UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Petr Čech

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Aplikace demonstrující práci s nadreálnými čísly

Bakalářská práce

2024 Petr Čech

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr Čech**
Osobní číslo:	**I20080**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace demonstrující práci s nadreálnými čísly**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Americký matematik Jonh Horton Conway navrhl nový způsob konstrukce čísel. Jeho idea fascinovala počítačového vědce Donalda Knutha, který pro tato čísla v r. 1974 navrhl název nadreálná čísla. Knuth konstrukci těchto čísel popsal v povídce, ve které mezi sebou hovoří dva mladí milenci. Knuthovým záměrem bylo v novele odhalit pro čtenáře zásady, techniky, filosofii, krásu matematiky a vášně při jejím objevování, které nejsou předkládány ve škole. Nadreálná čísla souvisí s teorií her. Při konstrukci má důležitou roli indukce.
Cílem práce je prostudovat systém vzniku nadreálných čísel a napsat aplikaci, která jejích konstrukci provede.

Rozsah pracovní zprávy: **35-45**
Rozsah grafických prací: **5**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

*KNUTH, Donald Ervin. Surreal numbers: how two ex-students turned on to pure mathematics and found total happiness : a mathematical novelette. Reading, Mass.: Addison-Wesley Pub. Co., 1974. ISBN 0-201-03812-9.
*KNUTH, Donald E. Nadreálná čísla. Z anglického originálu preložila Helena Nešetřilová. Pokroky matematiky, fyziky a astronomie. Praha: Jednota českých matematiků a fyziků, 23(2) 1978, 66-76. ISSN 0032-2423. Dostupné z WWW: http://dml.cz/bitstream/handle/10338.dmlcz/139649/ PokrokyMFA_23-1978-2_2.pdf
*KNUTH, Donald E. Nadreálná čísla [Pokračování]. Z anglického originálu preložila Helena Nešetřilová. Pokroky matematiky, fyziky a astronomie. Praha: Jednota českých matematiků a fyziků, 23(3) 1978, 130-139. ISSN 0032-2423. Dostupné z WWW: http://dml.cz/bitstream/handle/10338.dmlcz/139926/ PokrokyMFA_23-1978-3_4.pdf
*KNUTH, Donald E. Nadreálná čísla [Pokračování]. Z anglického originálu preložila Helena Nešetřilová. Pokroky matematiky, fyziky a astronomie. Praha: Jednota českých matematiků a fyziků, 23(4) 1978, 187-196. ISSN 0032-2423. Dostupné z WWW: http://dml.cz/bitstream/handle/10338.dmlcz/138565/ PokrokyMFA_23-1978-4_2.pdf
*KNUTH, Donald E. Nadreálná čísla [Dokončení]. Z anglického originálu preložila Helena Nešetřilová. Pokroky matematiky, fyziky a astronomie. Praha: Jednota českých matematiků a fyziků, 23(5) 1978, 246-261. ISSN 0032-2423. Dostupné z WWW: http://dml.cz/bitstream/handle/10338.dmlcz/138207/ PokrokyMFA_23-1978-5_2.pdf

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Aplikace demonstrující práci s nadreálnými čísly jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2024

Petr Čech

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat Mgr. Jaroslavu Markovi, Ph.D. za odborné vedení při vypracování této bakalářské práce. Dále také děkuji své rodině, partnerce a všem přátelům, kteří mě v průběhu studia podporovali a pomáhali mi.

ANOTACE

Bakalářská práce se zaměřuje na návrh a implementaci webové aplikace demonstrující práci s nadreálnými čísly. Uživateli je představena množina nadreálných čísel, jejich základní vlastnosti, konstrukce dalších čísel a základní aritmetické operace s nimi. Práci s nadreálnými čísly si uživatel může sám vyzkoušet. Webová aplikace je jednoduchá jednostránková aplikace pracující na straně klienta.

KLÍČOVÁ SLOVA

nadreálná čísla, pseudo-číslo, narozeninové číslo, transfinitní čísla, infinitesimální čísla, nadkomplexní čísla

TITLE

Application demonstrating work with Surreal numbers

ANNOTATION

The bachelor’s thesis focuses on the design and implementation of a web application demonstrating work with surreal numbers. The user is presented with the set of surreal numbers, their basic properties, construction of more numbers and basic arithmetic operations on them. The user can try working with surreal numbers. The web application is a simple single-page client-side application.

KEYWORDS

surreal numbers, pseudo-number, birthday number, transfinite numbers, infinitesimal numbers, surcomplex numbers

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

Zkratka	Význam
HTML	HyperText Markup Language
JS	JavaScript
CSS	Cascading Style Sheets
OOP	Objektově Orientované Programování
OS	Operační Systém
SEO	Search Engine Optimisation

ÚVOD

Cílem této bakalářské práce je návrh a implementace webové aplikace demonstrující práci s nadreálnými čísly.

Aplikace je rozdělena do dvou vrstev. Datová vrstva obsahuje implementace datových struktur potřebných pro práci s nadreálnými čísly a umožňuje s nimi provádět základní aritmetické operace. Nejdůležitějším i nejkomplexnějším algoritmem této vrstvy je algoritmus hledání hodnoty daného nadreálného čísla.

Vrstva uživatelského rozhraní umožňuje uživateli vyzkoušet si interakci s nadreálnými čísly pomocí jednoduchých webových widgetů. Aplikace uživatele postupně zasvěcuje do tématu nadreálných čísel a souběžně s tím mu demonstruje, jak právě získávané znalosti aplikovat na práci s nadreálnými čísly.

1 TEORETICKÁ ČÁST

První definice nadreálných čísel vyplynula z výzkumu matematika Johna Hortona Conwaye, který jim ale tehdy říkal prostě čísla (Schleicher 2013, Gonshor 1986, s. 1). Jméno nadreálná jim dal až v roce 1974 americký matematik Donald Ervin Knuth ve své noveletě, v níž v kulisách matematického dobrodružství dvojice hlavních hrdinů postupně z naprostého minima buduje obor nadreálných čísel a objevuje jejich zákonitosti (Knuth, 1974).

1.1 Definice nadreálných čísel

Nadreálné číslo x definujeme jako:

$$x = \{X_L | X_R\}, X_L \not\geq X_R$$

Nadreálné číslo x je tedy uspořádaná dvojice množin nadreálných čísel X_L a X_R. Množině X_L můžeme říkat také *levá množina* daného čísla, množině X_R zase *pravá množina*. Žádný prvek množiny není větší ani roven žádnému prvku množiny X_R. Čili všechny prvky množiny X_L jsou menší než všechny prvky množiny X_R (Knuth, 1974, s. 10).

K porovnávání množin stačí porovnávat největší a nejmenší prvek každé množiny. Když x_{Lmax} je největší prvek množiny X_L a x_{Rmin} je nejmenší prvek množiny X_R, tak např. $x_{Lmax} < x_{Rmin} \Leftrightarrow X_L < X_R$ a tak dále pro všechna porovnání (Knuth, 1974, s. 22-23).

1.2 Pseudo-čísla

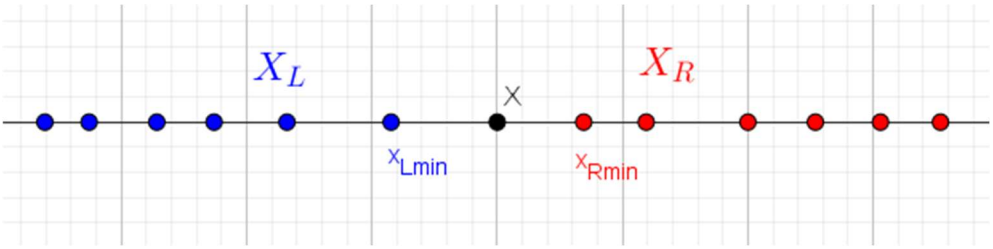
Pokud bychom měli $x = \{X\_L \mid X\_R\}$, ale $X_L \geq X_R$, a tedy by byla porušena předchozí podmínka, x není nadreálné číslo, ale tzv. *pseudo-číslo*. Pseudo-čísla jsou sice v jistém smyslu podobná číslům, ale nemají úplně stejné vlastnosti a chování (Knuth, 1974, s. 79).

1.3 Konstrukce nadreálných čísel

Čísla, která jsou v levé a pravé množině, určují, jaké číslo z množiny vznikne. Základní vztah hledaného čísla a množin, ze kterých je tvořeno:

$$X_L < x < X_R$$

To znamená, že hledané číslo x je větší než všechna čísla v X_L, ale menší než všechna čísla v X_R, leží tedy mezi čísly z X_L a čísly z X_R (Knuth, 1974, s. 30).



Obrázek 1 *Ukázka vzájemné pozice čísla a množin (zdroj vlastní)*

To napovídá první způsob, jak zkonstruovat kterékoli číslo $x = \{X_L|X_R\}$: do množiny X_L dosadíme všechna (tj. všechna existující) čísla menší než x a do množiny X_R dosadíme všechna (tj. všechna existující) čísla větší než x.

Vzhledem k rekurzivní definici nadreálných čísel jsou konstruována postupně. První nadreálné číslo zkonstruujeme ze dvou prázdných množin:

$$x = \{\emptyset|\emptyset\} = 0$$

Nadreálné číslo $x = \{\emptyset|\emptyset\}$ je ekvivalentní číslu $x = 0$. Tedy x je pořád to samé číslo, které ale má dvě různé reprezentace, $\{\emptyset|\emptyset\}$ a 0. Aby se nepletly, rozlišuje se *tvaru* a *hodnotě* nadreálného čísla. Např. hodnotou čísla $\{\emptyset|\emptyset\}$ je 0 a jedním z tvarů čísla 0 je $\{\emptyset|\emptyset\}$.

Číslo 0 umožňuje zkonstruovat další nadreálná čísla (Knuth, 1974, s. 11):

$$y = \{0|\emptyset\} = 1$$

$$z = \{\emptyset|0\} = -1$$

Čísla 1 a -1 umožňují zkonstruovat ještě další nadreálná čísla (Knuth, 1974, s. 44):

$$a = \{\emptyset|-1\} = -2$$

$$b = \{-1|0\} = -\frac{1}{2}$$

$$c = \{0|1\} = \frac{1}{2}$$

$$x = \{1|\emptyset\} = 2$$

1.4 Genealogie nadreálných čísel

Množina konstruovaných nadreálných čísel postupně roste: v prvním kroku vznikla jen 0, v dalším kroku přibyla -1 a 1, z nich v dalším kroku vznikla -2, $-\frac{1}{2}$, $\frac{1}{2}$ a 2 apod. Nadreálná čísla vznikají v generacích. *Generace* je množina všech číslech, která vznikla ve stejném kroku.

Každou generaci můžeme očíslovat: první je generace 0, pak generace 1, poté generace 2 atd. Přitom platí, že číslo generace je stejné jako hodnota největšího nadreálného čísla x_{max} dané generace: v generaci 0 bylo největší číslo $x_{max} = 0$, v generaci bylo největší číslo $x_{max} = 1$… (Knuth, 1974, s. 105).

Místo o generacích můžeme také mluvit o narozeninových číslech. *Narozeninové číslo* nadreálného čísla x, matematicky zapsané jako $\delta(x)$, je číslo generace, ve které se číslo x poprvé objevilo, a tedy největší nadreálné číslo ze stejné generace jako x. Všechna čísla dané generace mají stejná narozeninová čísla: $\delta(0) = 0$, $\delta(-1) = \delta(1) = 1$, a $\delta(-2) = \cdots = \delta(2) = 2$ atd.

Číslo x je *mladší* než y, když $\delta(x) < \delta(y)$, *starší* když $\delta(x) > \delta(y)$ a *stejně staré* když $\delta(x) = \delta(y)$,. V množinách také můžeme hledat, které číslo je nejstarší (nejdříve narozené, $\delta(x)_{min}$) anebo nejmladší (nejpozději narozené, $\delta(x)_{max}$) (Tøndering, 2019, s. 12-13).

Do každé generace δ přibyde 2^{δ} čísel, takže po každém kroku (generaci δ) existuje celkem nadreálných čísel.

Když seřadíme všechna dosud vytvořená čísla $x_1 < x_2 < \cdots < x_n$, po dalším kroku se všechna nová čísla zařadí takto mezi ně: $\{\emptyset|x_1\} < x_1 < \{x_1|x_2\} < x_2 < \{x_2|x_3\} < \cdots < \{x_{n-1}|x_n\} < x_n < \{x_n|\emptyset\}$. Mezi každými dvěma původními čísly x_i a x_{i+1} se nachází nové číslo, které má tvar $\{x_i|x_{i+1}\}$ a každé původní číslo je obklopeno novými čísly (zleva i zprava právě jedním) ve tvaru $\{x_{i-1}|x_i\}$ a $\{x_i|x_{i+1}\}$ (kde x_{-1} a x_{n+1} jsou $\emptyset$). Nová čísla tedy zaplňují volný prostor mezi těmi původními (Knuth, 1974, s. 36).

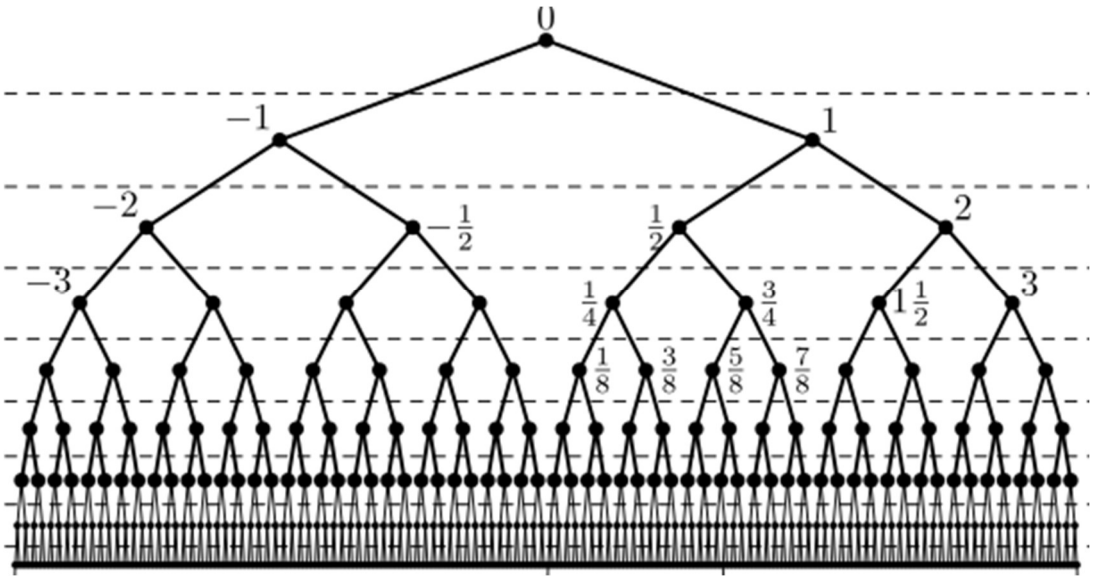
1.5 Kanonický tvar

Čísla $\{-1|1\}, \{-2|2\}, \ldots$ mají sice různé tvary, ale stejnou hodnotu 0 . Každé nadreálné číslo lze zapsat nekonečně mnoho ekvivalentními způsoby (Knuth, 1974, s. 36-37). Většinou se, pokud není řečeno jinak, používá tzv. *kanonický tvar*.

Kanonický tvar čísla $x = \{X_L|X_R\}$ je takový, že ze všech možných x_L a x_R vybereme buď $\emptyset$, nebo to číslo, které je nejstarší (neboli které má mezi všemi možnými čísly nejmenší narozeninové číslo δ). Zároveň je to ten tvar, ve kterém bylo dané číslo poprvé vytvořeno.

1.6 Algoritmus výpočtu hodnoty nadreálného čísla

Pokud vypíšeme pod sebe jednotlivé po sobě jdoucí generace nadreálných čísel tak, aby pokud platí $x_1 < x_2$ bylo x_1 vlevo od x_2, vynikne stromová struktura nadreálných čísel (Ehrlich, 2012, s. 9).



Obrázek 2: *Stromová struktura nadreálných čísel (Lánský, 2012)*

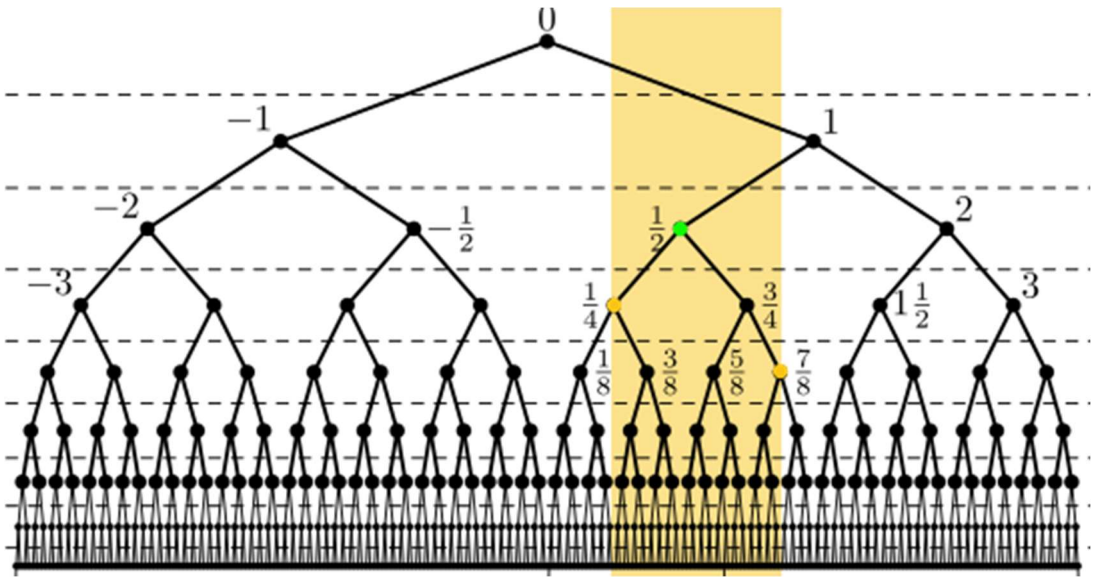
Nadreálná čísla vytváří strukturu binárního vyhledávacího stromu. *Strom* představuje strukturu propojených prvků, kde každý prvek má jednoho přímého předka (nebo žádného, viz číslo 0; takovému prvku se říká *kořen* a každý strom má právě jeden) a určitý počet přímých potomků. *Binární* strom znamená, že každý prvek má až dva potomky.
U *vyhledávacích* binárních stromů rozlišujeme potomky nalevo a napravo, přičemž všichni potomci prvku x nalevo mají menší hodnotu než x a všichni potomci napravo mají větší hodnotu než x (možnost duplikátních prvků nemusíme brát na zřetel, protože takový případ u nadreálných čísel nehrozí). Tuto vlastnost můžeme zapsat jako:

$$X_L < x < X_R$$

Což odpovídá základní vlastnosti hodnoty nadreálných čísel.

Ze způsobu konstrukce stromu nadreálných čísel vyplývá, že pro každé nadreálné číslo platí, že všichni jeho potomci ve stromové struktuře jsou mladší a všichni jeho předci ve stromové struktuře jsou starší.

Pro číslo $x = \{x_L|x_R\}$ platí, že x je *nejstarší* číslo takové, že $x_L < x < x_R$ (Knuth, 1974, s. 58). Snadno lze tuto hodnotu najít graficky na stromové struktuře nadreálných čísel. Stačí vzít čísla x_L a x_R a v prostoru mezi nimi vyhledat nejstarší číslo.



***Obrázek 3**: Stromová struktura nadreálných čísel s vyznačením hledání čísla (Lánský, 2012)*

„Nejstarší číslo mezi dvěma čísly“ může být buď starší než x_L i x_R – potom bylo vytvořeno už v některé z předchozích generací – nebo mladší než x_L i x_R a proto teprve musí být zkonstruováno do následující generace. V takovém případě bude forma $\{x_L|x_R\}$ kanonickým tvarem x. Existují čtyři možnosti, jak vytvořit nové nadreálné číslo:

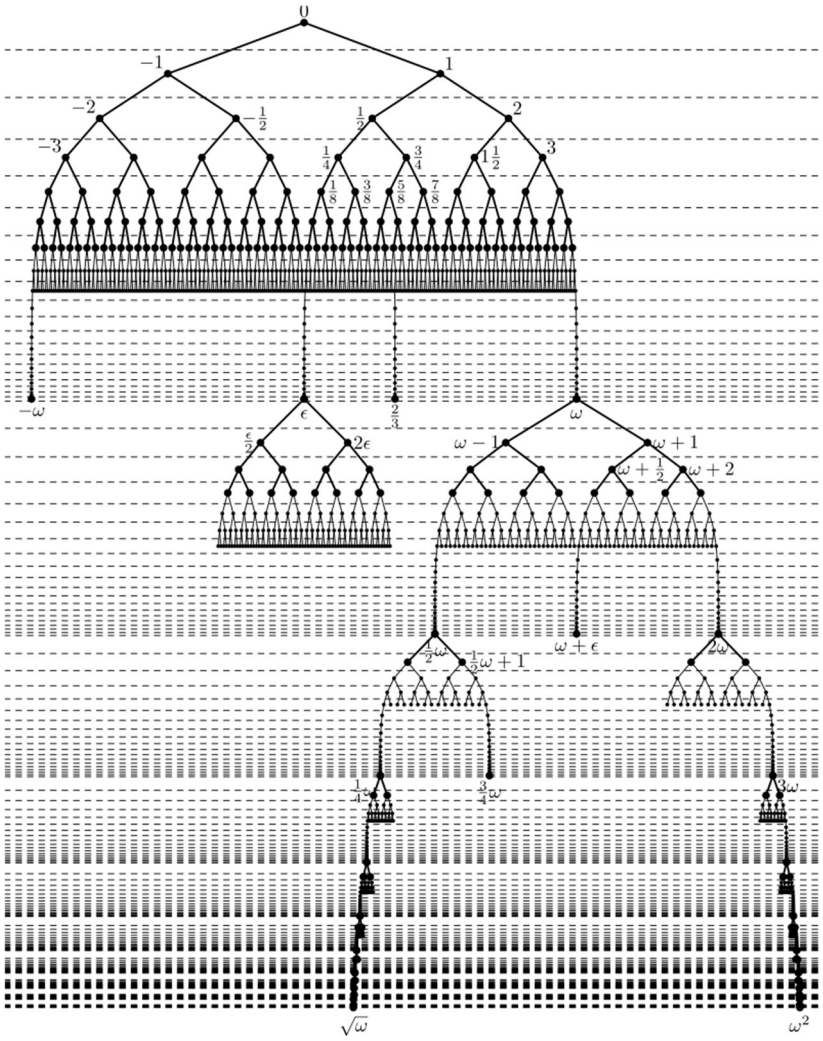
- Triviální případ $\{\emptyset|\emptyset\} = 0$, tedy prostě „nejstarší číslo“.
- $\{x_L|\emptyset\}$, tedy „nejstarší číslo větší než x_L“, jehož hodnota je $x = x_L + 1$
- $\{\emptyset| x_R\}$, tedy „nejstarší číslo menší než x_R“, jehož hodnota je $x = x_R - 1$
- $\{x_L|x_R\}$, když ani jedno z x_L ani x_R není prázdná množina, pak hodnota čísla $x_L < x < x_R$ leží přesně mezi x_L a x_R a vypočítáme ji vzorcem $x = \frac{x_L+x_R}{2}$, např. $\{5|6\} = \frac{5+6}{2} = \frac{11}{2}$ (Knuth, 1974, s. 59).

Konečným sledem těchto operací ovšem mohou vznikat pouze celá čísla nebo *dyadické zlomky*, tedy čísla ve tvaru $\frac{a}{2^b}$, kde a a b jsou celá čísla (Knuth, 1974, s. 92). K rozšíření množiny nadreálných čísel je třeba umožnit, aby X_L i X_R mohly být nekonečné množiny a nekonečné posloupnosti (Knuth, 1974, s. 93-94).

1.7 Reálná čísla

Pro konstrukci reálného čísla $r = \{R_L|R_R\}$ je potřeba najít takové množiny dyadických zlomků R_L a R_R, aby $R_L < r < R_R$. Hledané dyadické zlomky budou členy dvou nekonečných posloupností: jedné, která se blíží k limitě r *zdola* (pro prvky R_L), a druhé, která se blíží k limitě r *shora* (pro prvky R_R). Například:

$$\pi = \{\{3{,}125;\ 3{,}140625;\ 3{,}14111328125; \ldots\}|\ \{3{,}5;\ 3{,}25;\ 3{,}1875;\ 3{,}15625; \ldots\}\}$$



Obrázek 4 *Celá stromová struktura nadreálných čísel (Lánský, 2012)*

1.8 Transfinitní čísla

Množiny, z nichž vytváříme nadreálná čísla, mohou i divergovat do nekonečna. Hodnotu výsledných čísel nelze vyjádřit žádným ryze reálným číslem. Zavedeme tedy nová čísla:

$$\{\{1;\ 2;\ 3;\ 4; \ldots\}|\ \emptyset\} = \omega$$

$$\{\emptyset|\ \{-1;\ -2;\ -3;\ -4; \ldots\}\} = -\omega$$

Kde ω je „číslo větší než všechna reálná čísla“ a $-\omega$ je „číslo menší než všechna reálná čísla“ (Knuth, 1974, s. 95). Pokud bychom prováděli konstrukci postupně od 0, k $\pm\omega$ bychom se dostali až po nekonečně mnoha krocích. Těmto číslům se říká transfinitní, což znamená „za konečnem“, „za“ všemi konečnými čísly. Z těchto čísel lze tvořit ještě další nadreálná čísla, např. „čísla větší než ω“ $\{\omega|\emptyset\} = \omega + 1$, $\{\omega + 1|\emptyset\} = \omega + 2$, nebo „čísla menší než -ω“ $\{\emptyset|-\omega\} = -\omega - 1$.

Dále lze zkonstruovat také „čísla větší než všechna reálná čísla, ale menší než ω“ a „čísla menší než všechna reálná čísla, ale větší než $-\omega$“:

$$\{\{1;\ 2;\ 3; \ldots\}|\ \omega\} = \omega - 1$$

$$-\omega + 1 = \{-\omega|\{-1; -2; -3; \ldots\}$$

Takto můžeme pokračovat $-\omega + 2 = \{-\omega + 1|\{-1; -2; -3; \ldots\}$, $\omega - 3 = \{\{1;\ 2;\ 3; \ldots\}|\ \omega - 2\ \}$ atd. Při skládání transfinitních čísel se jejich reálné části chovají funguje podobně stejně jako u konečných čísel, např. $\{\omega + 1|\omega + 2\} = \omega + \frac{3}{2}$ (Knuth, 1974, s. 99-100).

Když umožníme divergovat do nekonečna i transfinitním číslům, získáme:

$$\{\{\omega + 1;\ \omega + 2;\ \omega + 3; \ldots\}|\ \emptyset\} = 2\omega$$

$$\{\{1;\ 2;\ 3; \ldots\}\}|\ \{\omega - 1;\ \omega - 2;\ \omega - 3; \ldots\}\} = \frac{1}{2}\ \omega$$

$$\{\emptyset|\ \{-\omega - 1;\ -\omega - 2;\ -\omega - 3; \ldots\}\} = -2\omega$$

$$\{\{-\omega + 1;\ -\omega + 2;\ -\omega + 3; \ldots\}\}|\ \{-1;\ -2;\ -3; \ldots\}\} = -\frac{1}{2}\omega$$

Opět lze se skládáním transfinitních čísel pokračovat dále: $\{\{\omega + 1;\ \omega + 2;\ \omega + 3; \ldots\}|\ 2\omega\} = 2\omega - 1$, $\{\{2\omega + 1;\ 2\omega + 2;\ 2\omega + 3; \ldots\}|\ \emptyset\} = 3\omega$, $\{\{\omega + 1;\ \omega + 2;\ \omega + 3; \ldots\}|\ \{2\omega - 1;\ 2\omega - 2;\ 2\omega - 3; \ldots\}\} = \frac{1}{2}\omega$ (Knuth, 1974, s. 100-101).

Kromě násobků transfinitních čísel lze konstruovat i jejich mocniny a odmocniny (Knuth, 1974, s. 101, 111):

$$\{\{\omega;\ 2\omega;\ 3\omega; \ldots\}|\ \emptyset\} = \omega^2$$

$$\{\{1;\ 2;\ 3; \ldots\}|\ \{\omega;\ \frac{1}{2}\omega;\ \frac{1}{3}\omega; \ldots\}\} = \sqrt{\{\omega\}}$$

$$\{\emptyset|\{\omega;\ 2\omega;\ 3\omega;\ldots\}\} = -\omega^2$$

$$\{\{-\omega;\ -\frac{1}{2}\omega;\ -\frac{1}{3}\omega;\ldots\}|\{-1;\ -2;\ -3;\ldots\}\} = -\sqrt{\{\omega\}}$$

I transfinitní čísla mají svá narozeninové čísla. $\delta(\omega) = \delta(-\omega) = \omega$, protože ω je největší číslo dané generace. Další generace pak mají narozeninová čísla $\omega + 1$, $\omega + 2$,… (Knuth, 1974, s. 105). Narozeninové číslo $\delta(2\omega) = 2\omega$, $\delta(3\omega) = 3$, $\delta(3\omega + 2) = 3\omega + 2$, $\delta(\omega^2) = \delta\left(\sqrt{\omega}\right) = \omega^2$,…

1.9 Infinitesimální čísla

Infinitesimální číslo je „číslo větší než 0, ale menší než všechna (kladná) reálná čísla blížící se 0.“ (Knuth, 1974, s. 96) Hodnotu takového čísla nelze vyjádřit žádným ryze reálným číslem. Zavedeme proto tato čísla:

$$\{0 \mid \{\frac{1}{2};\ \frac{1}{4};\frac{1}{8};\cdots\}\} = \epsilon$$

Podobně pro čísla blížící se 0 zleva:

$$\{\{-\frac{1}{2};\ -\frac{1}{4};-\frac{1}{8};\cdots\}|0\} = -\epsilon$$

Nekonečná množina, která tvoří infinitesimální číslo, se nemusí blížit jen k limitě 0, ale ke kterémukoli číslu, takže lze zkonstruovat třeba $1 - \epsilon$, $-\frac{1}{4} + \epsilon$ a později i $\frac{2}{3} - \epsilon$ či $\omega + \epsilon$.

$$1 - \epsilon = \{\{1 - \frac{1}{2}; 1 - \frac{1}{4}; 1 - \frac{1}{8};\cdots\}|\ 1\ \}$$

$$\omega + \epsilon = \{\omega|\{\omega + \frac{1}{2}\omega + \frac{1}{4}\omega + \frac{1}{8}\omega;\ldots\}\}$$

Číslo ϵ patří do stejné generace jako ω, tedy narozeninová čísla $\delta(\epsilon) = \delta(\omega) = \omega$. Stejně tak $1 - \epsilon$, $-\frac{1}{4} + \epsilon$, ale $\frac{2}{3} - \epsilon$ či $\omega + \epsilon$ mají narozeninové číslo až 2ω.

Také infinitesimální čísla můžeme použít ke konstrukci dalších nadreálných čísel. Mezi 0 a ϵ nalezneme $\frac{1}{2}\epsilon = \{0 \mid \epsilon\}$, mezi ϵ a všemi většími reálnými čísly $2\epsilon = \{\epsilon|\{\frac{1}{2};\ \frac{1}{4};\frac{1}{8};\cdots\}\}$ a podobně $-\frac{1}{2}\epsilon = \{-\epsilon|0\ \}$, $-2\epsilon = \{\{-\frac{1}{2};\ -\frac{1}{4};-\frac{1}{8};\cdots\}|\epsilon\}$, $1 + \frac{1}{2}\epsilon = \{1 \mid 1 + \epsilon\}$, $\frac{1}{4}\epsilon = \{0 \mid \frac{1}{2}\epsilon\}$, $\frac{3}{4}\epsilon = \{\epsilon|\ 2\epsilon\}$ a tak dále (Tøndering, 2019, s. 43). A konečně také:

$$\{0 \mid \{\frac{1}{2}\epsilon;\ \frac{1}{4}\epsilon;\ \frac{1}{8}\epsilon; \ldots\}\} = \epsilon^2$$

$$\{\{\epsilon;\ 2\epsilon;\ 3\epsilon; \ldots\} \mid \{\frac{1}{2};\ \frac{1}{4};\ \frac{1}{8}; \ldots\}\} = \sqrt{\epsilon}$$

A podobně vznikne třeba i $1 - \epsilon^2$ a $-2 - \sqrt{\epsilon}$ (Knuth, 1974, s. 111). Všechna tato čísla patří do stejné generace jako ω^2, takže jejich narozeninové číslo δ je také ω^2.

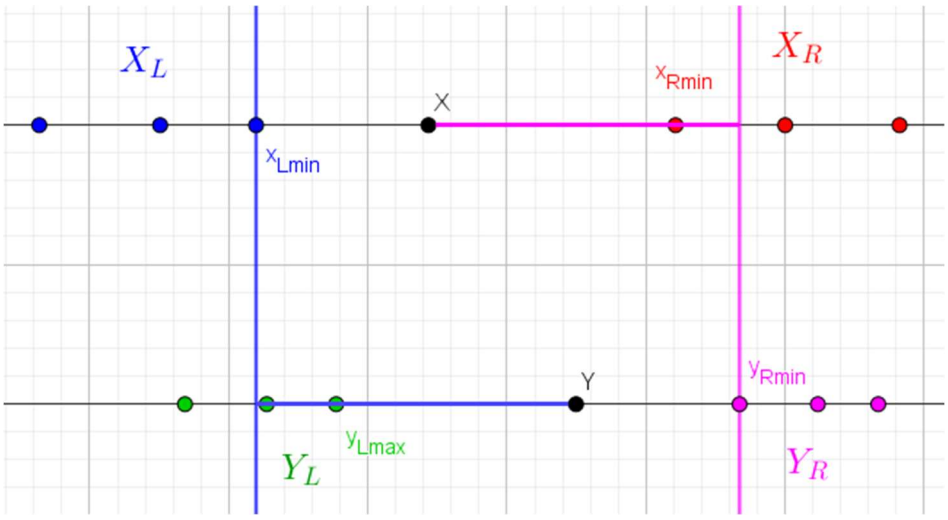
1.10 Porovnávání nadreálných čísel

Nadreálná čísla jsou *největší úplně uspořádanou množinou*. Úplně uspořádaná množina je taková, na které je definována úplná binární relace uspořádání, resp. jejíž každé dva prvky můžeme porovnat. Nadreálná čísla obsahují všechny takové množiny a všechna taková čísla (Bajnok, 2013).

Operace porovnání je pro nadreálná čísla naprosto zásadní. Je definováno jako:

$$x \leq y \Leftrightarrow X_L \not\geq y \wedge x \not\geq Y_R$$

Jinými slovy x je menší nebo rovno y právě tehdy, když žádné číslo množiny X_L není větší ani rovno y a zároveň x není větší ani rovno žádnému číslu Y_R (Knuth, 1974, s. 12).



Obrázek 5: *Ukázka porovnání dvou čísel (zdroj vlastní)*

1.11 Aritmetické operace s nadreálnými čísly

Sčítání

Sčítání je operace se dvěma nadreálnými čísly a , kterou definujeme následovně (Knuth, 1974, s. 51):

$$x\ +\ y = \{X_L + y \cup x + Y_L | X_R + y \cup x + Y_R\}$$

Neboli ke všem prvkům levé množiny x přičteme číslo y a k číslu y přičteme všechny prvky levé množiny x, dáme všechny tyto výsledky do jedné množiny a to bude výsledná levá množina. Stejně tak přičteme čísla k prvkům pravých množin a z nich vytvoříme výslednou pravou množinu. Výsledkem sčítání s prázdnou množinou je zase prázdná množina.

Sčítání nadreálných čísel má stejné chování a vlastnosti jako sčítání reálných čísel: je asociativní, komutativní, a má neutrální prvek 0.

Negace

Negace nadreálného čísla neboli číslo opačné je velice jednoduchou operací:

$$-x = \{-X_R | -X_L\}$$

Tedy prohodíme levou a pravou množinu a všechna čísla v nich vynásobíme -1, takže se obrátí jejich pořadí (Knuth, 1974, s. 52).

Odčítání

Operaci *odčítání* definujeme pomocí sčítání a negace (Knuth, 1974, s. 68).

$$x - y = x + (-y)$$

Rozepsáno do vzorce pro sčítání to vypadá takto:

$$\begin{aligned} x - y &= \{X_L + (-y) \cup x + (-Y_R) | X_R + (-y) \cup x + (-Y_L)\} \\ &= \{X_L - y \cup x - Y_R | X_R - y \cup x - Y_L\} \end{aligned}$$

Čili odčítání provádíme podobně jako sčítání, jen vyměníme množiny X_L a X_R a místo operace sčítání odčítáme.

Násobení

Operace *násobení* dvou nadreálných čísel už není definována tak jednoduše (Knuth, 1974, s. 108):

$$xy = \{X_L y + xY_L - X_L Y_L \cup X_R y + xY_R - X_R Y_R | X_L y + xY_R - X_L Y_R \cup X_R y + xY_L - X_R Y_L\}$$

kde např. $X_N y$ znamená vynásobení všech prvků množiny X_N číslem y a $X_N Y_M$ znamená roznásobení všech prvků množiny X_N se všemi prvky množiny Y_M. Ze všech takových kombinací pak vznikne množina, z níž se skládá výsledné nadreálné číslo. Pokud je kterákoli z množin $\emptyset$, výsledkem je také prázdná množina $\emptyset$.

Násobení nadreálných čísel má stejné chování a vlastnosti jako násobení reálných čísel: je asociativní, komutativní, distributivní, má neutrální prvek 1 a absorbující prvek 0.

Vztah ω a ϵ

Definice násobení umožňuje potvrdit závěr, že (Conway, 1976, s. 12-14):

$$\omega^n \epsilon^n = \omega^0 \epsilon^0 = 1$$

Z toho plyne např.:

Tabulka 1: *Vybrané produkty násobení ω a ε*

$\epsilon = \frac{1}{\omega} = \omega^{-1}$	$\omega = \frac{1}{\epsilon} = \epsilon^{-1}$
$\epsilon^2 = \frac{1}{\omega^2} = \omega^{-2}$	$\omega^2 = \frac{1}{\epsilon^2} = \epsilon^{-2}$
$\sqrt{\epsilon} = \frac{1}{\sqrt{\omega}} = \omega^{-\frac{1}{2}}$	$\sqrt{\omega} = \frac{1}{\sqrt{\epsilon}} = \epsilon^{-\frac{1}{2}}$

Hodnotu čísla $\omega^2 + \omega + \sqrt{\omega} + 1 + \sqrt{\epsilon} + \epsilon + \epsilon^2$ můžeme zapsat také jako $\omega^2 + \omega + \omega^{\frac{1}{2}} + 1 + \omega^{-\frac{1}{2}} + \omega^{-1} + \omega^{-2}$.

1.12 Nadkomplexní čísla

Stejně jako komplexní čísla vznikají rozšířením reálných čísel zavedením imaginární jednotky i, můžeme definovat *nadkomplexní čísla* jako číslo ve tvaru:

$$x = a + bi$$

kde a a b jsou nadreálné koeficienty a i je imaginární jednotka (Alling, 1987, s. 571).

1.13 Využití nadreálných čísel v teorii her

Nadreálná čísla najdou své využití, např. v teorii her. Představme si hru mezi dvěma hráči L a P, např. hru Go (koneckonců právě díky výzkumu konců hry Go byla nadreálná čísla vynalezena (Dokos, 2009, s. 1)). Hra musí splňovat základní kritéria (Dokos, 2009, s. 1-2):

- hra musí být deterministická (nesmí v ní hrát roli náhoda),
- nesmí osahovat skryté informace (hráči si takříkajíc musí „vidět do karet“),
- hráči se musí střídat (nemohou najednou zahrát vícero tahů)
- hráč prohrává, když už nemůže udělat žádný platný tah (podle pravidel).

V některých postaveních hry mívá jeden z hráčů výhodu. Bylo by proto užitečné nějakým způsobem přiřadit každé hře určitou hodnotu, určité číslo. Proto definujeme, že postavení neboli *hra* $H = \{L \mid P\}$, kde L je množina tahů, které v dané pozici může udělat hráč L, tedy množina všech postavení, do kterých se může hra dostat jedním tahem hráče L; P je obdobná množina tahů hráče P (Tøndering, 2019, s. 46). Narozdíl od nadreálných čísel ale hry mohou tvořit i množiny L a P, pro které neplatí $L < P$ (tedy hrou může být nadreálné číslo i pseudo-číslo).

Jak už bylo řečeno, hráč prohrává, když už nemůže udělat žádný platný tah.

- Když $H = \{\emptyset \mid \emptyset\}$, není v daném postavení k dispozici žádný platný tah, takže hráč, který je další na řadě, nemůže nic dělat a prohraje.
- Hru $H = \{0 \mid \emptyset\}$ vyhraje hráč L, ať už je na řadě kdokoli.
- Hru $H = \{\emptyset \mid 0\}$ zase určitě vyhraje hráč P
- Hru $H = \{0 \mid 0\}$ vyhraje ten hráč, který je další na řadě.

(Tøndering, 2019, s. 47)

Za předpokladu dokonalých hráčů, kteří nedělají chyby, lze tato pravidla zobecnit podle porovnání hodnoty H (Knuth, 1974, s. 112):

- $H = 0$ – ten, kdo je další na řadě, prohraje
- $H > 0$ – vyhraje hráč L
- $H < 0$ – vyhraje hráč P
- H – ten, kdo je další na řadě, prohraje (kde H je pseudo-číslo nesrovnatelné s 0).

Některé hry lze rozdělit na nezávislé menší hry G a H, které můžeme zanalyzovat samostatně a celkovou hru pak popsat jako $G + H$.

2 EXPERIMENTÁLNÍ ČÁST

2.1 Použité technologie

2.1.1 Jazyky

HTML, JS, CSS

Webová aplikace je implementována v trojici standartních jazycích: HTML je značkovacím jazykem definujícím obsah a strukturu webových stránek jako sérii vnořených prvků. CSS slouží k popisu, jak by se měly prvky HTML zobrazovat, jak by měly vypadat a jak by měly být rozvržené. JS je skriptovacím jazykem, který určuje, co bude web dělat, jak se bude chovat a jak bude reagovat na různé události (Young, 2024).

2.1.2 Knihovny

Webová aplikace nebyla vyvíjena v žádném frameworku, využívá ovšem dvě JS knihovny:

MathJax

MathJax je open-source platforma pro přístupné zobrazování matematických zápisů. Mezi podporované formáty patří LaTeX, MathML a ASCIImath. Pochází z populárního projektu jsMath vytvořeného Davidem Cervonem v roce 2004. Aktuální verze MathJax 3 byla vydána v roce 2017 (MathJax, 2023)

DradDropTouch

Většina mobilních internetových prohlížečů neimplementuje funkcionalitu „drag and drop“. Tato open-source knihovna „překládá“ události dotyku na události typu „drag and drop“, a zpřístupňuje tak tuto možnost i na mobilních zařízeních. Vytvořil Bernardo Castilho v roce 2016, současná verze pochází z roku 2017 (Castilho, 2016).

2.1.3 Vývojové prostředí

Visual Studio Code je integrované vývojové prostředí od společnosti Microsoft dostupné pro operační systémy Windows, macOS a Linux s vestavěnou podporou technologií JavaScript, TypeScript a Node.js a s možností instalace rozšíření pro širokou škálu dalších technologií, např. PHP, C/C++, C#, Java, Python, Go apod. (Microsoft, 2024)

2.1.4 Ostatní nástroje

GeoGebra

Matematický software s širokým spektrem možností zahrnujícím geometrii, algebru, grafy, tabulky, statistiku a mnoho dalšího (GeoGebra, 2024). V této práci použit pro tvorbu ilustračních obrázků.

2.2 Rozčlenění aplikace

2.2.1 Souborová struktura aplikace

- index.html – hlavní webová stránka aplikace
- surreal/ - složka s dalšími soubory aplikace
    - imgs/ - složka s používanými obrázky
    - maths/ - složka s knihovnou JS skriptů implementujících dané datové struktury
    - view/ - složka pro komponenty používané uživatelským rozhraním
        - textbubble/ - složka s JS a CSS soubory widgetu TextBubble
        - dragndrop/ - složka s JS a CSS soubory widgetu DragNDrop

2.2.2 Rozdělení vývoje aplikace

Vývoj aplikace byl rozdělen do dvou částí: jednak vývoj datových struktur umožňující libovolným programům zacházet s nadreálnými čísly, jednak frontendové uživatelské rozhraní, totiž samotnou webovou aplikaci, která uživatelům vysvětluje a demonstruje, jak pracovat s nadreálnými čísly.

2.3 Datové struktury

2.3.1 Návrh datových struktur

Cílem této části práce bylo vytvořit knihovnu datových struktur, které by vývojářům umožňovaly v libovolných programech zacházet s nadreálnými čísly. K tomu bylo potřeba implementovat jednak datové struktury reprezentující různé druhy čísel, jednak datovou strukturu reprezentující množinu čísel.

Hlavním omezením, se kterým se bylo potřeba vypořádat, je konečnost počítačů: počítačové programy musí běžet v konečném čase a mají k dispozici pouze konečné množství paměti a konečné množství elektrické energie.

Např. desetinná čísla jsou v počítačové paměti uložená v konečném binárním tvaru, což ovšem znamená, že jakákoli nedyadická reálná čísla jako $\frac{1}{3}$ nelze reprezentovat přesně, pouze jako nejbližší dyadický zlomek (Knuth, 1974, s. 93-94). To znemožňuje věrnou a jednoznačnou reprezentaci úplné množiny reálných hodnot. Zároveň je v množině nadreálných čísel ke konstrukci nedyadických reálných čísel potřeba nekonečné množství kroků, což by si vyžádalo významnou úpravu algoritmu jejich konstrukce. Kvůli těmto omezením byla opuštěna snaha vytvořit datové struktury umožňující reprezentaci libovolných reálných hodnot a implementace se zaměřila pouze na dyadické zlomky jako koeficienty.

Nekonečné množství kroků ke své konstrukci potřebují i nadreálná čísla jako ω a ε. Avšak v případě těchto čísel se lze nekonečnu vyhnout: jednak rozdělením hodnot nadreálných čísel do tvaru např. $a\omega + b + c\epsilon$, jednak jazyk JS umožňuje do číselných datových typů ukládat hodnotu ±Infinity, tedy $\pm\infty$, čehož lze využít k reprezentaci nekonečného množství konstrukčních kroků. Objevuje se však další otázka: množina nadreálných čísel obsahuje i hodnoty jako $\omega, \omega^2, \omega^\omega, \omega^{\omega^\omega}, \cdots$, obecně ve tvaru $a_1\omega^{e_2} + a_2\omega^{e_2} + \cdots + a_n\omega^{e_n}$, kde exponenty e mohou být další nadreálná čísla. Tato práce se ale omezila pouze k hodnotám s nejrozvinutějším tvarem maximálně $a\omega^2 + b\omega^{e_1} + c\sqrt{\omega} + d + e\sqrt{\epsilon} + f\epsilon + g\epsilon^2$.
K demonstraci práce s nadreálnými čísly to bohatě stačí, čemuž nasvědčuje i to, že někteří

autoři literatury úvodu do problematiky nadreálných čísel jako např. Knuth (1974) a Tøndering (2019) další mocniny ani nezmiňují.

Datová struktura Dyadic

Tato datová struktura reprezentuje dyadický zlomek s čitatelem n a jmenovatelem 2^e. Podporuje základní operace porovnání, sčítání, opačné hodnoty, odčítání, násobení, průměrné hodnoty a převedení na textový řetězec. Tuto datovou strukturu pak používají ostatní pro reprezentaci dyadických koeficientů.

Datová struktura SurrealValue

Tato datová struktura představuje hodnotu nadreálného čísla vyjádřenou ve výše popsaném vzorci $a\omega^2 + b\omega + c\sqrt{\omega} + d + e\sqrt{\epsilon} + f\epsilon + g\epsilon^2$. Opět podporuje základní operace porovnání, sčítání, opačné hodnoty, odčítání, násobení, průměrné hodnoty a převedení na textový řetězec. Slouží jak dalším datovým strukturám, tak v uživatelském rozhraní k přímému zobrazení hodnot nadreálných čísel a manipulací s nimi.

Datová struktura SurrealNumber

Tato datová struktura reprezentuje celou formu nadreálného čísla $x = \{L \mid R\}$, kde x je hodnota daného nadreálného čísla a L a R jsou levá a pravá množina, z nichž se dané nadreálné číslo skládá. Podporuje základní operace porovnání, sčítání, opačného čísla, odčítání, násobení (konečných čísel) a převedení na textový řetězec. Nejdůležitější schopností této datové struktury je výpočet hodnoty nadreálného čísla ze zadaných množin.

Datová struktura SurrealSet

Tato datová struktura představuje množinu čísel, slouží hlavně pro konstrukci nadreálných čísel a musí podporovat i operace potřebné k základním operacím s nadreálnými čísly. Vzhledem k tomu, že ke konstrukci i dalším operacím s nadreálnými čísly stačí znát minimální a maximální prvek množiny, tato datová struktura obecně počítá pouze s nimi a žádnými jinými. Podporuje základní operace porovnání s prvky druhé množiny (buď jsou všechny prvky množiny menší než všechny prvky druhé množiny, nebo ne), sjednocení množin, opačné množiny (tedy množiny opačných čísel), sčítání, odčítání a násobení jedné nadreálné hodnoty ke všem prvkům množiny, sčítání a odčítání všech prvků se všemi prvky druhé množiny.

Musí být zároveň schopná reprezentovat různé druhy množin, ze kterých se nadreálná čísla skládají: konečné množiny, nekonečné množiny s prvky rostoucími do nekonečna (příp. klesajícími do -∞) i nekonečné množiny blížící se konkrétní hodnotě limity.

Datové struktury odvozené od SurrealSet

Protože pro práci datové struktury SurrealSet stačí znát minimální a maximální hodnoty množiny, nepracuje s žádnými čísly mezi těmito dvěma extrémy. Pro účely datových struktur a jejich algoritmů je to dostačující, ovšem co se týče frontendového uživatelského rozhraní je uživatelsky přívětivější a přehlednější zobrazovat jednotlivé prvky množin.

Proto byly od datové struktury SurrealSet odvozeny následující datové struktury pro jednotlivé typy množin:

- SingleValueSurreal – reprezentující (konečnou) množinu s jediným prvkem
- InfiniteSurrealSet – představující nekonečnou množinu, jejíž prvky rostou do nekonečna (příp. klesajícími do -∞)
- InfinitesimalSurrealSet – reprezentující nekonečnou množinu, jejíž prvky se blíží konkrétní hodnotě limity
- DiminishingTransfiniteSurralSet – představující nekonečnou množinu transfinitních hodnot, jejichž koeficient se blíží 0 (buď zleva, nebo zprava)
- DiminishingInfinitesimalSurrealSet – reprezentující nekonečnou množinu infinitesimálních hodnot, jejichž koeficient se blíží 0 (opět buď zleva, nebo zprava)

Všechny tyto odvozené datové struktury podporují stejné operace jako původní SurrealSet s tím, že přepisují metodu výpisu do textového řetězce na uživatelsky přívětivější podobu.

2.3.2 Implementace datových struktur

Jednotlivé datové struktury byly implementovány jako třídy OOP kvůli snadnému zapouzdření dat a příslušných metod a kvůli možnosti snadné adaptace jednotlivých datových struktur pro konkrétní účely pomocí dědičnosti. Jazyk JS nepodporuje přepisování operátorů (na rozdíl od např. programovacího jazyk C++), proto jsou všechny operace implementovány jako metody jednotlivých tříd. Porovnání je implementováno standardní metodou *x.compareTo(y)*, které vrací 0 pro $x = y$, 1 pro $x > y$ a -1 pro $x < y$.

Dynamické typování jazyka JS si vyžaduje kontrolu datových typů vstupních parametrů. Pokud neodpovídají očekávaným datovým typům nebo jiným způsobem nenaplňují požadavky, je vystavena příslušná výjimka *Invalid argument(s).* Pokud je výsledkem operace objekt dané třídy, je vrácen nový objekt; operace obecně nemění existující objekty.

Specifika implementace třídy Dyadic

Dyadický zlomek je reprezentován dvojicí celých čísel n (=*numerator,* tj. čitatel) a e (=*exponent*, tj. exponent jmenovatele). Všechna celá čísla jsou reprezentována jako $\frac{z}{2^0}$, třebaže např. číslo 4 lze matematicky zapsat i jako $\frac{1}{2^{-2}}$. Tato vlastnost je výhodná pro algoritmy s nadreálnými čísly, které často s celými čísly nakládají specifickým způsobem, a proto se je hodí mít připravené v nejjednodušší podobě. Čitatel také může mít hodnotu ±Infinity, která představuje hodnotu rostoucí do nekonečna (či klesající do -∞). Jinak pokud je nenulový čitatel dyadického zlomku dělitelný dvěma (tedy $n = 2^k m$, kde $n \in \mathbb{N}$ a $m \in \mathbb{Z} \setminus \{0\}$; $2 \nmid m$), je dyadický zlomek převeden do základního tvaru $\frac{m}{2^{e+k}}$.

Kromě již řečených operací třída Dyadic implementuje ještě kontrolu celočíselnosti, kontrolu konečnosti, operace inkrementace a dekrementace celého zlomku $d \pm 1 = \frac{n}{2^e} \pm 1$, operace inkrementace a dekrementace čitatele $Num(d, \pm 1) = \frac{n \pm 1}{2^e}$, převod na celé číslo bližší 0 (Int(d) = $\lfloor d \rfloor$ pro $d \geq 0$ a $\lceil d \rceil$ pro $d < 0$), minimální a maximální číslo ze dvou zadaných a výpis jako číslo s desetinnou čárkou. Všechny operace jsou implementovány na základě jednoduchých matematických rovnic a nerovnic popsaných v komentáři kódu. Třída také obsahuje statické konstanty *Dyadic.zero* pro hodnotu 0 a *Dyadic.infinity* pro hodnotu ∞.

Specifika implementace třídy SurrealValue

Nadreálná hodnota s koeficienty a, b, c, d, e, a f je implementována atributy typu Dyadic pojmenovanými *omegaSquared*, *omega*, *omegaRoot*, *dyadic*, *epsilonRoot*, *epsilon* a *epsilonSquared* v tomto pořadí. V tomto pořadí jsou také seřazeny od nejvyššího po nejnižší řád. V této práci budeme pracovat s následnou notací: $X_{(\mathrm{d})}$ značí koeficient dyadic nadreálné hodnoty X, $X_{(\omega)}$ = koeficient omega, $X_{(\epsilon,\epsilon^2)}$ = koeficienty epsilon a epsilonSquared, $X_{not(\sqrt{\omega})}$ = všechny koeficienty kromě omegaRoot, $X_{(d)} \to 0^+$ = koeficient dyadic se blíží k limitě 0 zprava atd. Zápis ve tvaru např. X = Y: $X_{(d)} = 0$ znamená, že číslo X je stejné jako Y, ale má koeficient dyadic = 0 apod.

Třída SurrealValue podporuje stejné operace jako třída Dyadic, které jsou implementovány jako operace třídy Dyadic s příslušnými koeficienty (inkrementace, dekrementace a operace Int() působí pouze na koeficient dyadic). Při operaci násobení je třeba roznásobit každý koeficient s každým koeficientem druhého čísla a sečtení odpovídajících produktů, což je implementováno pomocí dvourozměrného pole produktů. Při roznásobení koeficientů mohou

vzniknout čísla, se kterými třída SurrealValue neumí pracovat, (např. ω²√ω). V takovém případě je vystavena výjimka *Unsupported operation*.

Tabulka 2: *Produkty roznásobení jednotlivých koeficientů (zdroj vlastní)*

	ω²	ω	√ω	1	√ε	ε	ε²
ω²	ω^4	ω³	ω²√ω	ω²	ω√ω	ω	1
ω	ω³	ω²	ω√ω	ω	√ω	1	ε
√ω	ω²√ω	ω√ω	ω	√ω	1	√ε	ε√ε
1	ω²	ω	√ω	1	√ε	ε	ε²
√ε	ω√ω	√ω	1	√ε	ε	ε√ε	ε²√ε
ε	ω	1	√ε	ε	ε√ε	ε²	ε³
ε²	1	ε	ε√ε	ε²	ε²√ε	ε³	ε^4

Kromě již zmíněných operací umožňuje třída SurrealValue ještě práci s koeficienty: vrácení hodnoty ryze dyadické části $D(X) = X{:}\,X_{not(d)} = 0$, transfinitní části $O(X) = X{:}\,X_{not(\omega,\sqrt{\omega},\omega^2)} = 0$, infinitesimální části $E(X) = X{:}\,X_{not(\epsilon,\sqrt{\epsilon},\epsilon^2)} = 0$, netransfinitní části $!\,O(X) = X{:}\,X_{(\omega,\sqrt{\omega},\omega^2)} = 0$ a neinfinitesimální části $!\,E(X) = X{:}\,X_{(\epsilon,\sqrt{\epsilon},\epsilon^2)} = 0$. Dále také snížení transfinitní části ($a\omega^2 + b\omega \rightarrow a\omega + b$), povýšení na transfinitní část ($a\omega + b \rightarrow a\omega^2 + b\omega + 0$), povýšení infinitesimální části ($a\epsilon + b\epsilon^2 \rightarrow a + b\epsilon$) a snížení na infinitesimální část ($a + b\epsilon \rightarrow 0 + a\epsilon + b\epsilon^2$). Třída také obsahuje statické konstanty *SurrealValue.zero*, *SurrealValue.omega*, *SurrealValue.omegaSquared*, *SurrealValue.epsilon, SurrealValue.epsilonRoot* a *SurrealValue.Squared* pro hodnoty odpovídající 0, ω, √ω, ω², ε, √ε, ε² v tomto pořadí

Specifika implementace třídy SurrealNumber

Nadreálné číslo je implementováno dvojicí množin L (=*left*, tj. levá množina) a R (=*right*, tj. levá množina) typu SurrealSet a hodnotou daného nadreálného čísla typu SurrealValue, která je vypočítána ihned při konstrukci čísla.

Tato implementace umí reprezentovat pouze nadreálná čísla složená z některých typů množin:

- prázdná množina (reprezentována hodnotou atributu null)

- konečná množina nadreálných čísel ve tvaru bω + d + fϵ.
- nekonečné množiny nadreálných čísel ve tvaru bω + d + fϵ, kde právě jeden z koeficientů b, d či f roste nebo klesá do ±∞ (reprezentováno hodnotou ±Infinity).
- nekonečná množina nadreálných čísel ve tvaru bω + d, kde koeficient d se blíží konkrétní konečné limitě, konkrétnímu konečnému číslu (reprezentováno neostrou množinou, viz dále).
- nekonečná množina nadreálných čísel ve tvaru bω + d + fϵ, kde koeficient f roste či klesá k 0 (reprezentováno neostrou množinou s extrémem hodnoty bω + d ± 1ε²).
- nekonečná množina nadreálných čísel ve tvaru bω + d, kde koeficient b roste či klesá k 0 (reprezentováno neostrou množinou s extrémem hodnoty 0ω ± ∞).

Nekonečné množiny (kromě posledního zmíněného případu, kde je to naopak) musí stoupat či klesat z příslušných směrů, tj. množina L stoupá nebo se blíží zleva, množina P klesá nebo se blíží zprava. Jiné možnosti nejsou smysluplné.

Algoritmus výpočtu hodnoty čísla je stěžejním bodem celé datové části a bude popsán samostatně v další části. Operace jsou implementovány jako operace s dvojicemi množin podle definicí nadreálných čísel a následné konstrukce nového čísla z výsledných množin. Kvůli způsobu reprezentace nekonečných množin dokáže operace násobení pracovat pouze s takovými nadreálnými čísly, prázdných nebo konečných množin; pokud zadané množiny tuto podmínku poruší, je vystavena výjimka *Unsupported operation.* Třída také obsahuje statickou konstantu *SurrealNumber.zero*, což je nadreálné číslo s hodnotou 0.

Specifika implementace třídy SurrealSet a odvozených tříd

Množina nadreálných čísel je reprezentována dvojicí extrémních hodnot *min* (=minimální prvek) a *max* (maximální prvek) typu SurrealValue a dvojicí korespondujících atributů typu Boolean, které určují, zda se množina z dané strany uzavřená či otevřená (např. množina $\{1; \frac{1}{2}; \frac{1}{4}; \frac{1}{8}; \cdots\}$ má sice minimální hodnotu min = 0, ale zároveň je nastavená jako otevřená zleva). Minimální prvky se k daným hodnotám mohou blížit pouze zprava, jinak by nebyl minimální; stejně tak maximální prvky se k daným hodnotám mohou blížit pouze zleva. Třída také podporuje reprezentaci nekonečných množin nadreálných čísel rostoucích do nekonečna (případně klesajících do -∞) tak, že umožňuje extrémům nabývat nekonečných hodnot. Nemůžou však oba extrémy ani oba růst do nekonečna, ani oba klesat do -∞.

Množiny odvozené od této třídy v podstatě jen přepisují metody vypsání textové reprezentace, plus uchovávají ty informace, které jsou k tomu třeba (např. první tři prvky nekonečné množiny).

2.3.3 Algoritmus nalezení hodnoty nadreálného čísla

Budeme používat následující notaci: X pro hledanou výslednou hodnotu, L pro největší prvek levé množiny a R pro nejmenší prvek pravé množiny.

Tento algoritmus implementovaný ve třídě SurrealNumber umí pracovat pouze s čísly b$\omega + d + f\epsilon$. Nejprve zkontroluje, zda zadané množiny splňují požadovaná kritéria. Pokud ne, vystaví výjimku *Invalid argument(s)*. Následně nastane jedna ze čtyř možností:

Obě množiny jsou prázdné

- Jedná se o triviální případ, výsledkem je hodnota X = 0.

Právě jedna z množin je prázdná

Popíšeme si případ, kdy je pravá množina prázdná. Případ prázdné levé množiny se řešen symetricky.

- Pokud L < 0, je výsledkem hodnota X = 0.

Jinak je výsledná hodnota dána obecně vztahem $X = I\mathrm{nt}\left(L_{(\omega^{n_{max}})} + 1\right) + L_{\omega^m} + 0$. To znamená: pokud aspoň jeden z neinfinitesimálních koeficientů L roste do nekonečna, je $L_{(\omega^{n_{max}})}$ takovým koeficientem s nejvyšším řádem, jinak je $L_{(\omega^{0=d})}$. L_{ω^m} jsou všechny koeficienty vyššího řádu než $L_{(\omega^{n_{max}})}$ a ty zůstávají stejné. Všechny ostatní koeficienty (tj. koeficienty nižšího řádu než $L_{(\omega^{n_{max}})}$) jsou nulové.

Tato část algoritmu je implementována následujícím způsobem:

- Pokud !E(L) neroste do nekonečna, je výsledkem následující celé číslo větší než L, tedy hodnota X=$O(L) + Int\left(L_{(d)} + 1\right) + 0$.
- Pokud $O(L) \to \infty$, je zavolána rekurzivně tento bod algoritmu na L se sníženou transfinitní částí. Takto získané X' je zpátky povýšeno na transfinitní část, čímž získáme výslednou hodnotu.
- Zbývá případ, kdy roste do nekonečna právě $L_{(d)}$, a výsledkem je hodnota X=O(L): $X_{(\omega)} = Int\left(L_{(\omega)} + 1\right)$.

Žádná z množin není prázdná

- Pokud L < 0 a zároveň 0 < R, je výsledkem hodnota 0.

Algoritmus se zde opět rozděluje na dvě části:

Žádná z množin není prázdná a O(L) < O(R)

- Mějme L' = L se sníženou transfinitní částí a R' = R se sníženou transfinitní částí.
- Pokud $L_{(d)}$ neroste do nekonečna, může být O(X) = O(L), proto $L'' = Num(L', -1)$, jinak $L'' = L'$.
- Podobně pokud $R_{(d)}$ neklesá do -∞, může být O(X) = O(R), proto $R'' = Num(R', +1)$, jinak $R'' = R'$.
- Nyní najdeme $O = \{L'' \mid R''\}$.
- Pokud $O = L'$, znamená to, že výsledná hodnota $X = Int(L + 1)$.
- Podobně pokud $O = R'$, znamená to, že výsledná hodnota $X = Int(R - 1)$
- Jinak dostaneme výslednou hodnotu X povýšením O na transfinitní část.

Žádná z množin není prázdná a O(L) = O(R)

V tomto případě bude O(X) = O(L) = O(R). Obzvlášť v této větvi záleží na konkrétní implementaci způsobu reprezentace některých specifických množin; např. množina $\{-\omega; -\frac{1}{2}\omega; -\frac{1}{4}\omega; \cdots\}$ je v této implementaci reprezentována s O(L) = 0.

- Pokud $O(X) \neq 0$, můžeme separátně vyřešit !O(X) rekurzivním hledáním hodnoty $X' = \{!\,O(L) | !\,O(R)\}$. Výsledná hodnota pak bude X = O(X) + X'.

Nyní je jisté, že O(X) = 0. To znamená, že !E(X) = D(X). Následně implementace řeší konkrétní speciální případy.

- Pokud $L_{(d)} \to \infty \wedge R_{(\omega)} \to 0^+$ a R je množinou transfinitních hodnot, jejichž koeficient omega se blíží 0 (zprava), je výsledná hodnota $X = \sqrt{\omega}$. Případ pokud $R_{(d)} \to -\infty \wedge L_{(\omega)} \to 0^-$ je řešen symetricky.

Poté řešíme speciálně čísla s $L_{(\epsilon)} \to 0^- \vee R_{(\epsilon)} \to 0^+$ (připomeňme, že tento případ je v této implementaci reprezentován hodnotami $K_{(\epsilon^2)} = \pm 1$)

- Ve speciálním případě pokud $L_{(\epsilon)} \to 0^- \wedge R_{(\epsilon)} \to 0^+ \wedge D(L) = D(R)$, pak se L zleva a R zprava blíží ke stejnému číslu, které je i hledanou výslednou hodnotou X = D(L) = D(R).
- Pokud $L_{(\epsilon)} \to 0^- \wedge$ D(L) = R, tj. $L \to R^+$, výsledná hodnota $X = !\,E(L) - \epsilon^2$. Symetricky řešíme pokud $R_{(\epsilon)} \to 0^+ \wedge$ D(R) = L, tj. $R \to L^-$.
- Jinak můžeme zjednodušit:
    - L': $L' \to D(L)$ pokud $L_{(\epsilon)} \to 0^-$, jinak $L' = L$
    - R': $R' \to D(R)$ pokud $R_{(\epsilon)} \to 0^+$, jinak $R' = R$
    - Výslednou hodnotu najdeme rekurzivně jako $X = \{L' \mid R'\}$.

Poté se řeší čísla s $E(L) \neq 0 \vee E(R) \neq 0$:

- Pokud $E(L) \neq E(R)$ můžeme zjednodušit:
    - L': $L' \to D(L)$ pokud $L_{(\epsilon)} < 0 \vee L \to 0^-$, jinak $L' = D(L)$ (což je to samé jako $L' = L$ pokud $L_{(\epsilon)} = 0$, $L' = D(L)$ pokud $L_{(\epsilon)} > 0$, jinak L': $L' \to D(L)$)
    - R': $R' \to D(R)$ pokud $R_{(\epsilon)} < 0 \vee$ R $\to 0^+$, jinak $R' = !\,E(R)$ (což je to samé jako $R' = R$ pokud $R_{(\epsilon)} = 0$, $R' = D(R)$ pokud $R_{(\epsilon)} > 0$, jinak R': $R' \to D(R)$)
    - Výslednou hodnotu najdeme rekurzivně jako $X = \{L' \mid R'\}$.
- Pokud $L \to D(L) = D(R) \wedge R_{(\epsilon)} \to -\infty$, výslednou hodnotou je $X = D(L) - \sqrt{\epsilon}$. Symetricky řešíme pokud $R \to D(L) = D(R) \wedge L_{(\epsilon)} \to \infty$.
- Jinak zavedeme L' jako povýšení infinitesimální hodnoty L a R' jako povýšení infinitesimální hodnoty R.
    - Pokud $L \to n^+$:
        - Pokud $R_{(\epsilon)} < 0$, tak $L' = \emptyset$. Jinak je výsledná hodnota X = D(L).
    - Symetricky řešíme pokud $R \to n^-$.
    - Rekurzivně vypočítáme novou hodnotu $E' = \{L' | R'\}$ a ponížíme ji na infinitesimální část.

- Hledaná výsledná hodnota $X = D(L) + E'$.

Předcházející algoritmus už vyřešil všechny případy s transfinitními i infinitesimálními hodnotami. Můžeme se tedy spoléhat, že O(x) = E(X) = 0 a !O(X) = !E(X) = D(X). Dále řešíme čísla $L \to n^+ \vee R \to n^-$:

- Pokud $D(L) \neq D(X)$: $L' = Num(L, -1)$ pokud $L \to n^+$, jinak $L' = L$; symetricky vypočítáme i R'. Výslednou hledanou hodnotu pak vypočítáme rekurzivně jako $X = \{L' | R'\}$.
- Jinak pokud D(L) = D(R): pokud $L \to n^+ \wedge R \to n^-$, výsledná hodnota X = D(L); pokud jen pokud $L \to n^+$, výsledná hodnota $X = D(L) - \epsilon$; pokud jen $R \to n^-$, výsledná hodnota $X = D(L) + \epsilon$.
    - Tato operace je implementována následovně: vezmeme mezivýsledek $X' = D(L)$. Pokud $L \to n^+$, přičteme ϵ. Potom pokud $R \to n^-$, odečteme ϵ. Nakonec $X = X'$.

Algoritmus konečně dospěl do fáze, ve které jsou L i R jednoduché dyadické zlomky. Nejdříve si představíme obecný algoritmus nelezení hodnoty mezi dvěma dyadickými zlomky a pak si ukážeme některé optimalizace.

Algoritmus nalezení hodnoty mezi dvěma dyadickými zlomky

Obecný algoritmus využívá následující vlastnosti nadreálných čísel: pokud je nadreálné číslo $a = \{a_L | a_R\}$ v kanonickém tvaru, pak pro všechny jeho potomky b platí: $\forall b: a_L < b < a_R$. Pokud $a < b$, platí $a_L < a < b < a_R$; pokud naopak $b < a$, platí $a_L < b < a < a_R$. Jak vidíme, b spadá právě do jednoho konkrétního intervalu určeného podle porovnání s a. Zároveň pro libovolnou trojici nadreálných čísel c, d, e platí: pokud $c < d < e$, pak $f = \{c \mid d\}$ je buď samotné číslo d, nebo jeden z předků d. Tento vztah platí i pro $a < b < a_R$ a $a_L < b < a$.

Definujme proto následující operaci: mějme hledané neznámé číslo v kanonickém tvaru $x = \{x_L | x_R\}$ a jeho libovolného předka $p = \{p_L | p_R\}$ taktéž v kanonickém tvaru. Pokud $p > x_R$, pak $q = \{p_L | p, p_R\} = \{p_L | p\}$ je také předek x; pokud $p < x_L$, pak $q = \{p_L, p | p_R\} = \{p | p_R\}$ je také předek x. Přičemž je vždy pravda $\delta(p) < \delta(q)$ (tzn. q mladší, je to tedy bližší předek x). Pokud by $\delta(p) > \delta(q)$, pak by se hodnota $\{p_L | p_R\} = q$, což je spor s původní definicí

$\{p_L|p_R\}$ jako kanonického tvaru p. Nemůže platit ani $p = q$, protože pro $q = \{p|p_R\}$ nebo $\{p_L|p\}$ musí platit nerovnost $p < q < p_R$ nebo $p_L < q < p$, jinak by nešlo o nadreálné číslo.

Když tuto operaci budeme provádět opakovaně, každé další q je mladší a mladší, blíže a blíže generaci svého potomka q, a dříve či později dosáhneme generace $\delta(q) = \delta(x)$, kde $x_L \not\geq q \wedge q \not\leq x_R$, což lze zapsat jako $x_L < q < x_R$, a tedy $x = q$. Vzhledem k iterativní povaze tohoto algoritmu ho ovšem nelze prakticky použít pro nadreálná čísla x a p, jež dělí nekonečný počet předků, protože k dosažení výsledku bychom potřebovali nekonečně mnoho kroků.

Při iniciaci algoritmu potřebujeme zvolit vhodné výchozí číslo, vhodný výchozí bod na stromu nadreálných čísel, protože dané číslo musí být předkem x, jinak algoritmus nedosáhne výsledku (ba ani neskončí). Nejjistějším výchozím číslem je nejjednodušší nadreálné číslo 0, protože všechna ostatní nadreálná čísla jsou jeho potomky.

0. Mějme na vstupu dvojici x_L, x_R (číslo nebo prázdná množina).
1. Iniciace iterační proměnné i_0 = 0 (nadreálné číslo $\{i_L|i_R\}$.
2. Začátek cyklu.
3. Pokud $i_n \leq x_L$: $i_{n+1} = \{i_n|i_{Rn}\}$.
4. Jinak pokud $i_n \geq x_R$: $i_{n+1} = \{i_{Ln}|i_n\}$.
5. Jinak jdi na konec cyklu.
6. Jdi na začátek cyklu.
7. Konec cyklu.
8. Výsledné $x = \{x_L|x_R\} = i$.

Když za výchozí číslo zvolíme 0, musíme projít všech $\delta(x)$ generací. Pokud bychom zvolili vhodnější výchozí číslo

Optimalizace algoritmu celočíselnými pásy

Celočíselným pásem rozumějme všechna všechna čísla mezi dvěma po sobě jdoucími celými čísly, neboli celočíselný pás jsou $\forall x: i \leq x \leq i + 1; i \in \mathbb{Z}$. Pro hodnotu $a = \{a_L|a_R\}$ platí: pokud a_L a a_R nejsou ze stejného celočíselného pásu, a bude celé číslo mezi nimi (jinak

protože se v tomto algoritmu stále pohybujeme v oboru konečných dyadických zlomů, bude a také dyadický zlomek).

- Vypočítejme i_L jako nejmenší celé číslo větší než x_L hledaného čísla a i_L jako největší celé číslo menší než x_R hledaného čísla. K tomuto výpočtu můžeme použít operace inkrementace/dekrementace a Int().
- Pokud $i_L = i_R$, je to zároveň hledaná hodnota $x = i_L = i_R$.
- Pokud $i_L < i_R$, bude hledaná hodnota jedno z čísel i:
    - Pokud $0 < i_L < i_R \Rightarrow x = i_L$.
    - Pokud $i_L < i_R < 0 \Rightarrow x = i_R$.
    - Možnost $i_L < 0 < i_R \Rightarrow x = 0$ jsme vyřešili už na začátku původního algoritmu nalezení hodnoty nadreálného čísla
- Pokud $i_L > i_R$, znamená to, že x_L a x_R leží ve stejném celočíselném pásu. V tento moment můžeme začít hledat hodnotu x iterativní metodou, přičemž výchozí hodnotu i_0 můžeme nastavit na $\{i_R|i_L\}$, protože to je garantovaný společný předek všech ostatních dyadických čísel daného celočíselného pásu.

Optimalizace algoritmu krajními hodnotami

Ještě před iterativní metodu s optimalizací celočíselnými pásmy zařadíme výpočet krajních hodnot $k_L = Num(x_L, +1)$ a $k_R = Num(x_R, -1)$. Tyto hodnoty nám umožňují rychle zkontrolovat některé možnosti:

- Pokud $k_L = x_R \vee k_R = x_L$, znamená to, že mezi danými čísly x_L a x_R neleží žádná starší čísla, a hledaná hodnota proto bude $x = \frac{x_L + x_R}{2}$.
- Pokud $k_L = k_R$, hledaná hodnota bude právě $x = k_L = k_R$
- Pokud $k_L > x_R$, pak x_R je potomek x_L a výchozí hodnotu iterační metody nastavíme na $i_0 = \{x_L|k_L\}$. Symetricky pokud $k_R < x_L$, pak x_L je potomek x_R a výchozí hodnotu iterační metody nastavíme na $i_0 = \{k_R|x_R\}$.
- Teprve pokud se nenaplní ani jedna z těchto podmínek přejde se na výpočet celočíselných pásů.

2.3.4 Testování datových struktur

Datové struktury byly testovány dvěma způsoby. Zaprvé osobním vybráním několika náhodných testovacích případů, na kterých se daná funkcionalita ručně ozkoušela. Testovací případy byly vybrány tak, aby jednak postihly všechny možné větve algoritmů (zvlášť algoritmu nalezení hodnoty nadreálného čísla), jednak aby měly širokou škálu různorodosti.

Po tomto testování probíhalo ještě fuzz testování. Testovací program opakovaně náhodně vybíral z pole několika desítek ručně zkonstruovaných nadreálných čísel a zkoušel s nimi provádět testované operace. Přitom byla sledována správnost výsledků a absence chybových hlášek.

Tyto dvě fáze se za sebou při odstraňování chyb mohly i několikrát zopakovat.

2.3.5 Možnosti rozšíření datových struktur

Možnosti rozšíření datových struktur navazují na překážky objevené při návrhu a implementace. V této bakalářské práci k demonstraci práce s nadreálnými čísly stačí částečné řešení, které ovšem omezuje možnosti aplikace.

Prvním možným směrem rozšiřování je rozšíření implementace datových struktur nejen na dyadické koeficienty, ale na reálná čísla. Minimálně pro libovolná racionální čísla by to mělo být možné, pokud by se třída Dyadic nahradila podobnou třídou Rational (která by fungovala velice podobně jako Dyadic s tím, že by jmenovatelem reprezentovaného zlomku mohla být jakákoli nezáporná celá čísla) a pokud by se patřičně ošetřil algoritmus nalezení hodnoty nadreálného čísla.

Druhým možným směrem je rozšíření implementace datových struktur na jakékoli mocniny ω. To by mělo být proveditelné změnou implementace třídy SurrealValue z prostého napevno stanoveného výčtu koeficientů buď na rekurzivní strukturu typu $\omega^0\{a + \omega^1[b + \omega^2(c + \cdots)]$, nebo na dynamický seřazený list koeficientů.

Zatřetí, bylo by možné lépe vyřešit otázku interpretace nekonečných množin, jdoucích do nekonečna či blížících se konkrétní limitě, což by mohlo dovolit naimplementovat operaci násobení pro všechna nadreálná čísla. Případně by bylo možné implementovat další typy množin dědící třídu SurrealSet, např. MultipleValuesSurrealSet, která by dokázala věrneji reprezentovat a zobrazovat konečnou množinu daných nadreálných čísel.

Začtvrté, datové struktury v této práci implementují pouze základní aritmetické operace sčítání, odčítání a násobení, ale nadreálná čísla lze také dělit, umocňovat a odmocňovat,

logaritmovat,… Tyto operace nebyly implementovány nejen proto, že jsou obtížné, ale hlavně protože po datových strukturách vyžadují schopnost reprezentovat reálná čísla. Pokud by se podařila vyřešit alespoň reprezentace racionálních čísel, umožnilo by to implementaci alespoň operace dělení.

V neposlední řadě se může stát centrem zájmu, zkoumání a snahy o zefektivnění algoritmus nalézání hodnoty nadreálného čísla. Některé optimalizace by se mohly skrývat za matematickými operacemi s narozeninovými čísly $\delta(X)$.

Konečně datové struktury byly navrženy tak, aby je mohli využívat i další vývojáři. Tento záměr může být dokončen vytvořením knihovny JS tříd a jejím publikováním na vhodné platformě, jakou je například GitHub, GitLab či správce JS balíčků npm.

2.4 Uživatelské rozhraní

2.4.1 Návrh uživatelského rozhraní

Při návrhu rozhraní pro běžné uživatele byly zvažovány tři druhy aplikace: desktopová aplikace, mobilní aplikace a webová aplikace. V konečném rozhodnutí byly zohledněny následující výhody webových aplikací:

- **kompatibilita mezi platformami:** webová aplikace je nezávislá na OS, zařízení, atd. Není tedy potřeba řešit separátní řešení pro různé platformy ani to, že by pro určitou platformu aplikace nebyla dostupná.
- **přístupnost:** k přístupu k webové aplikaci stačí internetové připojení a prohlížeč, nejsou potřeba žádné další nástroje
- **připravenost:** webová aplikace nevyžaduje žádnou instalaci ani nestavování, uživatel ji může začít používat jednoduše do pár sekund po otevření
- **snadnější aktualizace**, doplnění obsahu, oprava chyb,…
- **dohledatelnost:** k přístupu k webové aplikaci stačí mít odkaz, který lze snadno poslat někomu jinému, sdílet na internetu či sociálních sítích, případně si může uživatel aplikaci vyhledat v internetovém vyhledávání

Vzhledem k jednoduchosti výsledné aplikace se nevýhody webových aplikací výrazně neprojevují. Hlavní překážkou tak zůstává nutnost přístupu k internetu, což lze ale poměrně lehce eliminovat stažením souborů aplikace.

Cílový uživatel

Uživatel musí mít dostatečné matematické vědomosti, aby mohl porozumět probírané látce. Určité minimální požadavky formuluje např. Tøndering (2019, s. 4-5). Zároveň by ovšem pro matematicky vysoce vzdělaného uživatele bylo pravděpodobně efektivnější i podnětnější studovat téma nadreálných čísel četbou odborné literatury. Cílový uživatel je proto měl být matematicky vzdělaný dost, ale ne příliš. Za cílového uživatele proto byl zvolen student střední školy či prvních ročníků vysoké školy, který má alespoň minimální zájem o matematiku. Tato volba byla zohledněna i při tvorbě obsahu pro uživatelské rozhraní.

Demonstrace práce s nadreálnými čísly neznamená jen, aby uživatel tuto práci viděl. Měl by ji také pochopit a měl by mít možnost sám si práci s nadreálnými čísly poučeně vyzkoušet. Vzhledem k relativní nerozšířenosti povědomí o tak specifickém matematickém oboru v běžné populaci je proto třeba zároveň s (nebo ještě lépe před) demonstrací práce s nadreálnými čísly uživateli přiblížit a vyložit probíranou látku. Praktická ukázka práce s nadreálnými čísly se pak může stát přirozenou součástí výkladu.

Kritéria textového obsahu

Hlavním cílem textů pro uživatelské rozhraní je naučit uživatele, co to jsou nadreálná čísla, jaké mají vlastnosti a jak se s nimi pracuje. Z tohoto cíle vyplývají hlavní požadavky na textový obsah, totiž přehlednost a srozumitelnost. Tato kritéria jsou upřednostňována před stručností i rigorózností matematických zápisů, termínů, definicí a výroků.

2.4.2 Implementace uživatelského rozhraní

Webová aplikace byla implementována jako jednoduchá jednostránková aplikace. Aplikace pracuje čistě na uživatelově straně, takže kromě prvotního získání potřebných souborů není potřeba žádná komunikace se serverem.

Obsah aplikace je rozdělena do horizontální osy na jednotlivé kapitoly. Každá kapitola zabírá celou šířku obrazovky a přechod mezi nimi je možný buď horizontálním scrollováním, nebo kliknutím na šipku na pravém či levém okraji obrazovky. Každá kapitola má vlastní id, takže je pak možné sdílet odkaz přímo na vybranou sekci.

Kvantitativně tvoří obsah jednotlivých kapitol především odstavce textů občas doplněné o bodové seznamy či ilustrační obrázky. Kromě nich se ale v aplikaci nachází také dva druhy widgetů.

Widget TextBubble

Jedná se o jednoduchou zavíratelnou vyskakovací textovou bublinu, která uživateli zobrazuje zprávy, konkrétně hlásí chyby při práci s funkčními prvky aplikace. Textová bublina má tři prosté operace:

- funkce *textBubbleUp()* zobrazí textovou bublinu
- funkce *textBubbleDown()* textovou bublinu skryje
- funkce *showTextBubble(text, duration = 5000)* zobrazí textovou bublinu s daným textem a po zadané době (v ms, výchozí hodnota 5000) ji opět skryje

Výhodou tohoto widgetu je snadné ovládání, znovupoužitelnost v jakémkoli dalším projektu a relativně přímočará možnost uzpůsobení vlastním potřebám.

Widget DragNDrop

Tento widget, který se v aplikaci objevuje celkem patnáctkrát, funguje jako miniaturní kalkulačka. Skládá se ze tří vertikálních vrstev: Vzorce, Možností a Dalších možností:

- Vzorec představuje (abstraktně i graficky) operaci s nadreálnými čísly, např. konstrukci nadreálného čísla, sčítání, odčítání či násobení. Vlevo ve Vzorci jsou volná místa, kam mohou být umisťovány vstupní hodnoty. Vpravo ve Vzorci je tlačítko, kde se zobrazuje výsledek operace. Případně může Vzorec obsahovat ještě výběr operace.
- Možnosti je seznam hodnot, které uživatel může zadávat do vzorce.
- Další možnosti je soubor číselníků a tlačítek, který uživateli umožňuje přidání libovolné hodnoty mezi Možnosti. Uživatel nejprve na číselníku nastaví vybranou hodnotu, kterou pak mezi Možnosti přidá klepnutím na odpovídající tlačítko.

Zadávání hodnot do vzorce funguje pomocí „drag and drop“, neboli přetáhni a pusť. Uživatel kursorem myši (případně dotykem na dotykových displejích) „uchopí“ zvolenou hodnotu v Možnostech, přetáhne ji na příslušné místo ve Vzorci a pustí, čímž ji zadá do vzorce. Pokud na daném místě už nějaká hodnota byla, je odstraněna.

Když se zaplní všechna vstupní pole ve Vzorci, automaticky se vypočítá výsledná hodnota, která je pak uživatelovi zobrazena. Pokud při výpočtu dojde k nějaké chybě, je tato skutečnost uživatelovi oznámena vyskakovacím okénkem TextBubble.

Uživatel může výslednou hodnotu přidat do Možností stisknutím tlačítka, na kterém se zobrazuje (je-li to v daném případě povoleno). Pokud naopak chce odstranit nějakou hodnotu ze Vzorce či Možností, stačí na ni dvakrát poklepat.

Tento widget je do aplikace přidáván dynamicky, tj. na daném místě zdrojového kódu stačí zavolat patřičná funkce. Vytváření widgetu je flexibilní, lze zvolit typ hodnot, s nimiž se bude pracovat, filtr hodnot, které nelze přidávat do Možností, lze nastavit konkrétní podobu Dalších možností, do Možností lze přidávat výchozí hodnoty (které nelze odstranit poklepáním)…

Texty aplikace

Textový obsah byl vytvářen zároveň s vlastním studiem problematiky nadreálných čísel a přípravou teoretické části této práce. Na konec aplikace byly přidány tři hlavní zdroje této bakalářské práce jako nabídka literatury k podrobnějšímu studiu. Jazyk textu byl přizpůsoben předpokládanému cílovému uživateli i za cenu toho, že bude ze striktního matematického hlediska nepřesný či neúplný. Celkové množství obsahu (kapitol, textu) je poměrně veliké, dané informace by jistě bylo možné prezentovat stručněji a s vyšší hustotou. Přednost ovšem dostala srozumitelnost: raději aby uživatel úžeji zaměřil svou pozornost na menší počet konceptů naráz, zato více popsaných.

Matematické vzorce a symboly jsou zapisovány ve formátu LaTeX a zobrazovány díky knihovně MathJax. Zobrazení těchto prvků je opravdu kvalitní a pomáhá s přehledností a srozumitelností obsahu. Bohužel má i své nedostatky, které budou probrány v následujícím pododdílu.

2.4.3 Testování uživatelského rozhraní

Uživatelské rozhraní bylo testováno ručně: jednak samotné rozvržení stránky, posouvání kapitol atd., jednak bylo samostatné testování věnováno jednotlivým widgetům.

Výkon aplikace byl testován také online nástrojem PageSpeed Insights, který aplikaci ohodnotil skórem 52 za výkon, 90 za přístupnost, 100 za dodržování doporučených postupů a 100 za SEO. Některá doporučení tohoto nástroje byla implementována, ovšem výrazně nízké skóre za výkon je způsobeno překreslováním prvků a blokováním hlavního vlákna po prvním načtení webové aplikace. Toho je důsledek knihovna MathJax, která po prvním načtení všechny označkované prvky v aplikaci přepisuje na matematické symboly. Vzhledem k opravdu vysokému počtu matematických zápisů může zpomalení během prvních pár sekund

používání aplikace znatelné. Tyto matematické zápisy jsou však vzhledem k typu a rozsahu obsahu v aplikaci nutné.

2.4.4 Možnosti rozšíření uživatelského rozhraní

Možnosti rozšíření obsahu

Téma nadreálných čísel je široké, obsah aplikace je proto možné doplnit o další kapitoly, např. o mocninách ω, o exponenciaci a logaritmování nadreálných čísel, o derivaci nadreálných čísel, o alternativních způsobech zápisu, o příbuzných číselných systémech (např. o vztahu s hyperreálnými čísly, popsaném Philipem Ehrlichem) či o využití v dalších vědních oborech, jako je teorie her či teoretická fyzika. Obsah uživatelského rozhraní je také omezen tím, co mu umožňuje implementace datových struktur: např. pokud by bylo implementováno dělení čísel, mohlo by být prezentováno i v uživatelském rozhraní.

Webovou aplikaci by také bylo možné obohatit dalšími druhy interaktivních widgetů. Uživatel by například mohl ocenit widget, který by vykresloval nadreálná čísla do stromové struktury (třeba při konstrukci nových čísel, případně by mohl ukazovat postup při hledání hodnoty daného nadreálného čísla).

Zlepšení uměleckého designu

Při implementaci aplikace byl zvolen jednoduchý minimalistický vizuální návrh, který je přehledný, nerozptyluje, připomíná čtečku. Zároveň však může být vnímán jako jednotvárný, nudný a nezajímavý. Přiměřenou úpravou webdesignu by se aplikace mohla stát už na první pohled uživatelsky atraktivnější. Zásahy se přitom nemusí omezovat jen na vizuální stránku, ale třeba i na zvukovou.

Responzivita a přístupnost aplikace

Aplikace je responzivní, ale byla implementována přednostně pro desktopová zařízení, takže její zobrazování a ovládání na mobilních zařízeních stále ještě není ideální. Obzvlášť dlouhé matematické vzorce vytvořené knihovnou MathJax mohou na mobilních zařízeních přetékat mimo obrazovku.

Aplikace není vůči handicapovaným uživatelům ani vyloženě nepřátelská, ani nijak zvlášť otevřená. V tomto směru tedy existuje ještě velký prostor k růstu, např. k vývoji funkčních prvků použitelných i nevidomými.

Vývoj výuková aplikace

Na základě webové aplikace, která práci s nadreálnými čísly pouze demonstruje, by bylo možné vyvinout aplikaci, která práci s nadreálnými čísly aktivně vyučuje, podobně jako existují aplikace vyučující cizí jazyky. K tomu by bylo potřeba lépe rozdělit obsah do lekcí, implementovat widgety umožňující testování uživatelových znalostí a systém správy uživatelského účtu, průběhu jeho studia a studijních výsledků.

ZÁVĚR

V teoretické části této práce byla popsaná nadreálná čísla, jejich definice, vlastnosti a struktury, které vytváří a popsán postup při konstrukci nadreálných čísel. Dále byla probírána problematika reálných, transfinitních a infinitesimálních čísel. Poté byly představeny základní operace porovnání, sčítání, čísla opačného, odčítání a násobení reálných čísel s dodatkem o vztahu transfinitních a infinitesimálních čísel a zmínkou o nadkomplexních číslech. Nakonec bylo nastíněno, jak lze využít nadreálná čísla v teorii her pro reprezentaci hry mezi dvěma hráči a jejího předpokládaného výsledku.

V experimentální části se práce na aplikaci rozdělila do dvou vrstev: práce na datových strukturách a práce na uživatelském prostředí. Datové struktury byly navrženy tak, aby reprezentovaly základní objekty, s nimiž se při práci s nadreálnými čísly setkáváme. Byly objeveny některé problémy reprezentace nadreálných číslech na konečných výpočetních strojích, zejména jak věrně reprezentovat nedyadické reálné hodnoty a jak nekonečné hodnoty.

Při návrhu uživatelského prostředí byla především zohledňována přehlednost a srozumitelnost. Samotné uživatelské prostředí je jednoduchou jednostránkovou aplikací na straně klienta.

POUŽITÁ LITERATURA

1. SCHLEICHER, Dierk. Interview with John Horton Conway. *Not. AMS*, 2013, 567-575. Dostupné z: https://scholar.archive.org/work/hbcrw2oibndhxci4aep2rugoau/access/wayback/http://www.ams.org/journals/notices/201305/rnoti-p567.pdf [cit. 2024-05-01]

2. GONSHOR, Harry. *An Introduction to the Theory of Surreal Numbers*. Cambridge University Press, 1986. ISBN 9780511629143. Dostupné z: https://doi.org/10.1017/CBO9780511629143. [cit. 2024-05-01].

3. KNUTH, Donald Ervin. *Surreal Numbers: How Two Ex-students Turned on to Pure Mathematics and Found Total Happiness*. 12. Addison-Wesley Publishing Company, 1974. ISBN 9780201038125.

4. TØNDERING, Claus. *Surreal Numbers – An Introduction.* Online. 2019. Dostupné z: https://www.tondering.dk/download/sur.pdf. [cit. 2024-04-29].

5. LÁNSKÝ, Lukáš. *A visualization of the surreal number tree*. Online. In: Wikimedia Commons. 2012. Dostupné z: https://commons.wikimedia.org/wiki/File:Surreal_number_tree.svg. [cit. 2024-04-26]. Úpravy vlastní.

6. EHRLICH, Philip. The Absolute Arithmetic Continuum and the Unification Of all Numbers Great and Small. Online. *The Bulletin of Symbolic Logic*. 2012, roč. 18, č. 1, s. 1-45. ISSN 1079-8986. Dostupné z: https://doi.org/10.2178/bsl/1327328438. [cit. 2024-05-01].

7. BAJNOK, Béla. *An Invitation to Abstract Mathematics*. Springer, 2013. ISBN 9781461466369.

8. CONWAY, John Horton. *On Numbers and Games*. Academic Press, 1976. ISBN 0-12-186350-6.

9. ALLING, Norman L. *Foundations of Analysis over Surreal Number Fields*. North-Holland, 1987. ISBN 0-444-70226-1.

10. DOKOS, Theodore. *Surreal Numbers and Games*. 2009. Dostupné z: https://math.osu.edu/sites/math.osu.edu/files/SurrealNumbers.pdf [cit. 2024-04-30]

11. YOUNG, Sean. *The Best Coding Languages for Web Development in 2024*. Online. Code institute. 2024. Dostupné z: https://codeinstitute.net/global/blog/the-best-coding-languages-for-web-development-in-2022/. [cit. 2024-05-02].

12. *MathJax*. Online. 2009, 2023. Dostupné z: https://www.mathjax.org/. [cit. 2024-03-01].

13. CASTILHO, Bernardo. *DragDropTouch*. Online. GitHub. 2016. Dostupné z: https://github.com/Bernardo-Castilho/dragdroptouch. [cit. 2024-02-11].

14. MICROSOFT. *Documentation for Visual Studio Code*. Online. 2024. Dostupné z: https://code.visualstudio.com/docs. [cit. 2024-05-02].

15. *O programu GeoGebra*. Online. GEOGEBRA. GeoGebra. 2024. Dostupné z: https://www.geogebra.org/about. [cit. 2024-05-10].