáře

- Angličané, na rozdíl třeba od Australanů, přestávají věřit v účinnost tržních sil ve vzdělávání. (R.Kvačková,LN 10.3.2015)

  Jenom Angličané? U nás tomu věří většina politiků a ti, kteří získali vzdělání „netržně“. A také různé více či méně pochybné vzdělávací agentury. Co to je trh se vzděláním? Když nebude poptávka po matematice, tak nebude ani nabídka.

- Proč českým dětem matematika nejde? A čím to je, že se chlapcům v testech daří výrazně lépe? Zaměřujeme se především na práci se slabšími žáky nebo těmi, kteří nemají matematiku v oblibě. (B.Cihelková, LN 10.3.2015)

  Co jim vlastně nejde? Matematika nebo je škola nepřipravila dobře k vyplňování testů? Specifický typ zvrácenosti. Nikoho nenapadne z tělesně zanedbaného nebo baculatého staršího adepta vychovávat sprintera nebo tenistu.

Co se můžeme naučit od Hongkongu, Koreje a Singapuru, od států, které v posledních letech obsazují první místa v mezinárodních srovnávacích testech? Podle nové americké studie (2010) zaměřené na srovnávání matematických standardů v těchto zemích se tu výuka matematiky v 1. až 6. ročníku zaměřuje zejména na čísla, metody měření a geometrii a klade jen malý důraz na analýzu dat a algebru. Žáci rovněž probírají lekce v logickém sledu tak, že probírané téma navazuje na předchozí, což konceptuálně posiluje vědomosti žáků, uzavírá studie.

Kdo si myslí, že nedostatek specialistů doplníme importem mladých mozků z Číny nebo Koreje, je idiot.

Žijeme v zemi, kde se zrušení povinné maturity z matematiky bere jako součást demokratických svobod zavedených s novým režimem. Součástí maturity by měla být povinná zkouška z matematiky. Vždycky mě překvapuje, že si někdo může myslet něco jiného. Jenže takových lidí je překvapivě mnoho.

Berou matematiku jako nějakou formu útlaku nebo dokonce druh diskriminace. Koho? Přece těch, kteří „na matematiku“ nejsou dost chytří. Ale tím mohou obalamutit jen oni sami sebe. My ostatní víme, že je to spolčení hloupých, líných nebo prostě pohodlných.

> „Hrůzu vzbuzují názory typu: matematiku jsem v životě k ničemu nepotřeboval. Tyto názory pramení pravděpodobně i z toho, že samotní učitelé matematiky sami neví, k čemu je matematické vzdělání zapotřebí.“
>
> (M.Čermák, LN 1.2.2010)

Lze jen souhlasit s celým článkem. Ale má to háček. Co vlastně rozumět matematikou, ze které by se mělo maturovat? Je to jazyk potřebný k formulaci technologických a přírodovědných otázek a také ke zkoumání vztahů a struktur? Je to prostředek hledání odpovědí na formulované otázky.

Maturant z matematiky by měl prokázat svoje kompetence tím, že umí psát (=formulovat), číst (=rozumět), počítat (=pracovat se strukturami), myslet (=argumentovat).

Nabízíme jeden test:

Přiřaďte správné odpovědi:

1. $x^2 + y^2 = r^2$
2. $a^2 + b^2 = c^2$
3. $š^2 + č^2 = ž^2$

1. Pythagorova věta
2. vtip
3. rovnice kružnice
4. něco jiného

(správná odpověď je 4): Proč? Protože 1), 2), 3) to nejsou!)

Zpravodaj JČMF: 28.listopadu 2014 se konala conference JČMF „Moderní technologie ve výuce fyziky“.

Z kontextu informací je patrné, že moderními technologiemi se rozumí tablety, iphony, snad i mobily. Poněkud se děsíme toho, až se takové „conference“ budou organizovat i pro výuku matematiky, chemie, biologie. Nicméně tato zmínka potvrzuje náš názor, že zmodernizování výuky, resp. spíše kontinuální modernizování, je možné pouze když budeme mít zmodernizované učitele.

8.4.3 Potíže s nekonečnem

Nekonečno je pojem dosti zvláštní a značně ošidný. Myslet nekonečno je dobrodružstvím. V matematice má podobu nekonečna potenciálního a je vyjádřením procesu nekonečnosti jako trvale možného pokračování. Leckdy má ale také podobu nekonečna aktuálního (vystiženého především v teorii množin), v němž nekonečno je uchopeno vcelku jako hotové a je objektem našich dalších manipulací jako kterýkoliv jiný objekt naší zkušenosti.

Záludností nekonečna a vztahů diskrétního a spojitého si byli lidé vědomi odedávna. Jsou vyjádřeny např. v Zenonových paradoxech (želva, letící šíp, ap.). Staří Řekové se nekonečna báli.

Gauss vyjádřil svůj odmítavý postoj slovy: „Protestuji proti použití nekonečných velikostí jako skutečného celku, to není v matematice dovoleno. Nekonečno je jen způsob mluvy...“.

Z dalších uveďme ještě Kroneckera: „Celá čísla stvořil Bůh, vše ostatní je dílo člověka“ a Poincarého: „Aktuální nekonečno neexistuje...“

Základy teorie nekonečných množin a tím i teorie aktuálního nekonečna položil před koncem 19. století *Georg Cantor, 1845-1918).* Bohatstvím struktur zde vytvořených byli matematici tak fascinováni, že *Hilbert* vyjádřil přesvědčení: „Nikdo nás nevykáže z ráje, jenž pro nás Cantor vytvořil.“

8.5 Samé prostory kolem nás

Nejvíce používaný termín pro „vesmír“ mezi starověkými řeckými filozofy od dob Pythagora byl $\tau o \pi \alpha \nu$ (všechno), definované jako celek ($\tau o\ o \lambda \nu$) a prostor ($\tau o\ \kappa \epsilon \nu o \nu$). Další synonyma vesmíru u starověkých řeckých filozofů byla $\kappa o \sigma \mu o \zeta$ (což znamenalo svět, kosmos) a $\varphi \upsilon \sigma i \zeta$ (jež původně znamenalo živou přírodu a z něhož pochází slovo fyzika).

Podobná synonyma se vyskytují v latině (totum, mundus, natura) a přežila i v moderních jazycích, např. německá slova Das All, das Weltall a die Natur pro vesmír.

Podobná synonyma jsou také v angličtině, jako everything (v teorii všeho), cosmos (v kosmologii), world (hypotéza mnoha světů) nebo nature (přírodní zákony a přírodní filozofie).

Seznam obrázků

Seznam tabulek

Seznam tématických bloků

Seznam příkladů