UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MATEMATICKÁ ČÍTANKA

BAKALÁRSKA PRÁCA

2022 Barbora FORGÁČOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MATEMATICKÁ ČÍTANKA

BAKALÁRSKA PRÁCA

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Vedúci práce:	RNDr. Peter Borovanský, PhD.
Konzultant:	RNDr. Michal Winczer PhD.

Bratislava 2022 **Barbora FORGÁČOVÁ**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Barbora Forgáčová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Matematická čítanka
Math Textbook

Anotácia: Existujúca matematická čítanka vytvorená RNDr. Vladimírom Jodasom z GJH Bratislava rozvíja matematické myslenie vo vybraných kapitolách stredoškolskej matematiky. Lenže má formát textov (.doc), excelov (.xls) a geometrických konštrukcií v prostredí Cabri Geometria, ale v neaktuálnej a platenej verzii. Cieľom práce je materiály adaptovať na dnešné technológie, vytvoriť interaktívnu učebnicu, ktorá bude obsahovať pôvodné texty (napr. html), prepojenie na nejakú obdobu excelovských tabuliek a grafov a Geogebru (namiesto geometrických konštrukcií v Cabri geometrii). Od autora sa neočakáva žiaden metodický posun predloženého materiálu, ale navrhnúť elektronickú online verziu tak, aby bola interaktívna a príjemná pre žiakov 3.stupňa. Ak to situácia dovolí, prvé testovanie so žiakmi v triede sa predpokladá v apríli, druhé testovanie v triede sa predpokladá v júni.

Literatúra: https://jupyter.org/
https://www.geogebra.org/

Vedúci: RNDr. Peter Borovanský, PhD.
Konzultant: RNDr. Michal Winczer, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 20.09.2020

Dátum schválenia: 03.11.2020 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Ďakujem môjmu vedúcemu za pomoc a podporu pri písaní tejto práce.

Abstrakt v štátnom jazyku

FORGÁČOVÁ, Barbora: Matematická čítanka [Bakalárska práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej informatiky; školiteľ: RNDr. Peter Borovanský, PhD., Bratislava, 2022, 40 s.

Cieľom tejto bakalárskej práce bolo vytvorenie interaktívnej matematickej čítanky z materiálov vytvorených RNDr. Vladimírom Jodasom, tak aby bolo jej použitie jednoduché a aby si ju mohol pozrieť každý používateľ a nie len ten kto má jej materiály k dispozícii. Zautomatizovali sme prerábanie niektorých materiálov, pri ktorých to bolo možné a ostatné sme prerobili ručne. Týmto spôsobom sa nám podarilo vytvoriť novú, modernejšiu, lepšiu verziu čítanky.

Kľúčové slová: GeoGebra, Cabri, čítanka, DeadLine, Excel

Abstract

FORGÁČOVÁ, Barbora: Math Textbook [Bachelor Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Informatics; Supervisor: RNDr. Peter Borovanský, PhD., Bratislava, 2021, 40 p.

Target of this bachelor's thesis was to create interactive math notebook using materials created originally by Rndr. Vladimír Jodas, so that it is easy to use and so that it can be viewed by every user and not just by those who have its materials at their disposal. We automated conversion of some materials, when possible and other materials we converted manually. This way we have created new, modern, better version of notebook.

Keywords: GeoGebra, Cabri, notebook, DeadLine, Excel

Obsah

Úvod

Žijeme v dobe kedy už dnešná generácia pomaly nevie čo je to kniha, alebo ich aspoň nevyhľadáva v takom množstve ako kedysi. Ak si chcú ľudia niečo zistiť, tak hľadajú práve na internete.

Matematika je dôležitou súčasťou vzdelania, stretávame sa s ňou pomaly každý deň. Pred niekoľkými rokmi RNDr. Vladimír Jodas napísal skvelú matematickú čítanku, ktorá interaktívnym spôsobom vysvetľuje vybrané matematické témy. Túto čítanku napísal už v elektronickej podobe avšak v programoch, ktoré sa dnes už veľmi nepoužívajú a máme lepšie a modernejšie náhrady. Materiály tejto čítanky sú rozvrhnuté v rôznych formátoch a ich spúšťanie vyžaduje aby sme mali k dispozícii softvéry, ktorými vieme tieto formáty otvárať. Pre dnešných žiakov nemusí byť veľmi atraktívne čítať a učiť sa z textov napísaných vo worde a preklikávať sa medzi súbormi, keď v tejto dobe vedia mať všetko pokope.

Naším cieľom je prerobiť túto matematickú čítanku, tak aby bola atraktívna nielen pre žiakov ale aj pre ostatných. Adaptovať existujúce materiály na dnešné technológie a vytvoriť tak interaktívnu učebnicu.

V prvej kapitole tejto práce si rozoberieme ako vyzerala pôvodná čítanka, popíšeme si existujúce matematické softvéry, ktoré by sme mohli pri našej práci použiť. V druhej kapitole sa zamyslíme ako by mala naša stránka vyzerať, navrhneme si približne jej dizajn. Ďalej si popíšeme aké novodobé technológie využijeme a zamyslíme sa nad rôznymi riešeniami. V poslednej kapitole si popíšeme ako sme postupovali pri prerábaní materiálov do nových formátov, na aké problémy sme narazili a ako sme ich vyriešili.

1 Teoretické východiská

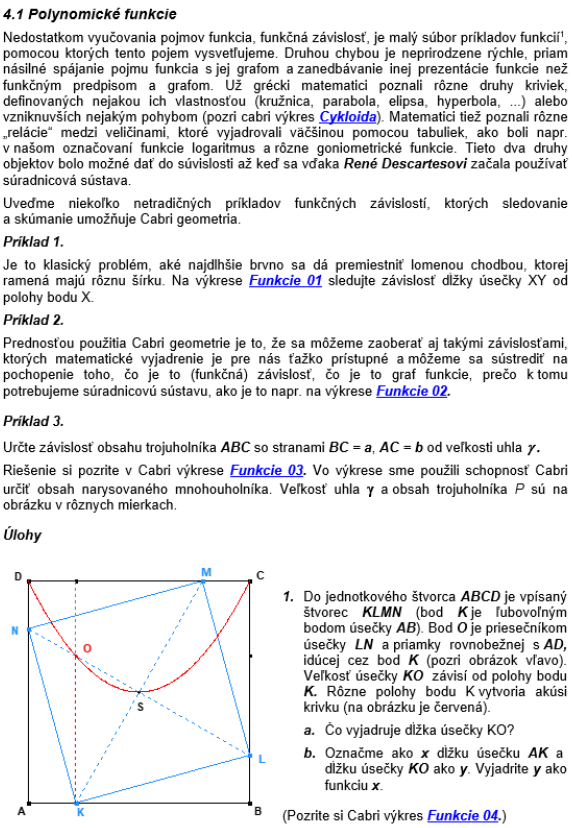
V tejto kapitole si rozoberieme ako vyzerajú druhy súborov, ktoré máme k dispozícii, navrhneme si a popíšeme vhodné existujúce riešenia.

1.1 Zdrojové súbory

Matematická elektronická čítanka (ďalej už len čítanka), ktorú vytvoril RNDr. Vladimír Jodas, obsahuje 26 wordových súborov, 87 excelových súborov, 110 Cabri súborov a 42 Deadline súborov. Aby mohol používateľ s čítankou pracovať potrebuje mať k dispozícii word, excel (alebo nejaký iný softvér, ktorý podporuje zobrazovanie excelovských tabuliek), softvéry DeadLine a Cabri.

1.1.1 Wordové súbory (.doc)

Wordové súbory obsahujú úvod, kde je predstavená čítanka ako taká a čo by sa malo nachádzať na počítači používateľa aby bol schopný s čítankou pracovať. Ďalej sa v nej nachádza mapa čítanky, vysvetlené učivo, rôzne príklady, úlohy a riešenia, hypertextové odkazy pomocou, ktorých si používateľ môže otvoriť mikro projekty, ktoré sú súčasťou tejto čítanky, ale takisto aj odkazy na doplňujúce materiály (niektoré odkazované stránky dnes už nie sú funkčné) alebo vhodné matematické applety.



4.1 Polynomické funkcie

Nedostatkom vyučovania pojmov funkcia, funkčná závislosť, je malý súbor príkladov funkcií[1], pomocou ktorých tento pojem vysvetľujeme. Druhou chybou je neprirodzene rýchle, priam násilné spájanie pojmu funkcia s jej grafom a zanedbávanie inej prezentácie funkcie než funkčným predpisom a grafom. Už grécki matematici poznali rôzne druhy kriviek, definovaných nejakou ich vlastnosťou (kružnica, parabola, elipsa, hyperbola, ...) alebo vzniknuvších nejakým pohybom (pozri cabri výkres ***Cykloida***). Matematici tiež poznali rôzne „relácie" medzi veličinami, ktoré vyjadrovali väčšinou pomocou tabuliek, ako boli napr. v našom označovaní funkcie logaritmus a rôzne goniometrické funkcie. Tieto dva druhy objektov bolo možné dať do súvislosti až keď sa vďaka ***René Descartesovi*** začala používať súradnicová sústava.

Uveďme niekoľko netradičných príkladov funkčných závislostí, ktorých sledovanie a skúmanie umožňuje Cabri geometria.

Príklad 1.

Je to klasický problém, aké najdlhšie brvno sa dá premiestniť lomenou chodbou, ktorej ramená majú rôznu šírku. Na výkrese ***Funkcie 01*** sledujte závislosť dĺžky úsečky XY od polohy bodu X.

Príklad 2.

Prednosťou použitia Cabri geometrie je to, že sa môžeme zaoberať aj takými závislosťami, ktorých matematické vyjadrenie je pre nás ťažko prístupné a môžeme sa sústrediť na pochopenie toho, čo je to (funkčná) závislosť, čo je to graf funkcie, prečo k tomu potrebujeme súradnicovú sústavu, ako je to napr. na výkrese ***Funkcie 02***.

Príklad 3.

Určte závislosť obsahu trojuholníka ***ABC*** so stranami ***BC = a***, ***AC = b*** od veľkosti uhla ***γ***.

Riešenie si pozrite v Cabri výkrese ***Funkcie 03***. Vo výkrese sme použili schopnosť Cabri určiť obsah narysovaného mnohouholníka. Veľkosť uhla γ a obsah trojuholníka P sú na obrázku v rôznych mierkach.

Úlohy

1. Do jednotkového štvorca ***ABCD*** je vpísaný štvorec ***KLMN*** (bod ***K*** je ľubovoľným bodom úsečky ***AB***). Bod ***O*** je priesečníkom úsečky ***LN*** a priamky rovnobežnej s ***AD***, idúcej cez bod ***K*** (pozri obrázok vľavo). Veľkosť úsečky ***KO*** závisí od polohy bodu ***K***. Rôzne polohy bodu K vytvoria akúsi krivku (na obrázku je červená).

a. Čo vyjadruje dĺžka úsečky KO?

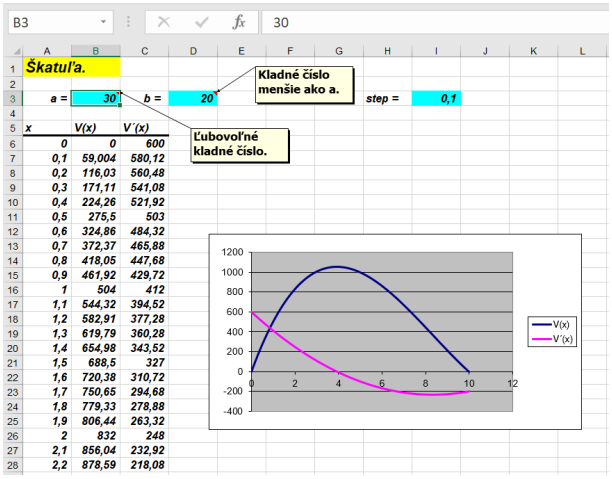
b. Označme ako ***x*** dĺžku úsečku ***AK*** a dĺžku úsečky ***KO*** ako ***y***. Vyjadrite ***y*** ako funkciu ***x***.

(Pozrite si Cabri výkres ***Funkcie 04***.)

Obr. 1: Ukážka wordového súboru čítanky s učivom a odkazmi na .fig a .ddl súbory.

1.1.2 Excelové súbory (.xls)

Pomocou excelu sú vytvorené grafy a tabuľky. Väčšina (napríklad Obr. 2) z nich je interaktívna, čiže používateľ môže zadávať rôzne hodnoty a následne sa podľa danej funkcie, graf alebo iné hodnoty v tabuľkách zmenia.



Obr. 2: Ukážka excelovského súboru čítanky - tabuľka s grafmi.

1.1.3 Cabri súbory (.fig)

Cabri Geometry

Cabri Geometria, ďalej len Cabri, je softvér, ktorý umožňuje vytvarať geometrické objekty, s ktorými sa dá manipulovať, skúmať a objavovať ich vlastnosti. Prvá verzia vznikla v roku 1988 na Univerzite Josepha Fouriera v Grenoblu pod vedením Jean-Marie Laborda. Cabri je výsledkom neustálej a plodnej spolupráce medzi informatikmi, matematikmi, špecialistami na vzdelávanie a praktickými učiteľmi. [3]

Používa sa na výučbu geometrie na stredných školách, na výučbu na univerzitnej úrovni a ako pomôcka pre matematikov pri ich výskumnej práci. Množstvo publikácií v pedagogických časopisoch aj na internete svedčí o tom, že ide o pomôcku, ktorá významne prispieva k výuke geometrie. [3]

Dokument programu Cabri Geometrie je obrázok, ktorý sa dá vytvárať kdekoľvek na virtuálnom liste papieru o rozmere 1 m $\times$ 1 m. Obrázok sa skladá z obvyklých geometrických objektov (body, priamky, kružnice, ...) a z ďalších objektov (čísla, texty, vzorce, ...). [2]

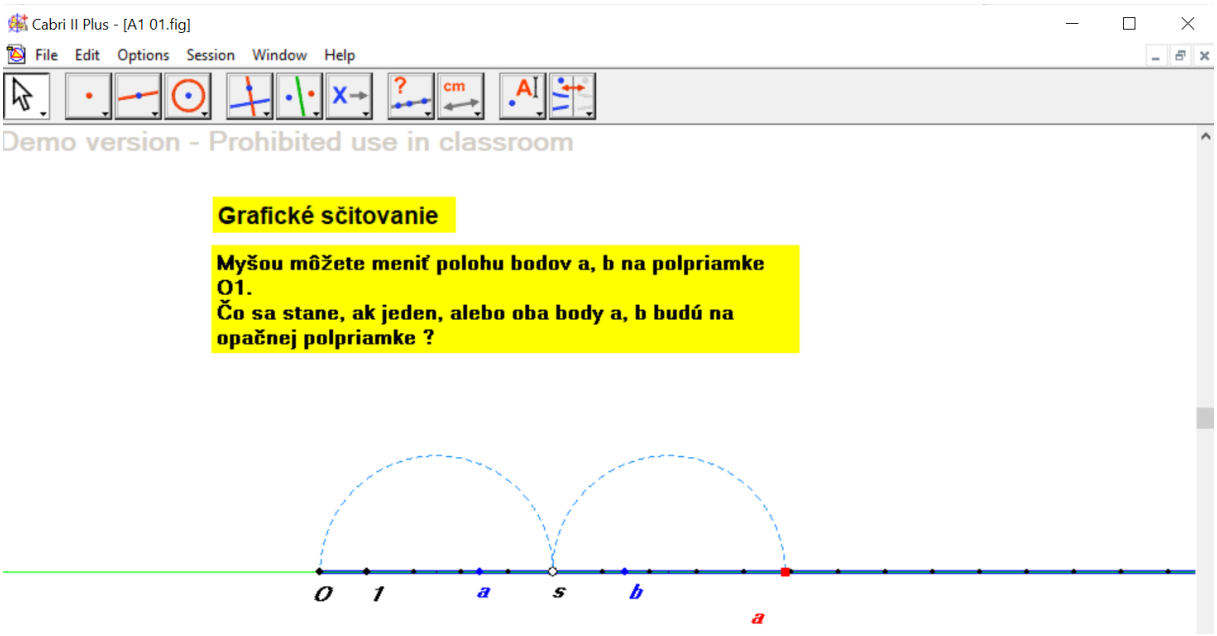
Problémom je, že Cabri je komerčný softvér a teda každý kto by chcel s touto čítankou pracovať by musel mať softvér zakúpený. Je možné stiahnuť si demoverziu Cabri II Plus, v ktorej idú otvoriť dané súbory, ale relácia vyprší za 15 minút a teda používateľ by si musel otvárať Cabri za každým, keď tento čas ubehne, alebo je ešte možné aktivovať si skušobnú verziu, ktorá ale platí len na jeden mesiac.

Súbory

Po spustení Cabri súboru sa nám okrem daného súboru otvorí aj Assistant okno, v ktorom sa nachádza:

- **Cabri Tour** - obsahuje video so všeobecným popisom a ukážkou programu
- **Getting started** - obsahuje rôzne tutoriály ako pracovať s programom
- **In my classroom** - obsahuje príklady, ktoré si môžeme spustiť a vyskúšať prácu s nimi

Cabri súbory obsahujú rôzne interaktívne ukážky a príklady, vďaka ktorým je jednoduchšie porozumieť preberanej látke, používateľ môže hýbať bodmi alebo meniť hodnoty a následne sledovať zmeny.

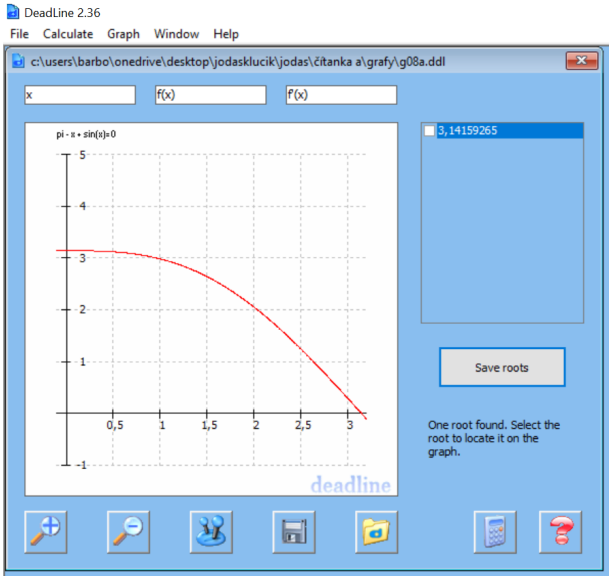


Obr. 3: Ukážka Cabri súboru.

1.1.4 DeadLine súbory (.ddl)

DeadLine je matematický softvér (voľne dostupný), ktorý pomáha pri riešení zložitých matematických rovníc, ktoré vykresľuje do grafov. Rieši rovnice vizuálne aj numericky. [4]

V čítanke tieto súbory slúžia ako ukážka grafov daných rovníc, používateľ ich nepotrebuje nijako meniť.



Obr. 4: Ukážka DeadLine súboru.

1.2 Existujúce riešenia

1.2.1 Jupyter Notebook

Jupyter Notebook, ďalej len Jupyter, je open source-ová aplikácia, ktorú môžeme použiť na vytváranie a zdieľanie dokumentov obsahujúcich kód, rovnice, vizualizácie a text. Jupyter kombinuje dva komponenty:

- Webová aplikácia - prehliadačový nástroj na interaktívne vytváranie dokumentov, ktorý kombinuje vysvetľujúci text, matematiku, výpočty a ich multimediálne výstupy.

- Dokumenty poznámkového bloku - znázornenie všetkého obsahu viditeľného vo webovej aplikácii vrátane vstupov a výstupov výpočtov, vysvetľujúceho textu, matematiky, obrázkov a multimediálnych zobrazení objektov. [5]

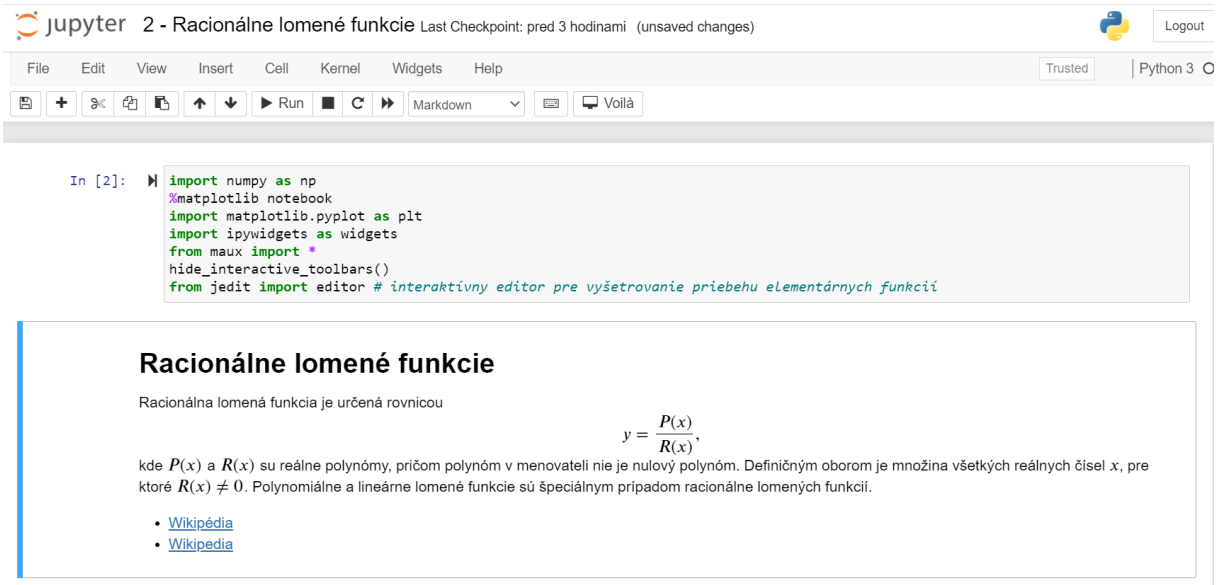
Výhodou je, že môžeme vidieť kód aj výsledky a prevádzkovať bunku po bunke, aby sme lepšie pochopili, čo tento kód robí. Medzi nevýhodu patrí napríklad to, že keď píšeme

kód do buniek namiesto do funkcií, tried alebo objektov, rýchlo získame duplicitný kód, ktorý robí to isté, čo sa veľmi ťažko udržiava. [6]

S Jupyterom sa študenti aplikovanej informatiky stretli na matematickej analýze, kde pomocou neho riešili zadané úlohy. Bolo potrebné aby si študenti stiahli softvér Anaconda, pomocou ktorého sa potom Jupyter spúšťal.

Po spustení sa zobrazia priečinky v používanom zariadení, kde si nájdeme súbor, ktorý chceme otvoriť.

Po otvorení súboru hneď na vrchu vidíme prvú bunku s pythonovským kódom, kde sa importujú knižnice, ktorý treba spustiť pomocou tlačidla Run. Ak by sa to nespustilo, tak zvyšok buniek by pri spustení vyhodil error.



Obr. 5: Prvá bunka - importy knižníc.

Pri spustení ostatných buniek s kódom sa nám vytvorí graf, ktorému môžeme v kóde zadať rôzne parametre (napríklad meno, aby sa vyznačili priesečníky, maximá alebo farbu).

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown Voilà

Príklad

Nakreslenie grafu racionálne lomenej funkcie danej rovnicou

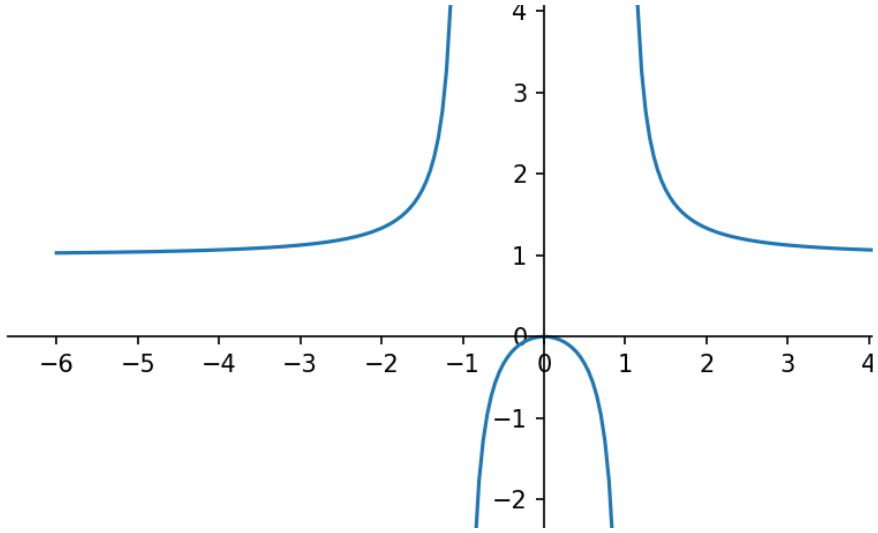
$$y = \frac{x^2}{x^2 - 1}.$$

Definičným oborom je množina všetkých reálnych čísel, ktoré nepatria do množiny $\{-1, 1\}$.

In [1]:

```
#####
##### nakreslenie grafu funkcie
#####

#### vstupné údaje
def f(X): return X ** 2 / (X ** 2 - 1) # ufunc verzia funkcie
X = np.linspace(-6, 6, 12*20+1) # výber hodnôt nezávislej premennej pre zaujímavú časť grafu
X1, X2, X3 = X[X < -1], X[(X > -1) & (X < 1)], X[X > 1] # čísla -1, 1 nepatria do oboru definície
Y1, Y2, Y3 = f(X1), f(X2), f(X3) # odpovedajúce hodnoty závislej premennej

#### obrázok s jedným diagramom
fig, ax = plt.subplots()
fig.set_size_inches(8, 8) # veľkosť obrázka (východzia hodnota je 6x4)

### diagram
init_subplot(ax) # inicializácia diagramu: vytvorí sa pravoúhla súradnicová sústava
ax.set_title(r"Graf funkcie $y = \dfrac{x^2}{x^2-1}$", fontdict={'verticalalignment': 'bottom'}) # pomenovanie diagramu
ax.set_aspect('equal') # nastavenie rovnakej mierky pre obe osi
```

Obr. 6: Ukážka príkladu v Jupyter Notebooku - zadanie + kúsok kódu.



Obr. 7: Ukážka príkladu v Jupyter Notebooku - výsledok kódu.

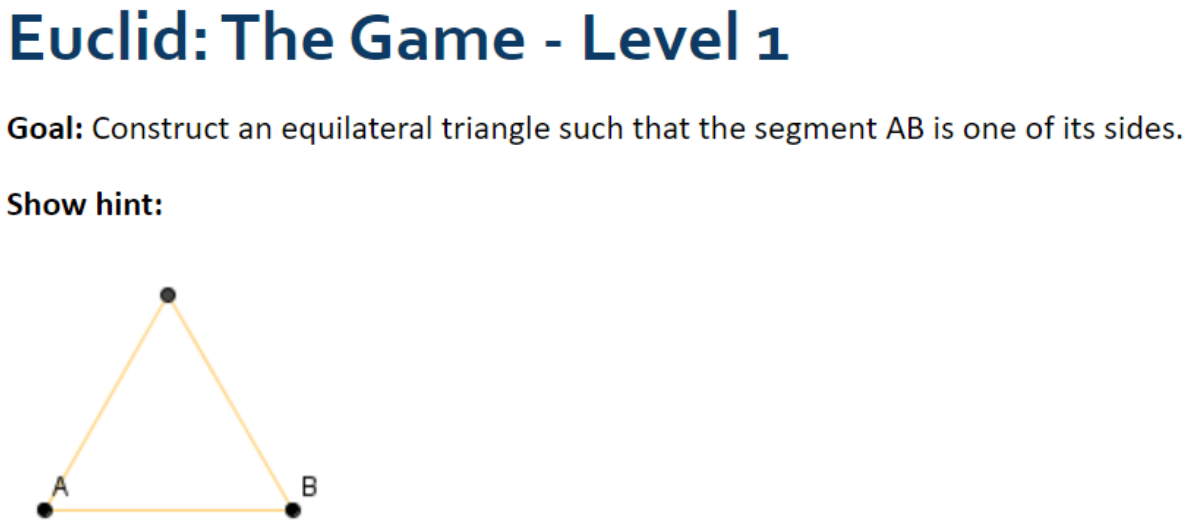
1.2.2 GeoGebra

GeoGebra je dynamický matematický software pre všetky úrovne vzdelávania, ktorý spojuje geometriu, algebru, tabulkový procesor, grafy, štatistiku a analýzu do jedného ľahko použiteľného balíčka. Výdatne pomáha pri vizualizácii konštrukčných úloh v rovine či v priestore, alebo zobrazuje priebeh a vlastností prakticky ľubovoľnej funkcie. [7]

Jeho veľkou výhodou je voľná dostupnosť a veľké množstvo používateľov, čo zabezpečuje mnoho dostupných pomocných materiálov. Takisto medzi výhody patrí aj veľké

množstvo jazykov, do ktorých je GeoGebra preložená. Môžeme používať priamo webové prostredie GeoGebry alebo si môžeme nainštalovať aplikáciu. [8]

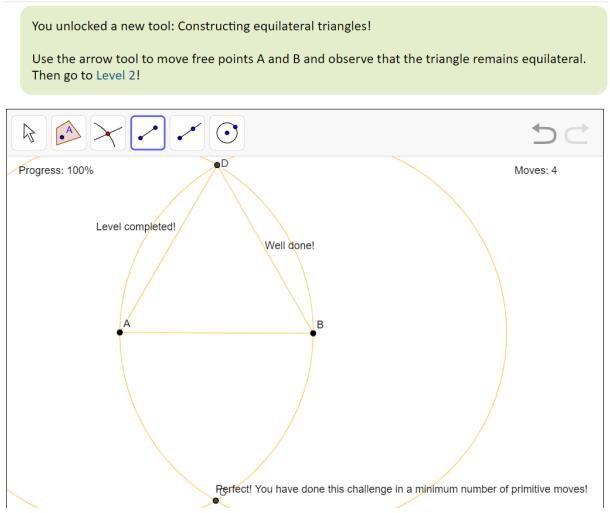
Na internete vieme nájsť veľa fascinujúcich príkladov použitia GeoGebry. Medzi ne patrí napríklad hra Euclid, ktorej tvorcom je Kasper Peulen. Hra je zložená z niekoľkých levelov, v každom z nich je potrebné splniť daný cieľ aby sme sa mohli posunúť do ďalšieho levela.



Obr. 8: Euclid - goal.

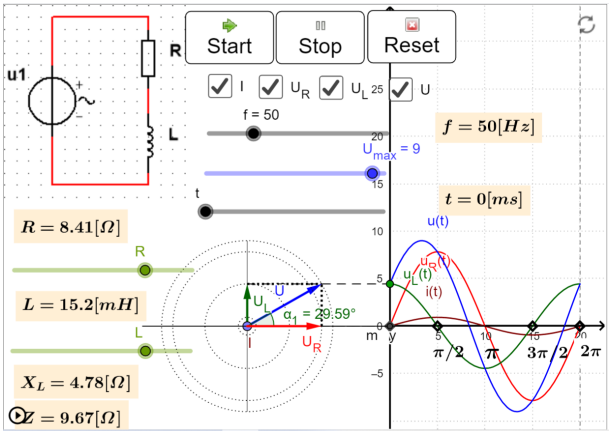
Pracujeme pri tom s oknom GeoGebry, ktoré je vložené pomocou JavaScriptu. JavaScript tu má množstvo použitia. Za každým, keď spravíme správny krok, tak sa nám zvýšia percentá v časti progess.

Keď level splníme tak sa môžeme posunúť ďalej. Ak sa nám podarí splniť level minimálnym počtom ťahov, tak sa nám to tiež vypíše.



Obr. 9: Euclid - prostredie GeoGebra, dokončený level.

Nájdu sa tu samozrejme aj nejaké chyby, ale aj tak je to veľmi pekná ukážka ako sa dá pracovať s GeoGebrou. A teda môžeme vidieť, že sa dá pekne prepojiť GeoGebra s html.



Obr. 10: GeoGebra - ďalšia ukážka.

2 Návrh

V tejto kapitole si navrhneme ako by mala naša webová stránka vyzerať, akým spôsobom zmeníme pôvodné materiály čítanky a popíšeme si rôzne možnosti, nad ktorými sme sa zamýšľali.

2.1 Hlavný cieľ webstránky

V sekcii 1.1 sme si popísali z čoho sa skladá pôvodná čítanka. Naším cieľom je pretvoriť tieto materiály do novodobejšej podoby, tak aby bola prehľadná a pohodlná pre používateľov. Riešením je vytvorenie webstránky, v ktorej sa budú nachádzať všetky súbory bez toho aby si musel používateľ sťahovať alebo kupovať nejaký softvér a bez zbytočného otvárania preklikávania sa medzi súbormi.

2.2 Návrh webstránky

Pri vytváraní webovej stránky je veľmi dôležité uvedomiť si kto bude primárne stránku navštevovať a premyslieť si akú by mala mať funkcionalitu a ná základe toho by sme si mali navrhnúť jej dizajn. Grafický dizajn stránky by mal byť intuitívny, aby sa používateľ vedel na stránke jednoducho orientovať.

2.2.1 Používateľské rozhranie

Našu webstránku môže navštíviť hocijaký používateľ, či už žiak, študent, učiteľ alebo len niekto kto sa chce dozvedieť viac o danej problematike. Používateľov nepotrebujeme medzi sebou rozlišovať a preto nepotrebujeme riešiť žiadne prihlasovanie a dizajn stránky je tým pádom rovnaký pre všetkých. Stránku sme navrhli tak aby obsahovala tri základné podstránky, a to "Úvod", "Čítanka“ a "Literatúra". V prvej podstránke sa bude nachádzať úvod, v ktorom nás autor oboznamuje s čítankou a so svojimi myšlienkami.

V podstránke "Čítanka"sa bude nachádzať obsah čítanky s hypertextovými odkazmi na jednotlivé kapitoly a podkapitoly a so samotným textom čítanky.

V poslednej podstránke s názvom "Literatúra"sa čitateľ dozvie odkiaľ autor naberal inšpiráciu pri písaní čítanky, na niektoré knihy / odkazy sa odkazuje v kapitolách samotnej čítanky.

2.2.2 Texty

Ako sme si spomenuli v kapitole 1.1.1, tak wordové súbory obsahujú texty, odkazy, obrázky, rovnice a tabuľky. Wordové súbory teda môžeme prepísať do html. Keďže čítanka obsahuje rôzne matematické výrazy a rovnice, tak potrebujeme vyriešiť ako ich písať v html. Na to by sme mohli využiť nejakú javascriptovú knižnicu.

2.2.3 Grafy a konštrukcie

V sekcii 1.2.2 sme usúdili, že na prerobenie grafov a matematických konštrukcií je najvhodnejšia GeoGebra. Chceme dosiahnuť to aby boli konštrukcie graficky čo najautentickejšie k pôvodným konštrukciám, ktoré boli pôvodne vytvorené v Cabri. Čo sa týka grafov, tam by sme chceli pridať priamo do grafického okna hľadanie priesečníkov, deriváciu funkcie a podobne aby sa používateľ nemusel zbytočne prehrábavať funkciami GeoGebry.

2.2.4 Tabuľky

Zamysleli sme sa ako by sme mohli zakomponovať tabuľky do webstránky. Prvou možnosťou je využiť doplnok Excelu, publisheet, ktorý nám umožňuje publikovať jednotlivé tabuľky ako interaktívne webové stránky uložené v cloude, ktoré môžeme jednoducho pridať pomocou elementu iframe do našej webovej stránky. Na Obr.11 môžeme vidieť ukážku takéhoto excelu. Pri takomto spôsobe prevedenia nastávajú dva problémy. Prvý je ten, že publisheet vie previesť maximálne 1000 buniek, pričom niektoré naše excely ich obsahujú o dosť viac. Ten druhý je taký, že autor čítanky v niektorých prípadoch očakáva od čitateľa aby na základe tabuľky vytvoril novú tabuľku (čo sa v takomto prípade nedá spraviť) alebo jednoducho ak by čitateľ chcel vidieť aké vzorce sa skrývajú za výpočtom tak si to pozrieť nemôže. Druhý spôsob, nad ktorým sme sa zamýšľali bola JavaScriptová knižnica SheetJS, pomocou ktorej by sme mohli tabuľky vytvoriť pomocou JavaScriptu. Tu ale nastáva taký problém, že by bolo potrebné si zakúpiť PRO Charts + Formula Calculator (balíčky tejto knižnice, potrebné k tomu aby sme mohli tabuľky vytvoriť) za nemalú sumu. Premýšľali sme aj nad google spreadsheets, ktoré by sme do našej webovej stránky vložili pomocou elementu iframe, tam ale ak používateľ zmení niečo v tabuľke tak to zmení na trvalo a to nechceme. Ako posledný spôsob sme sa pozreli na tabuľky v GeoGebre, ktoré sú funkcionalitou rozhodne najbližšie k excelu. Používateľ si môže pozrieť aký výpočet

je za bunkou, môže si vytvoriť tabuľky podľa seba a GeoGebra applet môžeme pomocou JavaScriptu pridať do webstránky. Aj pri tejto možnosti nastáva problém s príliš veľkým množstvom buniek, ak sa presiahne nejaký počet tak to GeoGebra prestane zvládať a začína sekať a nedá sa s ňou v takom prípade už pracovať, navyše GeoGebra obsahuje maximálne 350 riadkov.

Škatuľa.

a = 30 b = 20 step = 0,1

x	V(x)	V´(x)
0	0	600
0,1	59,004	580,12
0,2	116,032	560,48
0,3	171,108	541,08
0,4	224,256	521,92
0,5	275,5	503
0,6	324,864	484,32
0,7	372,372	465,88
0,8	418,048	447,68
0,9	461,916	429,72
1	504	412
1,1	544,324	394,52
1,2	582,912	377,28
1,3	619,788	360,28
1,4	654,976	343,52
1,5	688,5	327
1,6	720,384	310,72

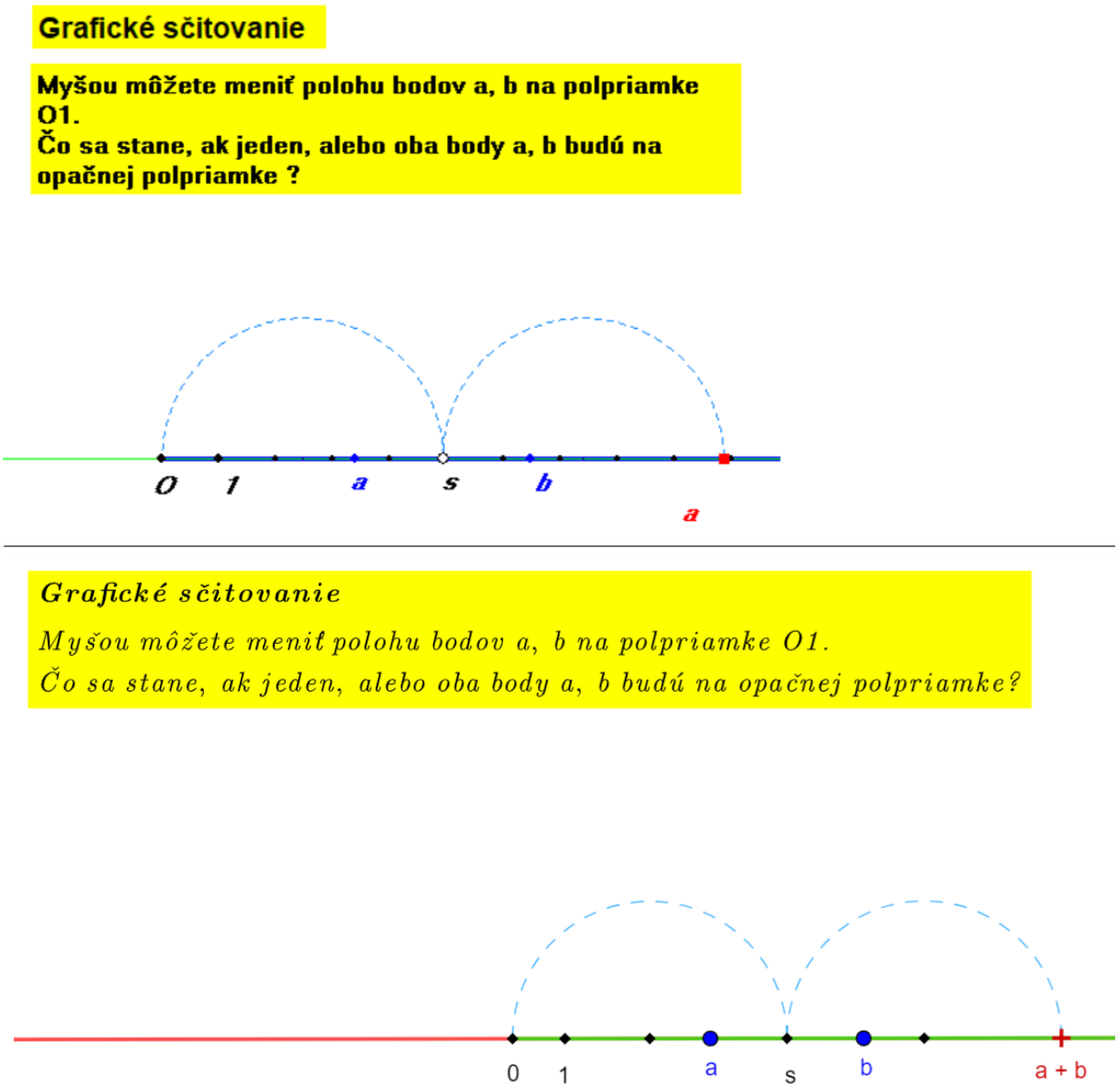
Obr. 11: Ukážka tabuľky vytvorenej pomocou publisheetu.

3 Implementácia

V tejto kapitole si popíšeme ako sme analyzovali a prerábali pôvodné materiály do novej podoby a ako sme ich implementovali do webstránky, ako sme postupovali pri vytváraní našej webovej stránky a niečo o jej dizajne.

3.1 Konštrukcie z Cabri do GeoGebry

Najprv sme si otvorili prvú konštrukciu v Cabri (.fig súbor) a zisťovali sme ako funguje. Hýbali sme bodmi a sledovali ako sa konštrukcia správa. Následne sme sa ju pokúsili vytvoriť v GeoGebre (na Obr. 12 môžeme vidieť ukážky tej istej konštrukcie v oboch softvéroch). Takýto spôsob je veľmi neefektívny a trvá veľmi dlho, pretože pochopiť čo je za vytvorenou konštrukciou, zistiť čo je s čím a ako poprepájané je náročné a nie vždy možné.



Obr. 12: Konštrukcia v Cabri (horný obrázok) a v GeoGebre (spodný obrázok).

Otvorili sme si .fig súbor prvej konštrukcie v textovom editore a začali sme analyzovať čo by mohli jednotlivé riadky znamenať. Keby sme mali Cabri Geometriu zakúpenú tak by pravdepodobne stačilo skúšať rôzne nástroje ako vytvorenie priamky, kružnice, symetrie a ďalšie veci alebo iba meniť už vytvorené konštrukcie. Následne by stačilo pozorovať zmeny v textovom editore. V našej neplatenej verzii síce môžeme meniť konštrukcie ale nemôžeme ich ukladať a teda takýmto spôsobom zmeny pozorovať nemôžeme. Funguje to ale naopak, vieme urobiť zmeny v textovom editore a následne ich pozorovať v Cabri. Takýmto spôsobom sme ako-tak dokázali zanalyzovať čo znamenajú jednotlivé riadky.

Pomocou Cabri manuálu sme si dokázali odvodiť jednotlivé skratky nástrojov a zistili sme ako ich použiť.

Nazvyme prvkom všetko čo môžeme vytvoriť v Cabri alebo v GeoGebre, (bod, priamka, úsečka, trojuholník, uhol, rotácia bodu, premenná, ...).

```
Figure Cabri II Plus vers. MS-Windows 1.2.5

Window center x: 0.66cm y: 0.05cm Window size x: 26.50cm y: 16.03cm
Resolution: 38 ppc

1: Pt, Val: -5.76315789473684 -1.42105263157895
color:B, thicker, 1x1, invisible,

2: Axes, Const: 1, cart, Val: 1 0 0 1
invisible,

3: Grid, Const: 2, cart,
invisible,

4: Pt/, Const: 3, Val: -4.76315789473684 -1.42105263157895
color:B, thick, 1x1,
"1"
TP:  -4.76315789473684,  -1.63157894736842, TS:  0.5,  0.473684210526316
p: 0, Arial CE, S: 12, C: 40, Fa: 3, p: 1, System, S: 10, C: 40, Fa: 1

5: Pt/, Const: 3, Val: -3.76315789473684 -1.42105263157895
color:B, thicker, 1x1,

6: Pt/, Const: 3, Val: -2.76315789473684 -1.42105263157895
color:B, thicker, 1x1,

7: Pt/, Const: 3, Val: -1.76315789473684 -1.42105263157895
color:B, thicker, 1x1,

8: Pt/, Const: 3, Val: -0.763157894736842 -1.42105263157895
color:B, thicker, 1x1,

9: Pt/, Const: 3, Val: 0.236842105263158 -1.42105263157895
color:B, thicker, 1x1,

10: Ray, Const: 1,  4,
color:Bl, thicker,
```

Obr. 13: Prvá konštrukcia v Cabri otvorená v textovom editore.

Na Obr. 13 vidíme ako vyzerá útržok .fig súboru prvej konštrukcie otvorený v textovom editore. Najprv si vysvetlíme, ktoré informácie nás budú v tomto konkrétnom prípade zaujímať a čo znamenajú. Prvých pár riadkov uvádza akú verziu Cabri máme a nastavenia okna, tieto informácie nás vôbec nebudú zaujímať. Ďalej môžeme vidieť, že text je rozdelený na malé paragrafy oddelený medzerami a vždy na začiatku paragrafu sa nachádza číslo, vždy začíname od čísla 1. Môžeme teda predpokladať, že to je poradie ako boli nástroje použité a teda ako konštrukcia vznikla ale takisto nám to označuje číslo daného prvku, s ktorým neskôr pracujeme. Za číslom vidíme buď skratku alebo celé slovo, ktoré nám hovorí o aký prvok ide. Napríklad 'Pt' je skratka pre Point, teda celý paragraf nám udáva ako je daný bod vytvorený, 'Ray' je polpriamka, 'Mid' stredový bod a podobne. Zaujímať nás bude ešte hodnota 'Const' a 'Val'. 'Const' môže obsahovať jedno alebo viacero čísiel. Pokiaľ je to číslo iba jedno, tak nám to udáva, že tento nový vytvorený prvok sa nachádza na prvku, ktorý bol vytvorený v paragrafe s daným číslom. Čiže napríklad ak by sme mali nejakú priamku, označme ju číslom 3 a následne vytvoríme bod, ktorý by mal pri 'Const' hodnotu 3, tak by to znamenalo, že tento bod sa nachádza na našej

priamke s číslom 3. Ak by bolo hodnôt viac, tak nám to hovorí o tom, že ktoré prvky využíva pri vytvorení. Napríklad majme úsečku, ktorá obsahuje v 'Const' hodnoty 1 a 2, kde 1 a 2 sú body, tak by to znamenalo, že koncové body našej úsečky sú práve tieto dva body. Je zrejmé, že novovytvorený prvok môže mať v 'Const' len také prvky, ktoré už boli vytvorené pred našim novým prvkom. 'Val' nám udáva buď konkrétne súradnice alebo hodnotu daného prvku, stretneme sa s ním hlavne pri vytváraní bodov alebo pri premenných. Ďalej sa v paragrafoch môžu nachádzať hodnoty, ktoré nám uvádzajú farbu prvku, či je prvok viditeľný alebo nie, hrúbku prvku a veľa iných vecí, ktoré nás ale nebudú zaujímať.

Súčasne sme porovnávali vytváranie prvkov s manuálom GeoGebry a jej príkazmi. Dôležité je poznamenať, že tak ako sa k prvku v Cabri dostávame cez číslo, tak v GeoGebre sa k prvku dostávame cez písmeno. Ukážeme a porovnáme si pár príkladov.

1. Vytvorenie bodu

   - Cabri - Podľa našich zistení sme prišli na to, že v Cabri vieme vytvoriť dva druhy bodov. Prvý je označený ako 'Pt' a druhý ako 'Pt/', rozdiel medzi týmito dvomi bodmi je nasledovný, pri 'Pt' môžeme vidieť, že sa nachádza len hodnota 'Val', ktorá nám určuje súradnice bodu. Pri 'Pt/' sa nachádza hodnota 'Val' aj 'Const'. Ako sme si vysvetlili na príklade vyššie, tak to znamená, že bod sa nachádza na nejakom prvku na daných súradniciach.
   - GeoGebra - V GeoGebre vieme takisto vytvoriť bod dvomi spôsobmi (momentálne sa bavíme len o vytváraní prvkov pomocou príkazového riadku). Ak chceme vytvoriť bod s danými súradnicami, tak zadáme príkaz 'A = (-5, 4)', kde A je názov prvku (ak by sme ho nezadali tak GeoGebra vytvorí názov sama) a (-5, 4) sú súradnice (x, y), na ktorých sa náš bod A nachádza. V druhom prípade majme úsečku h, pomocou príkazu 'B = Point(h)' vytvoríme bod B nachádzajúci sa na úsečke h. V tomto prípade narozdiel od Cabri nevieme zadať súradnice na akých sa má bod nachádzať. Môžeme ho jedine v grafickej ploche ručne posunúť.

2. Vytvorenie uhlu

   - Cabri - V Cabri je uhol označený ako 'angle' a môže obsahovať 3 hodnoty v

'Const', ktoré sú v poradí bod, vrchol, bod. Nezáleží na tom, ktorý bod zvolíme ako prvý, vždy dostaneme vnútorný uhol. Ak obsahuje 2 hodnoty tak zisťujeme uhol medzi úsečkami, priamkami alebo podobne.

- GeoGebra - Majme 3 body, A, B, C, kde B má byť vrchol. Príkaz na vytvorenie uhlu je 'Angle(C, B, A)'. Ak by sme zadali 'Angle(A, B, C)', tak by sme dostali vonkajší uhol a nie vnútorný. To isté platí pri uhle medzi úsečkami a podobne.

Ako vidíme existujú prípady kedy si musíme dávať pri tvorbe príkazov pozor na poradie čísiel prvkov v 'Const', alebo jednoducho, že niektoré veci sa nedajú urobiť rovnako a musíme ich vyriešiť inak. Napríklad v Cabri existuje prvok 'measurement transfer', ktorý jednoducho posunie daný prvok o danú hodnotu. Majme napríklad kruh, na ktorom sa nachádza bod a majme hodnotu napríklad 4. V Cabri zvolíme hodnotu, kruh a bod a vznikne nám nový bod, ktorý má ekvivalentnú dĺžku oblúka z existujúceho bodu. V GeoGebre príkaz ako 'measurement transfer' neexistuje, ale pre uvedený príklad môžeme použiť 'rotate around point', kde ale poradie našich 'Const' je iné ako v Cabri. Ak by sme toto isté chceli použiť pre napríklad polpriamku, tak v Cabri je to ten istý príkaz ale v GeoGebre sme našli pre takýto prípad iba také riešenie, že vytvoríme nový bod na našej polpriamke a potom ho ručne posunieme.

Zistili sme teda, že vieme z Cabri konštrukcie v textovom editore získať príkazy a prepísať ich na príkazy GeoGebry. Najprv sme si to overili a krok po kroku sme menili príkazy a ručne ich zadávali do príkazové riadku GeoGebry. Keď sa nám takýmto spôsobom podarilo prerobiť prvú konštrukciu, tak sme sa začali zamýšľať akoby sa to dalo zjednodušiť. Ako prvé nám napadla GeoGebra Api, ktorá bola použitá v hre Euclid, ktorú sme si popísali v 1.2.2. Tu sme ale hneď narazili na problém, že pomocou tejto Api vieme iba meniť už vytvorené applety a nevieme ich pomocou nej vytvárať. Rozhodli sme sa preto použiť Selenium WebDriver, ktorý nám umožní automaticky ovládať webový prehliadač a napísať script, ktorý nám konštrukcie vytvorí. Script sme sa rozhodli písať v pythone, ako prvé sme naimportovali Selenium WebDriver a naučili sme sa s ním pracovať.

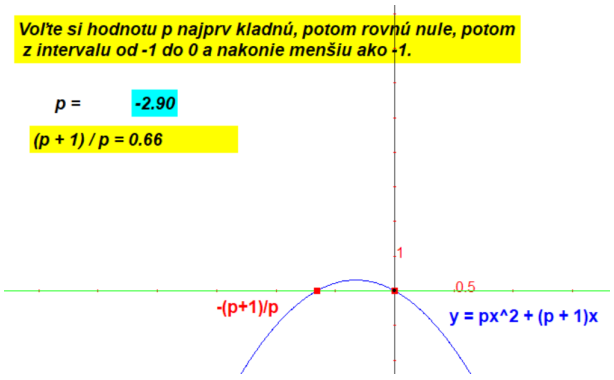
Aby sme zistili či to bude fungovať, tak sme program napísali tak aby fungoval pre prvú konštrukciu. Náš program, najprv otvoril a prečítal prvý Cabri súbor, vybral z neho len tie prvky, ktoré nás zaujímajú (veľakrát sa tam nachádzajú body, ktoré sú úplne zbytočné

a nikde ich nepoužívame). Z týchto prvkov následne zistil aký je to prvok, jeho hodnoty 'Const', 'Val', ak je to text tak obsah textu a následne ich napísal v podobe príkazov GeoGebry. S tým, že všetky body, priesečníky, symetrické body a podobne pomenovával veľkými písmenami od A po W a zvyšné prvky malými písmenami od a po w, ak sa dostal po w (alebo W), tak pomenovával prvky opäť od A (alebo a) ale pridal k nemu číslo, čiže A1-W1, a tak ďalej. Po w pomenovávame z toho dôvodu, že od x po z to GeoGebra berie špeciálne a naše príkazy sa správajú inak ako by mali (buď to zmenia na funkciu alebo z toho vytvoria vektor a podobne). Potom sa otvoril webový prehliadač, načítal stránku GeoGebry. Prihlásil nás do nášho profilu a prešiel na stránku GeoGebra Classic, v ktorej zapol input bar, do ktorého začal písať naše vytvorené príkazy. Nakoniec našu konštrukciu pomenoval a uložil. Následne sme si konštrukciu v GeoGebre otvorili a začali ju upravovať. To znamená, že prvky, ktoré nepotrebujeme vidieť sme skryli, upravili sme farby, tvary bodov, čo bolo potrebné sme pomenovali a podobne. Keď sme videli, že to funguje tak ako má, tak sme prešli na ďalšiu konštrukciu.

Program sme upravovali postupne s pribúdajúcimi novými prvkami nachádzajúcimi sa v konštrukciach. Čím viac konštrukcií sme prerobili tým viac sme pôvodný program menili, pretože sme postupne zisťovali, že aj prvky, ktoré sme si mysleli, že už ich máme vyriešené, tak sa objavil príklad kedy mal prvok napríklad miesto 3 hodnôt v "Const"5 hodnôt a poradie bolo iné ako pri tých 3 a podobne. Riešili sme rôzne komplikácie. Napríklad majme kruh c, cez ktorý ide priamka d a chceme ich priesečník. Ak by sme využili príkaz, ktorý nám dá priesečníky napríklad L = Intersect(c, d) tak dostaneme 2 priesečníky, ktoré sú pomenované GeoGebrou L_1 a L_2. Keďže nevieme dopredu koľko priesečníkov sa bude nachádzať medzi našimi objektmi, tak nevieme ako ich GeoGebra pomenuje a my nevieme zistiť koľko sa ich vytvorilo a teda náš program nemôže ďalej pracovať. Síce v takomto prípade vieme z Cabri vyčítať, ktorý priesečník nás zaujíma a vieme to aj napísať príkazom v GeoGebre (ak by nás zaujímal ľavý priesečník tak by sme len zadali L = Intersect(c, d, 1)) ale neskôr sme zistili, že v inom prípade to nefunguje. Konkrétne sa to týka ak by sme mali napríklad 2 obdĺžniky, ktoré sa prekrývajú a chceli by sme jeden konkrétny priesečník, opäť z Cabri sa dá viacmenej vyčítať, že ktorý ale v tomto prípade to nevieme GeoGebre zadať, pretože ak vytvoríme obdĺžniky v Cabri, tak sa vytvoria iba tie obdĺžniky, pričom v GeoGebre sa okrem obdĺžnikov vytvoria zvlášť aj strany, ktoré opäť nevieme zistiť ako

GeoGebra pomenovala. Riešením teda je aby náš program postupne vypisoval príkazy aj do shell okna, pokiaľ sa dostane do situácie, ktorú nevieme vyriešiť inak ako ručne, tak pozastavíme program tým, že od nás očakáva nejaký vstup a bude pokračovať až vtedy, keď ho dostane. Následne sa pozrieme na akom príkaze sa výpis pozastavil a otvoríme si Cabri súbor v textovom editore, nájdeme si tam náš konkrétny príkaz, zvolíme hodnotu na 'visible' a pozorujeme, kde sa prvok nachádza. Ak ho nevieme nájsť hneď tak všetky prvky, ktoré sú vytvorené neskôr vymažeme a náš hľadaný prvok zmeníme späť na 'invisible' a spustíme konštrukciu v Cabri. Následne zmeníme 'invisible' na 'visible' aby sme zistili aký je to prvok (v tomto prípade, ktorý priesečník nás zaujíma). Potom prejdeme do otvoreného prehliadača a pridáme tam príkaz ručne na základe našich zistení. Ak by to bol prípad s dvomi obdĺžnikmi tak musíme hľadať priesečník dvoch konkrétnych strán, ktoré sa nám vytvorili. Takýchto výnimiek je tu viacero. Ďalším problémom, na ktorý sme narazili bol prvok "Locus", konkrétne jeho priesečník s iným prvkom. Zatiaľ čo v Cabri s tým nie je žiadny problém, tak GeoGebra nedokáže spraviť s Locusom priesečník. Tento problém sme vyriešili tým, že sme si najprv zistili predpis funkcie a následne sme spravili priesečník s touto funkciou a funkciu sme skryli. Neskôr sme zistili, že samotné súradnice, ktoré zadávame sú niekedy problém. Ak sú v nesprávnom kvadrante, tak sa konštrukcie správajú inak ako by sme chceli. Poprípade ak by sme chceli zanechať čo najväčšiu autenticitu s pôvodnými konštrukciami, tak potrebujeme naše počiatočné body nastaviť na súradnice (0, 0) a podľa toho ručne poposúvať ostatné prvky.

V niektorých prípadoch sa nám neoplatí použiť náš program. Napríklad pre konštrukciu A3 11 (Obr. 14) je zbytočné aby sme konštrukciu vytvárali krok po kroku a následne skrývali prvky, ktoré nás nezaujímajú. V tomto prípade stačí ak vložíme do GeoGebry iba to čo vidíme otvorené v Cabri. Takýchto konštrukcií je viac a preto tieto konštrukcie, ktoré obsahujú iba grafy a k tomu nejaké drobnosti vytvárame ručne.



Obr. 14: Ukážka konštrukcie A3 11.

3.2 Grafy z Deadline do GeoGebry

Tak isto ako pri konštrukciách sme si najprv skúsili vytvoriť graf v GeoGebre. Otvorili sme si prvý graf v DeadLine a tam sme si pozreli predpis funkcie, ktorý sme následne vložili do GeoGebry. Následne sme si pozreli ako vyzerá daný graf v textovom súbore (Obr. ...). Je to oveľa čitateľnejšie než konštrukcia a o to jednoduchší program na to potrebujeme spraviť. Z týchto informácií nás bude zaujímať iba predpis funkcie, ktorý vyčítame z riadku, v ktorom sa nachádza "Name". V GeoGebre si vytvoríme šablónu (viac v 3.3), ktorá už bude obsahovať hľadanie priesečníkov, extrémov a deriváciu funkcie. Náš nový program otvorí a prečíta súbor, vyčíta predpis funkcie, otvorí GeoGebru, prihlási nás do nej a otvorí šablónu, ktorú sme vytvorili priamo pre grafy. Zmení predpis funkcie, šablónu pomenuje a uloží. A toto zopakuje pre všetky súbory s Grafmi. Keďže každý graf je iný, tak ešte potrebujeme urobiť nejaké vizuálne úpravy týchto grafov v GeoGebre.

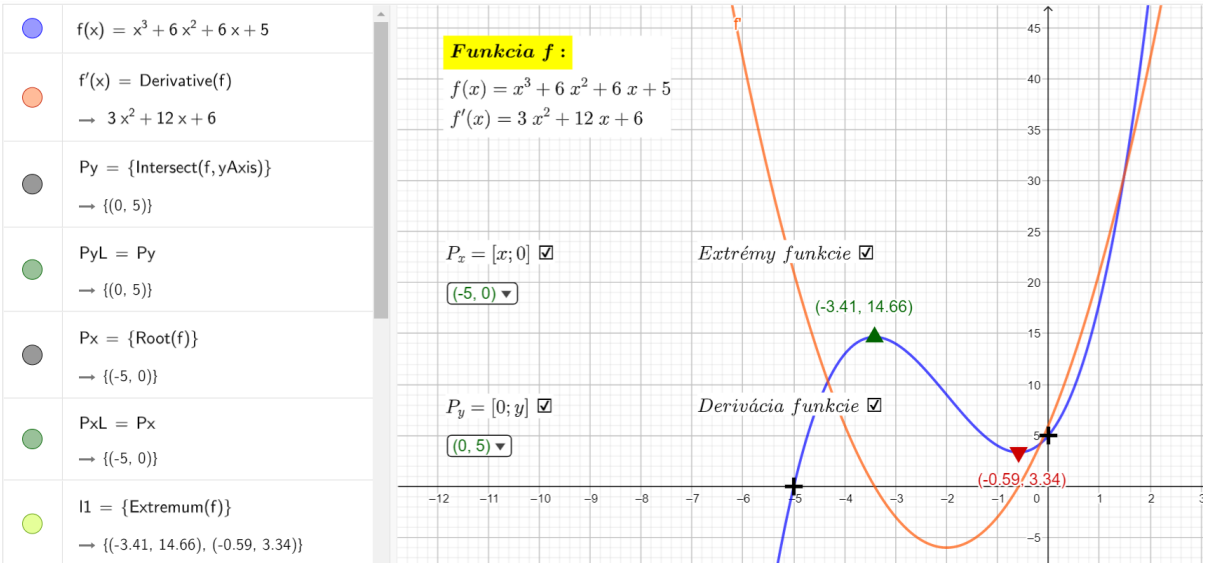
```
[DeadLine]
Version=2.26.962
[Function]
Name=x^3 + 6*x^2 + 6*x + 5
[Graph]
Title=x^3 + 6*x^2 + 6*x + 5=0
Xmin=-6
Xmax= 2
Ymin=-5
Ymax= 15
[Precision]
Graph points= 200
Root-finding steps= 1000
[Colors]
Axes= 0
Grid= 12632256
Graph= 255
[Show]
Grid= True
```

Obr. 15: Graf v DeadLine otvorený v textovom editore.

3.3 Šablóna grafu

Aby sme nemuseli zakaždým otvárať GeoGebru, prihlasovať sa do nej a vytvárať opakovane vyhľadávanie priesečníkov a podobne, tak sme si na to vytvorili šablónu, ktorú náš program otvorí a iba do nej vloží predpis funkcie. Na Obr. 19 vidíme ako vyzerá naša šablóna. Vľavo vidíme algebraické výrazy a vpravo vidíme grafické okno, ktoré uvidí používateľ. Najprv sme vložili predpis funkcie (v grafickom okne nám vznikol jej graf). Následne sme zistili deriváciu funkcie, priesečníky a extrémy funkcie. Všetky lokálne extrémy sme vložili do pola, z ktorého sme potom zisťovali najväčšie a najmenšie lokálne minimum a maximum, tie sme označili zvlášť. Keď sme mali už zistené všetky informácie, ktoré sme chceli pre daný graf vedieť, tak bolo na čase dostať ich do grafického okna. Na zobrazovanie sme použili checkboxy, ktoré sme prepojili s našimi prvkami a pomocou GeoGebrovských podmienok sme zaobstarali ako sa majú prvky v grafickej ploche správať pri zmene hodnoty v checkboxe. Niekedy máme viac priesečníkov, ktoré sa zobrazia na grafe ale takisto sa ich hodnoty v grafickom okne vypíšu pod checkboxami. Veľký červený trojuholník smerujúci nadol sme použili na označenie pre najmenšie (lokálne) minimum a veľký zelený trojuholník smerujúci nahor pre najväčšie (lokálne) maximum. Ostatné

lokálne extrémy sme označili zelenou guličkou. Priesečníky sme označili ako čierne plusy. Grafy sú modrou farbou a derivácie oranžovou.



Obr. 16: Šablóna grafu.

3.4 Tvorba tabuliek

V 2.2.4 sme si popísali rôzne možnosti, nad ktorými sme uvažovali pri vytváraní tabuliek. Ako zakomponujeme tabuľky do našej webstránky sme vyriešili nasledovne, excelovské súbory sme si rozdelili na tri druhy. Medzi prvý druh patria všetky súbory, ktoré sú neinteraktívne. Tieto sme vytvorili pomocou javascriptovej knižnice MathJax a pridali sme na stránku, nakoľko s ňou používateľ fyzicky nepracuje tak je zbytočné aby sme ich prerábali do GeoGebry. Tabuľky, ktoré neobsahujú príliš veľa buniek a sú interaktívne, sme prerobili do GeoGebry a následne ich vložili pomocou JavaScriptu do našej webovej stránky. Zvyšné tabuľky sme vložili ako odkaz na stiahnutie.

Tabuľky v GeoGebre sme vytvárali veľmi podobne ako sa vytvárajú tabuľky v Exceli. Pozreli sme si aký vzorec sa nachádza za bunkou v pôvodnej tabuľke a následne sme tento vzorec prepísali do vzorca v GeoGebre. Podmienky aj vytváranie vzorcov sa trochu líši. Napríklad v pôvodnej tabuľke v exceli sa v bunke skrýva '=KDYŽ(NEBO($A19 = 0; D$18 = 0); 0; CELÁ.ČÁST(($A19 + D$18) / B16) * 10 + MOD($A19 + D$18; B16))', v GeoGebre by takýto vzorec vyzeral nasledovne 'If($A19 == 0 —— D$18 == 0, 0, Div($A19 + D$18, B16) * 10 + Mod($A19 + D$18, B16)).' Najprv sme miesto

príkazu Div, používali príkaz Division, ktorý nám výsledok vyhodil v množine, kde prvé číslo bolo celá časť a druhé číslo bolo modulo a z tohto sme vyberali prvé číslo, ale neskôr sme v GeoGebra manuále našli funkciu Div, ktorá nám rovno vráti celú časť, ktorá nás zaujíma. Museli sme opäť riešiť komplikácie kedy niečo v GeoGebre nefungovalo tak ako v Exceli. Napríklad v Exceli sme mali vzorec '=KDYŽ(ŘÁDEK(A6) > ODMOCNINA($K4$); ; ŘÁDEK(A6))', ktorý ak riadok A6 je väčší než odmocnina z K4 tak je bunka prázdna inak tam vloží riadok A6. Poprípade sme mali aj také, že miesto prázdnej bunky vložíme prázdny reťazec. Takéto niečo sa v GeoGebre spraviť nedá, nevieme kombinovať reťazce a čísla v príkazoch / vzorcoch. Riešením bolo vložiť na takéto miesto číslo, ktoré vieme, že sa nikdy nebude v danej bunke výpočtom nachádzať miesto prázdnej hodnoty alebo prázdneho reťazca (väčšinou stačilo vložiť číslo 0 alebo ak sme len chceli schovať hodnotu, ktorú vieme, že tam určite bude ale pre určité vstupy ich nechceme zobrazovať tak sme zadali to číslo) a museli sme vypodmienkovať farbu. Čiže ak sa výpočet == 0, tak sme museli nastaviť pre R,G,B aby nám dokopy dali bielu farbu alebo naopak ak sa to rovná tomu čo chceme tak čiernu. Takže otvorili sme si nastavenia pre danú bunku, išli do kolonky 'advanced', kde sme nastavili pre R, G, B zvlášť takýto príkaz - If($A19 $\geq$ B16 $\vee$ D$18 $\geq$ B16, 1, 0).

Ďalšou komplikáciou, s ktorou sme sa pri vytváraní tabuliek stretli bolo nastavenie farby bunky. Väčšinou autor čítanky zmenil farbu bunky na 'cyan' vtedy, keď mal používateľ túto hodnotu meniť. Problém GeoGebry je ten, že ak máme v bunke číslo, tak sa farba bunky zmeniť nedá. Prišli sme ale na spôsob ako to obísť a to tak, že najprv sme do všetkých buniek, ktoré sme vedeli, že ich potrebujeme mať zafarbené vložili nejaký symbol alebo písmeno, v tomto prípade sa bunka dá zafarbiť a dá sa aj zmeniť hrúbka písma. Keď sme mali bunky zafarbené, tak sme ich obsah prepísali na čísla alebo vzorce, ktoré sme potrebovali.

Zo začiatku sme tabuľky pomocou funkcie create table vkladali do grafickej plochy GeoGebry a menenie hodnôt sme spravili tak, že sme si vytvorili input box, ktorý sme prepojili s číslom v tabuľke, ktorého hodnotu sme chceli meniť. Výzorovo to vyzeralo lepšie ale problém bol opäť v tom, že si používateľ nemohol pozrieť aký vzorec sa nachádza za bunkami, ktoré sa menili. A keď sme už potrebovali meniť viac čísiel, tak sme potrebovali aj viac input boxov a museli sme farebne odlišovať jednotlivé boxy a zafarbovať bunky

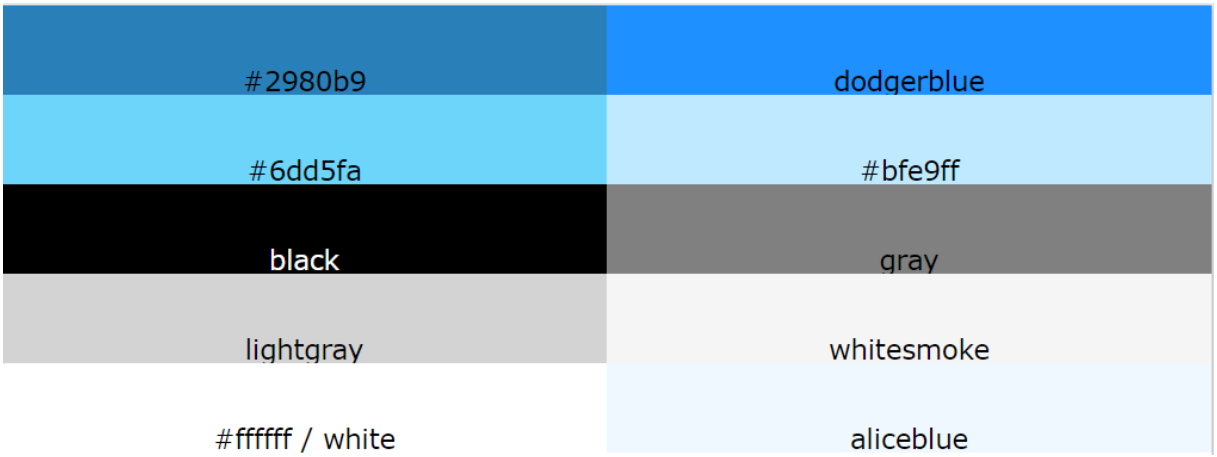
aby používateľ vedel, ktorý input box je prepojený s ktorou bunkou. Z týchto dôvodov sme teda tabuľky nechali len ako tabuľky a nevkladali sme ich do grafickej plochy.

3.5 Webová stránka

Text z wordových súborov sme prepísali do html. Všetky už neexistujúce hypertextové odkazy sme vymazali alebo nahradili podobnými. Na napísanie rovníc, matematických výrazov a tabuliek sme použili javascriptovú knižnicu MathJax. Ako prvú sme použili verziu 2, pri ktorej sme neskôr zistili, že v nej nevieme použiť príkazy na zafarbenie tabuliek (konkrétne sme potrebovali nastavovať cellcolor), ktoré v niektorých prípadoch potrebujeme, dalo by sa to samozrejme aj bez toho (pomocou príkazu 'bbox', ktorý nám vytvorí za textom akýsi box, ktorému môžeme zmeniť farbu) ale vizuálne to nevyzeralo dobre. Následne sme použili verziu 3, v ktorej síce zafarbenie tabuliek funguje ale nefunguje automatické zalamovanie textu, ktoré je ale na stránke MathJaxu uvedené, že jeho implementácia je medzi prioritami, takže neskôr môžeme toto zalamovanie do stránky pridať.

3.5.1 Grafický dizajn

Pri dizajne našej webstránky sme využili kombináciu modrej, čiernej a bielej farby v rôznych odtieňoch. Použité farby môžete vidieť na Obr. 12.



Obr. 17: Použité farby v CSS.

- Header a footer

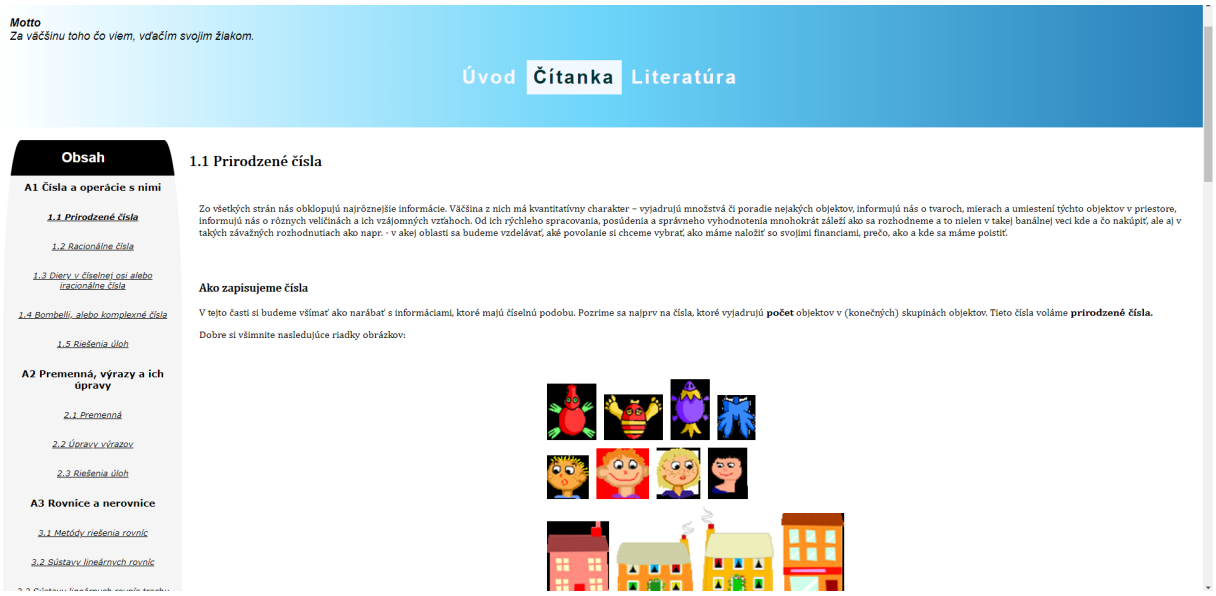
  Na zafarbenie pozadia hlavičky a pätičky sme využili lineárny gradient troch farieb, konkrétne '#2980b9' , '#6dd5fa' a '#ffffff'. Súčasťou hlavičky je aj hlavná navigácia pomocou, ktorej sa môže používateľ preklikávať medzi stránkami úvodu, samotnej čítanky a literatúry. Písmo elementu navigácie a (nav a), ktorý sa odkazuje na stránku, ktorá je aktuálne zobrazená, je čiernej farby a pozadie tohoto elementu má 'aliceblue' farbu. Písma zvyšných elementov navigácie sú bielej farby a pozadie farbu nemá. Ďalej sme nastavili element hover, ktorý pri prekrytí elementu nav a myšou zmení farby na také isté ako pri zobrazení aktuálnej stránky.

Obr. 18: Hlavička s navigáciou + ukážka podstránky Úvod.

- Podstránka 'Čítanka'

  V 2.2.1 sme si popísali čo sa nachádza v podstránke s názvom Čítanka. Prvú vec, ktorú používateľ uvidí, keď sa dostane na túto podstránku je obsah čítanky, ktorý sa nachádza v ľavej časti stránky a text čítanky (konkrétne kapitola A1 a podkapitola 1.1 prirodzené čísla), táto podkapitola je aj hrubo vyznačená v obsahu. Obsah a text čítanky je rozdelený pomocou elementu grid container, kde obsah tvorí 15% šírky stránky a čítanka 85%. Pozadie obsahu je zafarbené 'whitesmoke' farbou, ktorá obsah pekne oddelí od textu čítanky. Obsah čítanky sa odkazuje na ďalšie podstránky s rôznymi podkapitolami. Podľa toho, ktorú podkapitolu má používateľ zobrazenú,

tak je názov tejto podkapitoly hrubo vyznačený.



Obr. 19: Ukážka podstránky Čítanka s obsahom čítanky.

– Stránkovanie

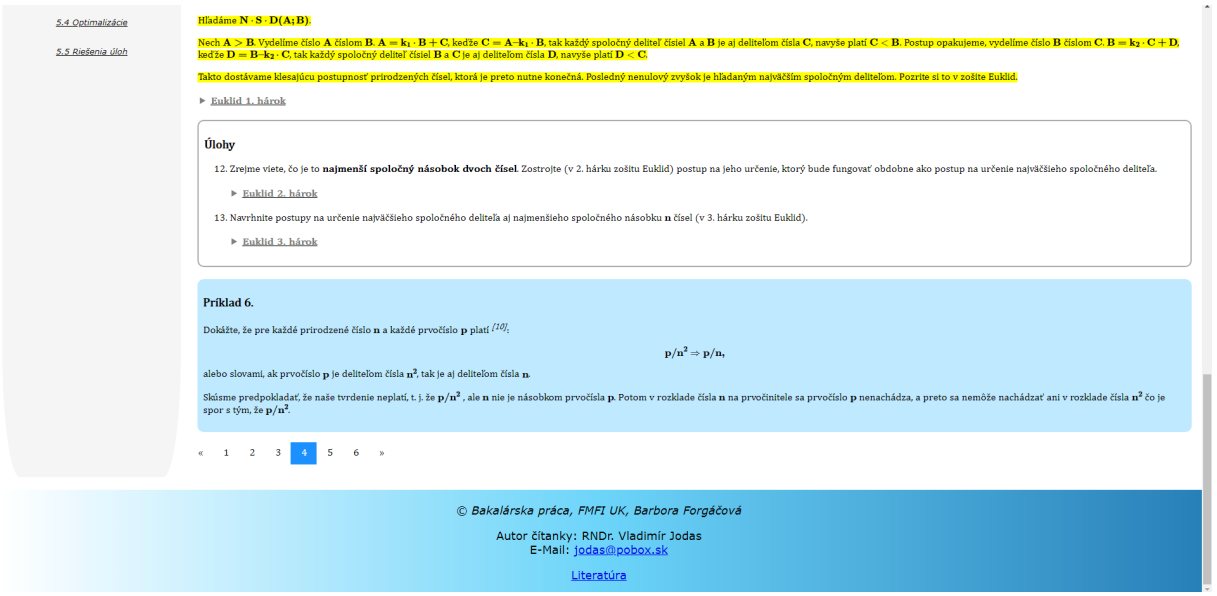
  Každá podkapitola je rozdelená na ďalšie podstránky (rozdelili sme ich podľa elementu section) a vytvorili sme stránkovanie, ktoré sa na dané podstránky odkazujú. Toto stránkovanie sa nachádza na spodku stránky (pred pätičkou). Číslo stránky, na ktorej sa používateľ momentálne nachádza je vyznačené farbou pozadia 'dodgerblue' a písmo je bielej farby, zatiaľ čo ostatné čísla stránok majú čiernu farbu a sú bez pozadia. Pomocou elementu hover meníme farbu pozadia na 'lightgray' pri prekrytí elementu myšou.

– Úlohy

  V čítanke sa môžeme stretnúť s rôznymi úlohami, ktoré môže používateľ riešiť. Chceli sme ich nejako oddeliť od bežného textu aby bolo na prvý pohľad jasné, že to nie je obyčajný text. Úlohy sme oddelili pomocou CSS, konkrétne sme pridali border s rádiusom 10px a šírkou 2px.

– Príklady

  Takisto ako s úlohami sa môžeme stretnúť s príkladmi, ktoré sme chceli tiež odlíšiť od bežného textu ale aj od úloh. Preto sme im zmenili farbu pozadia na '#bfe9ff' a nastavili sme border radius takisto na 10px.



Obr. 20: Ukážka úloh a príkladov s elementom details + pätička a stránkovanie.

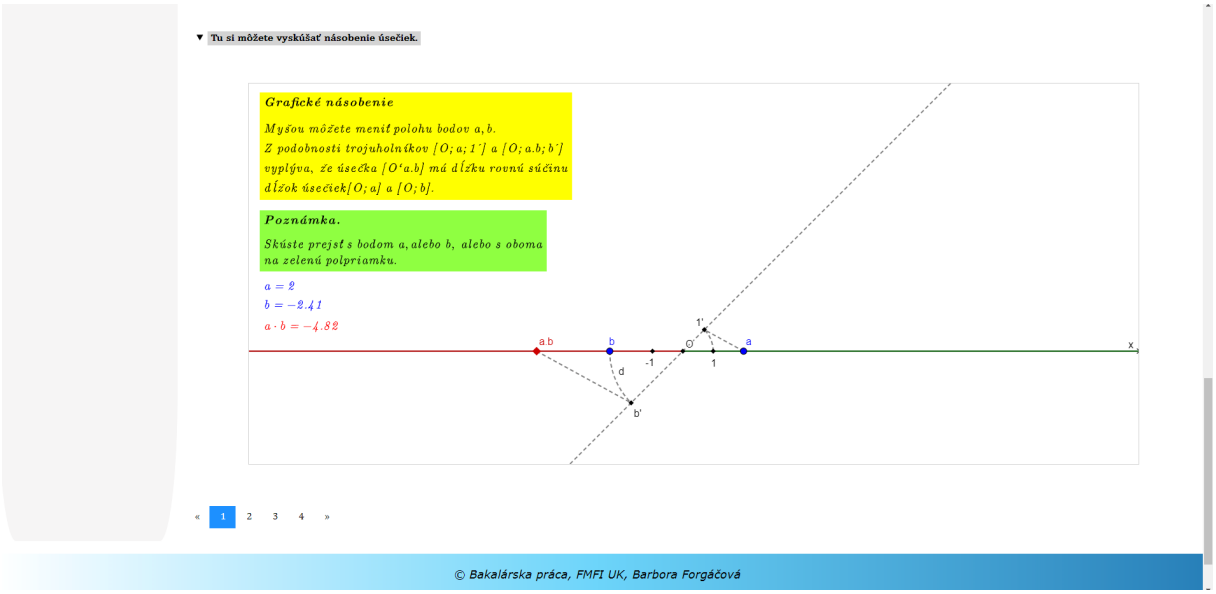
– Konštrukcie, grafy, tabuľky

Konštrukcie, grafy a tabuľky, ktoré sme prerábali a používateľ s nimi môže pracovať sme pridali do našej webovej stránky pomocou javascriptu. Tam, kde sme ich chceli mať pridané sme vytvorili element div, ktorému sme zadali id s kódom nášho GeoGebra appletu a pred koncom html sme vytvorili script, ktorý vyzeral nejak takto (uvedieme si príklad pridania iba dvoch appletov):

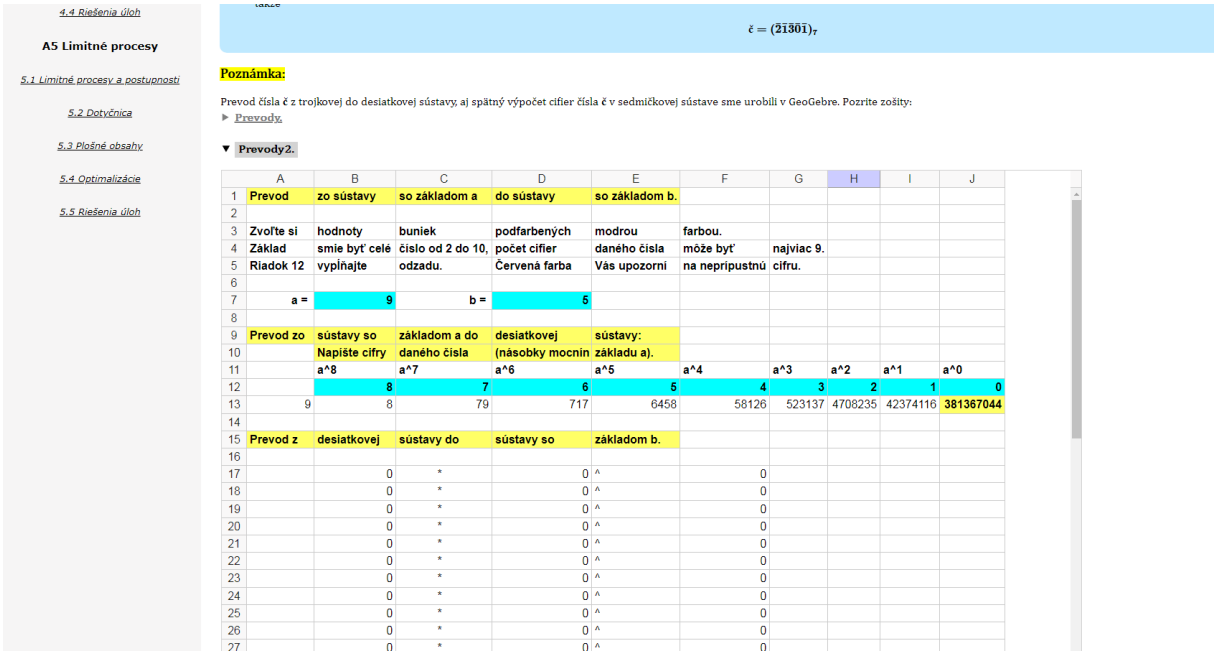
```
<script>
    parameters={"material_id":"qsmsw84a", "width":"1200"};
    var applet1 = new GGBApplet(parameters, '5.0', views);

    parameters={"material_id":"s5587rkn","width":"1100"};
    var applet2 = new GGBApplet(parameters, '5.0', views);

    window.onload = function() {
        applet1.inject('qsmsw84a');
        applet2.inject('s5587rkn');
};
</script>
```

Najprv sme zmenili defaultné parametre, aké sme chceli aby mal náš applet na stránke, tých parametrov je veľmi veľa ale my sme chceli zmeniť iba id materiálu, šírku a pri niektorých aj výšku. Tieto parametre sme pridali do nášho appletu, ktorý sme následne načítali na našu webstránku.

Čo sa týka grafickej stránky, tak sme ich schovali za element details pomocou, ktorého si ich môže používateľ zobrazovať a schovávať. V stave kedy nie sú zobrazené je písmo zafarbené na 'gray' a podčiarknuté. Akonáhle ich používateľ otvorí tak sa farba písma zmení na čiernu, podčiarknutie zmizne a nastaví sa farba pozadia na 'lightgray'.



Obr. 21: Ukážka konštrukcie vo webstránke.



Obr. 22: Ukážka tabuľky vo webstránke.

Záver

V našej práci sme prerobli existujúcu matematickú čítanku do podoby webovej stránky, tak aby bola intuitívna a interaktívna.

V prvej kapitole sme si popísali ako vyzerala pôvodná matematická čítanka, ukázali sme podobné existujúce riešenie, konkrétne hru Euclid, ktorá nás inšpirovala pri riešení a popísali sme si Jupyter Notebook a GeoGebru, ich výhody a nevýhody, nad ktorými sme sa zamýšľali čo z toho použiť pri riešení našej webovej stránky. V druhej kapitole sme si navrhli a popísali ako by mala naša stránka vyzerať, popísali sme aké technológie sme využili. Wordové súbory sme prerobili do html, matematické vzorce a rovnice sme zapisovali pomocou javascriptovej knižnice MathJax, matematické konštrukcie a grafy sme prerobili do softvéru GeoGebra a tabuľky sme si rozdelili na niekoľko častí, ktoré sme následne prerobili buď do GeoGebry alebo MathJaxu. Popísali sme si aj rôzne možnosti prerábania tabuliek za pomoci publisheetu, GeoGebry, javaScriptovou knižnicou SheetJS, ich výhody a nevýhody. V poslednej kapitole sme si popísali ako sme postupovali pri riešení prerábania materiálov. Ako sme analyzovali pôvodné materiály a postupne ich prerábali. Dokázali sme zautomatizovať prerábanie konštrukcií a grafov do GeoGebry v pythone pomocou knižnice Selenium.

Veríme, že sa nám podarilo vytvoriť lepšiu verziu pôvodnej čítanky a že zaujme moderných čitateľov.

Zoznam použitej literatúry

[1] BROŽA, P. 1999. *Tvorba WWW stránek pro úplné začátečníky.* Brno: Computer Press 1999. 128 s. ISBN 80-7226-164-9.

[2] BAINVILLE, E. 2002. *Cabri Geometrie II Plus: Příručka pro uživatele.* [online]. 27 s. [cit. 2021.01.20.] Dostupné na internete: <http://www.pf.jcu.cz/cabri/temata/vrba/manual_CabriPlus.pdf>.

[3] *Cabri.* [online]. Dostupné na internete: <http://www.cabri.net/cabri2/accueil-e.php>.

[4] 2018. *DeadLine.* [online]. Dostupné na internete: <http://www.canadiancontent.net/tech/download/DeadLine.html>.

[5] Jupyter Team. 2015. *The Jupyter Notebook.* [online]. Dostupné na internete: <https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>.

[6] SJURSEN, S. 2020. *The Pros and Cons of Using Jupyter Notebooks as Your Editor for Data Science Work.* [online]. Dostupné na internete: <https://medium.com/better-programming/pros-and-cons-for-jupyter-notebooks-as-your-editor-for-data-science\-work-tip-pycharm-is-probably-40e88f7827cbl>.

[7] *Co je GeoGebra?.* [online]. Dostupné na internete: <https://www.geogebra.org/about>.

[8] SVARDA, L. 2018. *GeoGebra vo vyučovaní odborných predmetov na stredných školách.* [online]. Dostupné na internete: <https://www.dps-az.cz/cad-cam-cae/id:57083/geogebra-vo-vyucovani-odbornych-predmetov-na-strednych-skolach>.

[9] WHATWG. 2021. *HTML.* [online]. Dostupné na internete: <https://html.spec.whatwg.org/#is-this-html5?>.

[10] The PHP Group. 2001-2021. *PHP.* [online]. Dostupné na internete: <https://www.php.net/>.

[11] HRUŠECKÝ R. *Prednáška z webových aplikácii*, FMFI UK Bratislava.

[12] KANTOR, I. 2007-2021. *An Introduction to JavaScript*. [online]. Dostupné na internete: <`https://javascript.info/intro`>.

[13] Mozilla and individual contributors. 2005-2021. *What is JavaScript?*. [online]. Dostupné na internete: <`https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript`>.

Príloha

https://davinci.fmph.uniba.sk/ forgacova37/citanka/