Univerzita Palackého v Olomouci
Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

2023 **Kryštof Parák**

Univerzita Palackého v Olomouci

Pedagogická fakulta

Katedra matematiky

Bakalářská práce

Využití programu wxMaxima ve výuce matematiky na 2. stupni ZŠ

Kryštof Parák

Olomouc 2023

Vedoucí práce: Mgr. Tomášek Talášek, Ph.D.

Prohlašuji, že jsem bakalářskou práci na téma „Využití programu wxMaxima ve výuce matematiky na 2. stupni ZŠ“, vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal, uvádím v seznamu použité literatury a zdrojů.

V Olomouci dne 20. června 2023 .....

Kryštof Parák

Poděkování

Rád bych touto cestou vyjádřil poděkování vedoucímu bakalářské práce Mgr. Tomášek Talášek, Ph.D. za jeho cenné rady, motivaci a trpělivost při vedení mé bakalářské práce.

OBSAH

ÚVOD

V dnešní době dochází k neustálému rozvoji nových technologií a ty mění náš způsob života. O významu matematiky, jako předmětu, který je podstatný pro další technologický vývoj není potřeba pochybovat. Změny ve vývoji a použití informační technologií se netýkají pouze volnočasových aktivit, ale taky změny ve vzdělávání. Již dnes se technologie využívají při výuce na školách a na všech stupních vzdělání. Dochází ke stanovení požadavků na digitální kompetence ve většině předmětů. V rámci matematiky můžeme například využít výpočetní programy, jako je program wxMaxima.

Cílem této bakalářské práce je se zaměřit na uplatnění programu wxMaxima ve výuce matematiky na druhém stupni základních škol v České republice. V práci se budeme snažit nastínit jakým způsobem program při výuce použít. V práci si ukážeme možný přínos využití programu učitelem nebo žáky ve výuce. Práce může taky posloužit, jako studijní materiál nebo manuál při práci s programem.

Pro úspěšné splnění cíle naší bakalářské práce je potřeba ji rozdělit do třech kapitol. V první kapitole se zaměříme na vzdělávání matematiky z pohledu didaktiky, rámcového vzdělávacího programu a možnosti procesu vzdělávání žáků s využitím počítače. Kapitola předá čtenáři informace o správné didaktice a požadavků z pohledu rámcového vzdělávacího programu a v poslední podkapitole si představíme způsob, jakým mohou být technologie využívány ve výuce. Dále si zde představíme rizika, ale i výhody využití technologií. Ukážeme si podmínky správného řízení vyučování při využití technologií.

Ve druhé kapitole bude představen program wxMaxima jako takový. Ukážeme si zde historii programu, instalaci, a hlavně instrukce k ovládání programu. Čtenář zde získá základní informace o využití programu.

V závěrečné kapitole se zaměříme na praktické využití programu wxMaxima. K ukázce praktického využití použijeme učebnice matematiky pro druhý stupeň základních škol od Jany Coufalové a kolektivu[1]. Postupně tedy budeme procházet učebnice v 6–9. ročníku a

[1] COUFALOVÁ, Jana. *Matematika pro 6. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 19.04.2023]. ISBN 978-80-7168-992-8.

COUFALOVÁ, Jana. *Matematika pro 7. ročník základní školy.* 3. vydání. Praha: Fortuna, 2017. [cit. 20.04.2023]. ISBN 978-80-7373-141-0.

COUFALOVÁ, Jana. *Matematika pro 8. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-994-2.

přestavíme si teorii a způsob vypočtu v učebnici, zároveň použijeme program wxMaxima k ukázce výpočtu pomocí programu. Na jednotlivých příkladech si vysvětlíme, jakým způsobem program a kód v programu pracuje.

Při tvorbě práce jsme využili volně stažitelný program wxMaxima a stejně dostupné teoretické opory pro tento program. Při úpravě a navrhování kódu jsme využili chatbota od společnosti OpenAI, který hlavně pomohl jako nápověda. Chatbot byl většinu času užitečný, ale došlo k situacím, kdy nedokázal navrhnout přínosnou nápovědu.

COUFALOVÁ, Jana. *Matematika pro 9. ročník základní školy.* 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-995-9.

1 PROCES VZDĚLÁVÁNÍ MATEMATIKY

Kapitola nám vytváří teoretický úvod do naší bakalářské práce. Zabýváme se vzděláváním matematiky a vzděláváním matematiky s využitím informačních a komunikačních technologií. Zaměřujeme se na didaktiku matematiky neboli na proces vyučování matematiky. Dále se zde zaměřujeme na rámcový vzdělávací program ve vzdělávací oblasti matematika a její aplikace na druhém stupni základních škol. Rozebíráme zde vzdělávání s využitím informačních technologií. Popisujeme odlišné způsoby vzdělávání s využitím informačních technologií, tak abychom dosáhli účinného a bezpečného vzdělávání.

1.1 Didaktika Matematiky

Didaktika matematiky je vědecká disciplína zaměřující se na procesy vyučování matematiky na různých úrovních od mateřských školek po vysoké školy. Všeobecně řeší detaily vyučování matematiky a které znalosti, jak do hloubky a jakým způsobem bychom měli matematiku vyučovat. Zajišťuje kontinuitu učiva tak, aby na sebe učivo navazovalo, aby učivo bylo vyučováno v určitém věku a náročnost učiva byla přizpůsobena věku a předchozím znalostem žáka. Stanovuje poměr mezi teoretickou a aplikační částí. Proces vzdělání musí vždy plnit i výchovné zaměření.[2]

Didaktika matematiky nemá tedy pouze vzdělávat v daném předmětu, ale musí rozvíjet i vlastnosti osobnosti. To matematika jako všechny předměty dělá a učí žáky pracovitosti, přesnosti, samostatnosti, ale taky schopnosti práce ve skupině. Cílem didaktiky je vychovat všestranně rozvinuté osobnosti.[3] V tom nám pomáhají i průřezová témata.

Didaktika matematiky, jako všechny didaktiky vychází z didaktických pravidel formulované Janem Amosem Komenským. Zmíníme si zde některá didaktická pravidla, která využíváme v didaktice matematiky. Postupování od snadného k obtížnějšímu na to navazující od známého k neznámému a od jednoduchého ke složitějšímu. Další důležitým pravidlem je pravidlo systematičnosti a soustavnosti, kdy to pravidlo platí nejenom pro jednotlivé předměty, ale taky mezi předměty a mezi stupni vzdělávání.[4]

[2] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky*. Olomouc: Univerzita Palackého, 2002. ISBN 80-244-0534-2.
[3] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky*. Olomouc: Univerzita Palackého, 2002. ISBN 80-244-0534-2.
[4] WOSSALA, Jan. *Využití ICT ve výuce matematiky*. [online]. [cit. 23.04.2023]. Dostupné z: https://www.kr-kralovehradecky.cz/assets/rozvoj-kraje/evropska-unie-EHP/krajsky-akcni-plan-rozvoje-vzdelavani/aktualni-informace/vyuziti_ict_ve_vyuce_matematiky.pdf

Když se podíváme na vzdělávání matematiky na druhém stupni, můžeme zde vidět výraznější vztahy mezi jednotlivými předměty. Matematika má zde mezipředmětové vazby s předměty přírodovědeckého nebo technického charakteru jako fyzika, chemie, zeměpis a další.

1.2 Rámcový vzdělávací program

Rámcový vzdělávací program nám pomáhá upřesnit si organizaci vzdělávání v rámci vzdělávacího systému. Stanovuje cíle, obsah a strukturu. Vzdělávací rámec je realizován ve školských zařízeních. V dokumentu jsou informace o cílech, způsobu hodnocení a dalších požadavcích v předmětech, které jsou vyučovány.

Rámcový vzdělávací program definuje vzdělávací oblast matematiky a její aplikaci v základním vzdělávání především na aktivních činnostech. Definuje oblast znalostí a schopností, které jsou nutné v praktickém životě. Tato vzdělávací oblast je nepostradatelným prvkem ve vzdělávání, a proto ji najdeme v celém základním vzdělávání.[5]

Na druhém stupni základních škol se oblast vzdělávání matematiky a její aplikace člení na čtyři tematické okruhy, které navazují a dále rozvijí oblast vzdělávání matematiky a aplikace z prvního stupně.

Prvním tematickým okruhem, který si zde zmíníme a najdeme ho na druhém stupni je okruh „Číslo a proměnná“. Z okruhu by si žáci měli osvojit aritmetické operace, algoritmické porozumění a významové porozumění. Umět získávat číselné údaje a pracovat s proměnnými.

Druhý tematický okruh je „Závislost, vztahy a práce s daty“. V tomhle okruhu by si žáci měli osvojit schopnost uvědomovat si závislosti jednotlivých jevů, co znamená růst, pokles nebo když se jev nemění. Měli by být schopni zanalyzovat grafy, tabulky a diagramy. K analýze nebo vhodnému vyjádření mohou využít počítačové softwary nebo grafické kalkulátory.

V tematickém okruhu „Geometrie v rovině a prostoru“ by se měli žáci naučit určit a znázornit geometrické útvary v rovině nebo prostoru. Porovnávat, měřit, spočítat obsah nebo obvod a celkově by měl mít žák schopnost pracovat s tvarem a prostorem.

[5] 5.2 Vzdělávací oblast - *Matematika a její aplikace* - úvod - DIGIFOLIO. Domů - DIGIFOLIO [online]. [cit. 22.04.2023]. Dostupné z: https://digifolio.rvp.cz/view/view.php?id=10289

Posledním tematickým okruhem je „Nestandartní aplikační úlohy a problémy“, který se od ostatních okruhu mírně liší. Úlohy a problémy nemusí souviset se znalostmi a dovednostmi školské matematiky, ale jsou to úlohy, při nichž je důležité použít logické myšlení. Použití logického myšlení prostupuje všemi okruhy. Zde se uplatnění promítá ve všech situacích ve škole i mimo školu. Je to schopnost řešit problémy.

Ve všech okruzích by kromě schopnosti logicky přemýšlet měla být rozvíjena schopnost práce s prostředky výpočetní techniky. Prostředky výpočetní techniky umožňují žákům kompenzovat možné slabé stránky jako nízká schopnost numerických výpočtů a podobně. Naopak rozvijí schopnosti jako samostatnost, práce se zdroji a schopnost se učit sám.[4]

1.3 Vzdělávání s využitím počítače

Vzdělávání matematiky s využitím infomačních komunikačních technologií (dále jen ICT) má své určité specifikace. Pro pedagogy a celou společnost to znamená výzvu, ale i příležitost. Je potřeba se podívat na vzdělávání matematiky s využitím ICT nebo vzdělávání s využitím počítače celkově z různých pohledů.

Maňák a Švec[6] řadí výuku s podporou počítače mezi komplexní metody. Komplexní metodu bychom mohli definovat jako metodu, kde jde o složité metodické útvary a má různou, ale ucelenou kombinaci a propojenost základních prvků didaktického systému.

Využití počítačů ve škole může být všestranné. Slouží jako informační systém školy, zdroj informací pro pedagogy i žáky, poskytuje programy pro zpracování textu, grafické editory, tabulkové kalkulátory a v našem případě i výpočetní programy. Počítač je komplexní zařízení a při správném zvoleném výukovém programu je schopen na žáka působit zvukově, obrazově, textově a graficky. Všeobecně ale slouží jako počítačová podpora ve výuce.

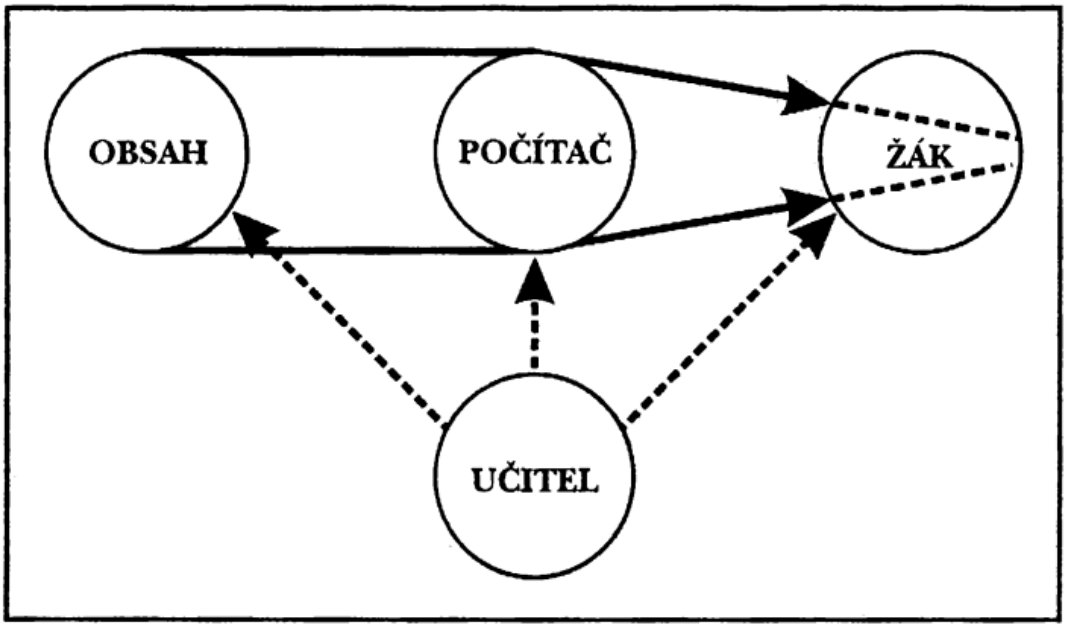
Při využití výukových programů žák postupuje samostatně nebo pod dohledem učitele (Obrázek 1). Je to způsob, který je podobný jiným modelům výuky. Učitel by měl žáka vést při seznámení s výukovými programy. Učitelova podpora žákovi by se postupem času měla snižovat až se žák nakonec stane plně samostatným. Počítat s v hodným výukovým programem dokáže tenhle proces velice individualizovat, jelikož se přizpůsobuje aktuálnímu stavu žáka. Díky schopnosti výukového programu přizpůsobovat se aktuální znalosti učení[7]

[6] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.

[7] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.

žáka, je možné dosáhnout ideální situace, kdy průběh vzdělávání řídí vzdělávající se a ne pedagog.

Využití výukových programu ve výuce může být různorodé. Může být individuální, kdy každý žák pracuje s počítačem samostatně a dále může být použit ve skupinové výuce nebo při demonstraci ukázek pro celou třídu.[8]



Obrázek 1 Schéma výuky pomocí počítače[9]

V dnešní době je stále důležitější, aby se žáci i společnost naučili orientovat v přemíře informací, aby se naučili vybírat údaje, které jsou relevantní. Rozvoj ICT a využití počítačů ve výuce tento problém více prohlubuje. Řešením je naučit žáky informace třídit, kategorizovat a hodnotit je podle důležitosti. Již dnes je tahle dovednost pro učitele potřebná a její význam bude nadále růst. Rozvoj ICT umožňuje žákům využít kvalitní materiály, které jsou dostupné celosvětově.[10]

Důvodů pro využití počítače nebo ICT ve výuce je mnoho. Neustálý rozvoj nových ICT a jejich předpokládaný rozvoj v budoucnosti. S tím související schopnost využívat technologie v praxi. Kladný vztah dětí k počítačům a technologiím celkově. Můžeme využít toho kladného vztahu v náš prospěch a například díky počítačům vytvořit kladný vztah k učení nebo k určitému předmětu. Práce s počítačem nebo technologiemi rozvijí logické myšlení.

[8] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.
[9] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.
[10] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.

Vhodný vzdělávací program umožňuje kvalitní procvičování znalostí a dále nám umožňuje ověření správného výsledku.

Žáci si mohou příklad vypočítat na papír a pomocí programu si zkontrolovat správnost svého výsledku. Dalším důvodem je rozvoj počítačové gramotnosti. Tato schopnost je už v dnešním světě samozřejmým požadavkem. [11]

Při vzděláváním s využitím počítače bychom měli zmínit některé negativní aspekty této metody. Může dojít k dopadu na zdraví žáků z důvodu špatného usazení a celkového nastavení počítače a prostředí počítače. Žáci již teď tráví mnoho času s upřeným zrakem na obrazovky a další čas, kdy by žáci trávili díváním se na obrazovky, by jim mohl poškodit zrak. Proto je potřeba žáky poučit o bezpečném trávení času před obrazovkami. Negativní stránku by mohlo mít zkoušení přes počítač, kdy zkoušející by nemohl vidět postup nebo rozsáhlou odpověď, ale pouze odpověď jako v testu. Rozšíření využití ICT ve vyučování by nemělo způsobit nahrazení mezilidských vztahů a oslabování sociálních vazeb, ať už mezi žákem a učitelem nebo mezi žáky.

S využitím počítačů ve vzdělávání dochází i ke změně role učitele ve výuce. Učitel se stává organizátorem a manažerem vyučovacího procesu, také didaktickým programátorem, technologem vyučovacích prostředků a výzkumníkem v oboru didaktiky, a hlavně partnerem žáka. Žáci nejsou pouze objektem, ale partnerem ve vzdělávání.[12]

Pro správné a efektivní vyučování s počítačem nebo jiného druhu ICT musí být použity vhodné nástroje pro správu třídy. Je potřeba, aby bylo možné plnit různé funkce k lepšímu řízení. K nim patří monitoring aktivit na jednotlivých zařízeních, omezení přístupu na internet, uzamčení či zablokování promítání dění na ploše určitého zařízení. Všechny tyto funkce a mnohé další pomáhají vyučujícímu vést kvalitní výuku s využitím počítačů nebo jiných ICT.[13]

Řada učitelů, žáků nebo rodičů bere počítač jen jako technologii na hraní počítačových her. Bohužel řada z nich si neuvědomuje potenciál využití počítače nebo celkově infomační komunikačních technologií ve výuce.[14]

---

[11] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky 2*. Olomouc: Univerzita Palackého v Olomouci, 2004. Skripta / Univerzita Palackého. Pedagogická fakulta. [cit. 22.04.2023]. ISBN 80-244-0815-5.

[12] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.

[13] WOSSALA, Jan. *Využití ICT ve výuce matematiky*. [online]. [cit. 23.04.2023]. Dostupné z: https://www.kr-kralovehradecky.cz/assets/rozvoj-kraje/evropska-unie-EHP/krajsky-akcni-plan-rozvoje-vzdelavani/aktualni-informace/vyuziti_ict_ve_vyuce_matematiky.pdf

[14] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky 2*. Olomouc: Univerzita Palackého v Olomouci, 2004. Skripta / Univerzita Palackého. Pedagogická fakulta. [cit. 22.04.2023]. ISBN 80-244-0815-5.

Studenti, kteří se ještě připravují na svoji karieru pedagoga i pedagogové, kteří již učí a je jedno, jak dlouho, by se měli seznámit s příležitostí začlenění informačních technologií ve výuce. Do přípravy na kariéru pedagoga nebo v průběžném dovzdělávání pedagogů by mělo být zařazeno seznámení se s počítači včetně programování, alespoň do určité úrovně.[15]

[15] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky 2*. Olomouc: Univerzita Palackého v Olomouci, 2004. Skripta / Univerzita Palackého. Pedagogická fakulta. [cit. 22.04.2023]. ISBN 80-244-0815-5.

2 ÚVOD DO PROGRAMU WXMAXIMA

První kapitola se věnuje obecnému představení programu wxMaxima, který budeme v bakalářské práci studovat a používat. V první kapitole si rozebereme historii, instalaci a uživatelské prostředí programu. Dále si ukážeme základní ovládání programu. Program wxMaxima je kompatibilní s operačními systémy Windows, macOS a Linux. My budeme využívat wxMaximu na operační sytém Windows, a to i z důvodu, že se jedná o nejrozšířenější počítačový operační systém. Využití operačního systému neomezuje použitelnost práce pouze na vybraný systém. Program wxMaxima je grafickou nástavbou programu Maxima.

2.1 Historie programu

Program Maxima původně vznikl z programu Macsymy. Počítačový algebraický program Macsymy byl vyvinutý koncem 60. let na Massachusettském technologickém institutu (MIT). Jednalo se o veřejně dostupný program, kterým se inspirovaly další programy, jako například Maple, Mathematica.[16]

Program Maxima byl od roku 1982 do roku 2001 spravován profesorem William Schelterem. V roce 1998 uvolnil zdrojové kódy programu a díky jeho snaze bylo umožněno zachování a následující rozvoj programu. Po jeho smrti se objevila řada uživatelů a vývojářů z komunity okolo programu Maximy, kteří pracovali na jeho vývoji nebo zpopularizování širšímu publiku. Díky tomu se pro program Maxima objevují nové aktualizace a dokumenty.[17]

Program wxMaxima je grafickou nástavbou programu Maxima. Program wxMaxima je stejně jako program Maxima volně dostupný program, který je schopen provádět numerické a symbolické výpočty, dále umožňuje grafické zobrazení ve 2D nebo 3D. Program wxMaxima spadá mezi systémy počítačové algebry a má částečnou lokalizaci v češtině.[18]

Protože je program wxMaxima grafickou nástavbou neboli graphical user interface (GUI) programu Maxima, umožňuje využívání funkcí programu Maxima. Maxima běží

[16] BESHENOV, Lyosha. *Maxima a Computer Algebra System*.[online]. [cit. 21.04.2023]. Dostupné z: http://maxima.sourceforge.net

[17] BUŠA, Ján. MAXIMA: *Open source systém počítačovej algebry*. [online]. [cit. 21.04.2023]. 2006. Dostupné z: http://sccg.sk/~batorova/UPG/priruckaSK.pdf

[18] HAŠEK, Roman, NORULÁKOVÁ, Michaela. *Program wxMaxima ve výuce matematiky* [online]. Jihočeská Univerzita v Českých Budějovicích, [cit. 7.04.2023]. Dostupné z: http://home.pf.jcu.cz/~hasek/VTM1/wxMaxima_ve_vyuce.pdf

v terminálu neboli v textově uživatelském rozhraní. Program wxMaxima může být díky svému přívětivému grafickému rozhraní upřednostňován před programem Maxima.[19]

2.2 Instalace programu

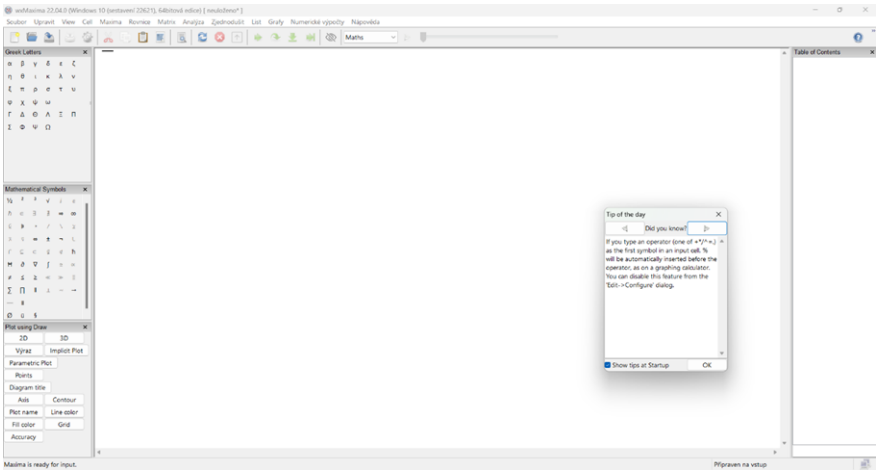
Jak již bylo zmíněno program wxMaxima je volně dostupný program. Jeho stáhnutí je možné z těchto stránek (https://sourceforge.net/projects/maxima/files/Maxima-Windows/). Po stáhnutí instalačního souboru, rozklikneme a instalační proces nás již intuitivně provede až k samotnému nainstalování programu. Během instalace se nás program ještě dotáže, které nadstavby programu chceme, aby byly součástí naší instalace. Zvolíme, že chceme všechny nadstavby programu k instalaci.

2.3 Seznámení s programem

Při spuštění programu se zobrazí hlavní okno s malým vyskakovacím oknem v něm (Obrázek 2). Při pohledu na program si můžeme všimnout mnoha funkcí, které nám budou sloužit k plnohodnotnému využívání programu. Program má částečnou lokalizaci, takže funkce jsou napsány buď česky nebo anglicky.

Předem zmíněné malé vyskakovací okno nám zobrazuje tipy dne. Tipy dne fungují jako základní návod k ovládání programu. Tipů dne je více a jdou mezi sebou proklikávat. Jeden z tipů obsahuje odkaz na internetovou stránku, kde najdeme podrobnější manuál zaměřený na ovládání. Následně jde celé okno zrušit, aby nám příště při startu aplikace nevyskakovalo.

[19] wxMaxima Developers. *wxMaxima* [online]. [cit. 15.03.2023]. Dostupné z: https://wxMaxima-developers.github.io/wxMaxima/wxMaxima.pdf



Obrázek 2 Hlavní okno programu wxMaxima

V hlavním okně na horní liště můžeme najít jednotlivé funkce. Funkce *Soubor*, pod které spadají další funkce jako *Nový Dokument*, *Otevřít dokument*, *Uložit dokument*, *Export*, *Tisk* a *ukončení programu*. Export může být ve formátu .html, .tex, .mac. Funkce *Upravit* nám umožňuje ve své podnabídce jít kroky zpět, vpřed, vložit či kopírovat.

Funkce *Náhled* nám umožňuje si grafické prostředí programu dále přizpůsobit. Můžeme si zatrhnout, které funkce jsou pro nás důležité a chceme je mít hned v hlavním okně, například řecká písmena, matematické symboly, funkce k vykreslení grafů, funkce k výpočtům a historii programu.

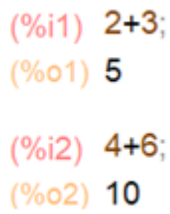
Dále si zde musíme zmínit nabídku funkce *Maxima*, která nabízí restart či přerušení programu. Funkce *Rovnice*, *Analýza*, *Zjednodušit Matice*, *Grafy* nebo *Numerické výpočty* nám umožňují použití funkcí u jednotlivých matematických oborů.

Užitečné funkce jsou taky *Nastavení* a *Nápověda*, která funguje i v off-line verzi, bohužel je nápověda v anglickém jazyce, takže to může být drobná komplikace.

V nastavení si můžeme program dále přizpůsobit svému obrazu, jako styl písma apod. Nicméně nejdůležitější pro nás bude možnost přenastavit funkci na vyhodnocení ze zkratky *Shift + Enter* pouze na funkci *Enter* a tím nám práci v programu nadále ulehčit a zkratka *Shift + Enter*, bude mít nový význam, a to přechod na nový řádek. K vyhodnocování taky využíváme funkce (*Vyhodnotit vstupní pole, Evaulate all cells*..), které najdeme v nabídce *Cell*.

2.4 Zadání a vypsání příkazů

Při představení programu je potřeba si ukázat, jaké jsou podmínky pro zadání a vypsání příkazů v programu wxMaxima. Vstupy v programu bývají označovány *(%ix)*, kde *x* bývá číslo vstupního příkazu. Proměnná *x* se zvyšuje s počtem vstupních příkazů. Výstupy se označují stylem *(%ox)*, kde *x* bývá číslo výstupu. Ke každému jednomu vstupu bývá přiřazen právě jeden výstup. Například při zadání vstupu *(%i1)* bude výstup označen *(%o1)*. Stejné číslo nacházející se ve vstupním a výstupním příkazu nám dokazuje, že právě tenhle vstupní příkaz patří k tomuhle výstupu (Obrázek 3). Písmeno *i* při zadání vstupu pochází z anglického slova input (vstup) obdobně *o* při vypsání výstupu pochází z anglického slova output (výstup). Středník na konci řádku, slouží k označení konce příkazu / výrazu. Dáváme tím najevo, že zadaný příkaz je kompletní a může dojít k jeho vyhodnocení.



Obrázek 3 Zadání vstupu a vypsání výstupu v programu

2.5 Základní operace

Pro náš vstup do programu je potřeba se seznámit s početními operacemi, které budeme používat (Obrázek 4). Nejdříve si představíme operace jako sčítání, odčítání, podíl a násobení. Operaci sčítání a operaci odčítání zapisujeme stejně jako ve většině programů anebo do sešitu a to pomocí + a -. Násobení zapíšeme jinak, než jak jsme zvyklí zapisovat v sešitu, kdo má ale zkušenosti například s programem Excel, již ví, jak operaci zapsat. Násobení zapíšeme pomocí klávesového znaku hvězdičky *. Program nám zapsaný znak * zapíše jako běžný znak pro násobení. Musíme si ale dát pozor, protože nemůžeme napsat *3x*, ale musíme napsat *3*x*. Nemůžeme tedy využít zkrácený zápis. Dělení zapíšeme pomocí znaku lomítka /. Jestli bychom chtěli zapsat zlomek, provedeme to pomocí operace dělení a tím pádem bychom opět použili lomítko /. V případě, že chceme zlomek převést na desetinné číslo, použijeme funkci *float*. Pokud tedy chceme zapsat zlomek 9/5 v desetinném čísle napíšeme *float(9/5)*. Program totiž nechává výstup ve zlomku. Jestli bychom chtěli potlačit provedení jakéhokoliv příkazu neboli chtěli bychom z nějakého důvodu nevypsat určitý řádek, umístíme za daný příkaz $. Nevypsání řádku můžeme využít při výpočtu mezivýsledku, který není významný. Klávesová zkratka pro Windows je AltGr + ů.

```
(%i1) 2+3;
(%o1) 5

(%i2) 6-4;
(%o2) 2

(%i3) 2·5;
(%o3) 10

(%i4) 6/3;
(%o4) 2

(%i5) 9/5;
(%o5) 9/5

(%i8) float(9/5);
(%o8) 1.8

(%i9) 2+3$
```

Obrázek 4 Ukázka základních operací v programu

Mezi další operace, které bychom měli ještě znát můžeme zařadit mocniny a odmocniny. V programu mocniny můžeme zapsat dvěma způsoby, a to dvěma hvězdičkami ** nebo pomocí střížky ^. Klávesová zkratka ^ pro Windows je AltGr + š. Do programu tedy zapíšu *2**64* nebo *2^64* (Obrázek 5). Kde 2 je základ mocniny (mocněnec) a 64 je můj exponent (mocnitel).

Odmocnina bohužel nemá v programu symbolický zápis, musíme teda použít funkci *sqrt*. Zápis odmocniny čísla 64 bude vypadat *sqrt(64)*, to platí jen pro odmocninu dvěma, jestliže bychom chtěli udělat odmocninu třemi, zapsali bychom to jako mocninu se zlomkem. Můžeme opět zapsat dvěma způsoby, a to *64^(1/3)* nebo *64**(1/3)*. Zde je už důležité používat správně závorky, jinak by nám nevyšel správný výsledek. Význam závorek se liší na základě jejich použití.

```
(%i1) 2^^64;
(%o1) 18446744073709551616

(%i2) 2^64;
(%o2) 18446744073709551616

(%i3) sqrt(64);
(%o3) 8

(%i7) 64^(1/3);
(%o7) 4

(%i8) 64^^(1/3);
(%o8) 4
```

Obrázek 5 Mocniny a odmocniny

K výpočtům v programu můžeme využít konstanty, které se v programu nachází. Pro nás nejdůležitější konstanta bude Ludolfovo číslo (pí) značíme π, v programu zapíšeme *%pi* (Obrázek 6). Při výpočtech s konstantou π musíme zmínit možnost výpočtu nesprávného výsledku. V situaci, kdy žáci budou počítat v sešitě použijí hodnotu π = 3,14 když, ale budeme počítat v programu dojde k použití jiné hodnoty a tím pádem nám vyjde odlišný výsledek.

```
(%i1) %pi;
(%o1) π
```

Obrázek 6 Zápis konstanty %pi

Při naší práci budeme přiřazovat proměnným určitou hodnotu (Obrázek 7). Tím docílíme pomocí symbolu dvojtečky : . Když budu chtít napsat, že *a* se má hodnotu dva, zapíšu to *a:2*. Zde si musím dát pozor, abych pro určité proměnné nestanovili více hodnot, jestliže bychom znovu použili proměnou *a*, ale v tomto případě ji přiřadili hodnotu pět *a*:5, dojde k přepsání původní hodnoty. Proměnná *a* již nemá hodnotu dva, ale má novou hodnotu, a to hodnotu pět. S proměnnými, které mají přiřazenou hodnotu, mohu provádět matematické operace. Symbol = používáme v programu pro vyjádření rovnic.

```
(%i1) a:2;
(%o1) 2

(%i2) a+2;
(%o2) 4

(%i3) a:5;
(%o3) 5

(%i4) a;
(%o4) 5
```

Obrázek 7 Využití proměnných v programu

Při zápisu funkcí budu využívat symbol : = (Obrázek 8). Pro funkce platí, stejná pravidla jako pro proměnné, mohu přiřadit určitou funkci a program si funkci zapamatuje, takže funkci mohu znovu vyvolat, ale taky nahradit. Při zápisu pomocí := oproti : dochází k symbolickému přiřazení a zachovává se algebraický zápis. Můžeme uplatnit u výpočtu obsahu čtverce $S(x):=a^2$.

```
(%i1) f(x):=x^2+3·x−1;
(%o1) f(x):=x^2+3 x−1

(%i3) f(x);
(%o3) x^2+3 x−1
```

Obrázek 8 Zápis funkce

Pro vymazání přiřazení určité proměnné a hodnoty můžeme použit výraz *kill*. Napíšeme výraz *kill* a do závorky za to proměnou, ze které chceme vymazat hodnotu. Pro vymazání hodnoty v proměnné *x* použijeme: *kill(x)*. Pro vymazání všech proměnných a hodnot

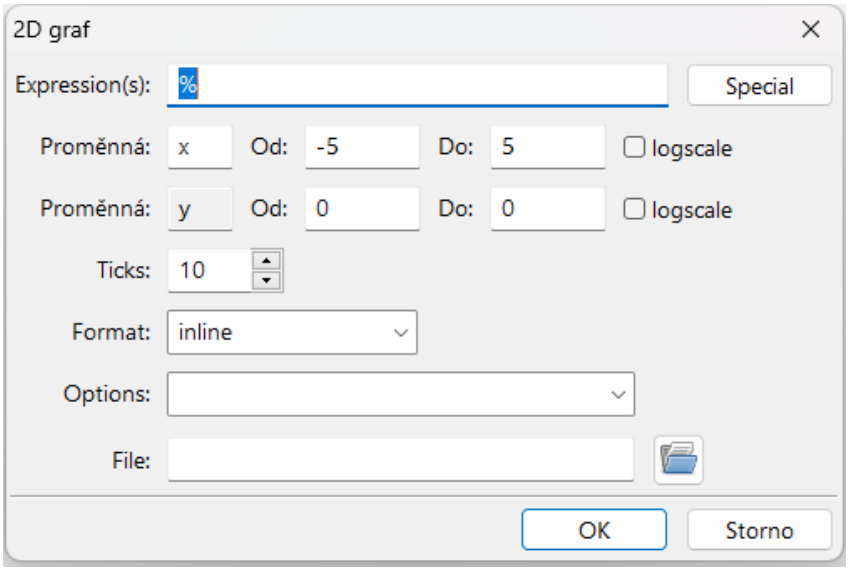
můžeme použít výraz: *kill(all)*. Výraz *kill(all)* využíváme při zápisu nového kódu abychom se ujistili o vymazání hodnot z předešlého kódu.

2.6 Grafické rozhraní

V programu wxMaxima můžeme využívat grafické funkce. Můžeme využít grafické znázornění ve 2D nebo 3D podobě.

2D Graf

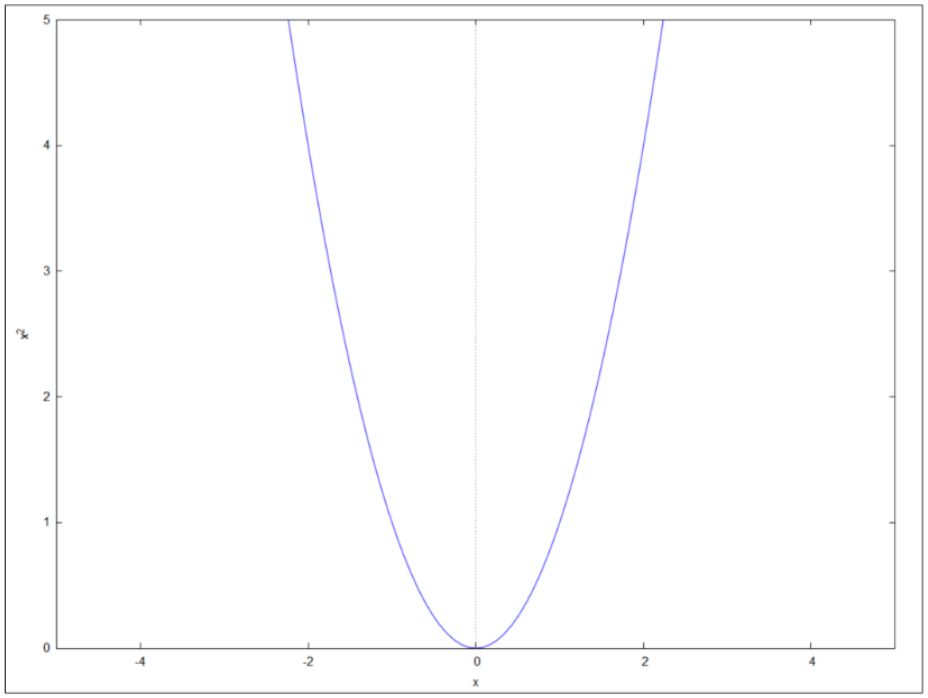
Graf můžeme vytvořit různými způsoby. První způsob je pomocí funkce *Grafy* (Obrázek 9), kterou najdeme na horní liště. Po vybrání nám vyskočí okno k vyplnění, zde vyplníme údaje o požadovaném vzhledu grafu. Údaje jako rozsah na ose x a na ose y, formát vykreslení grafu, jestli chceme vykreslit graf do programu nebo do vedlejšího okna.



Obrázek 9 Tabulka pro vykreslení 2D grafu

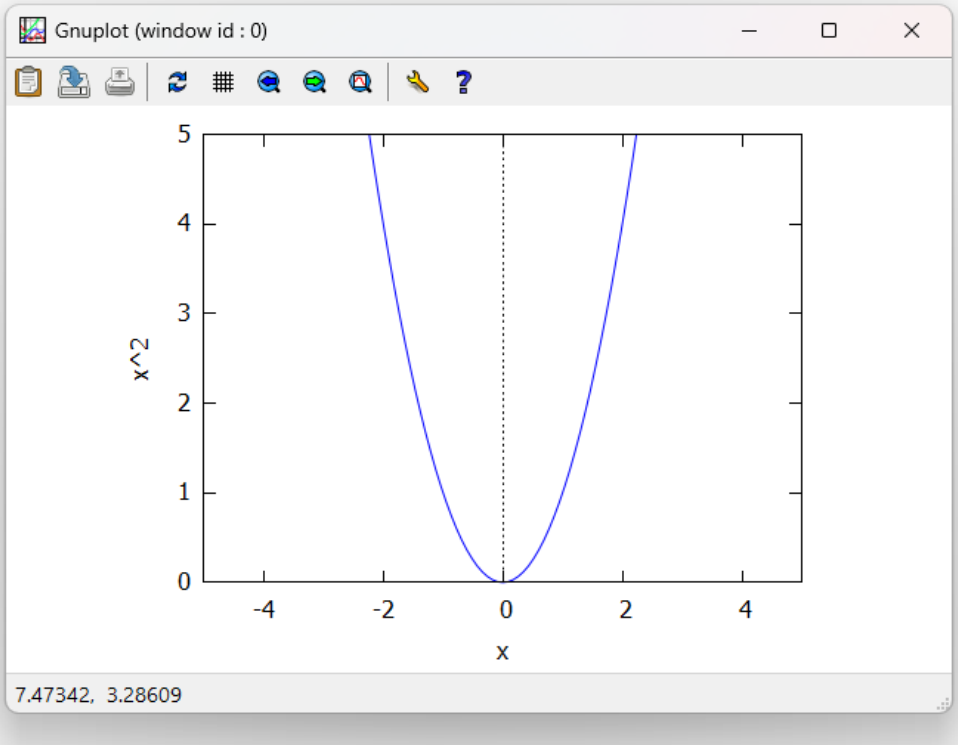
Další způsob je pomocí příkazu *wxplot2d* (Obrázek 10). Příkaz obsahuje tři části, které jsou odděleny závorkami a jednotlivé části jsou v hranatých závorkách. První část obsahuje zadanou funkci, druhá a třetí část nám stanovuje rozsah x a y souřadnic grafu, který se nám zobrazí. Jestliže použijeme příkaz *plot2d* (Obrázek 11) místo příkazu *wxplot2d*, dojde k vykreslení grafu v jiném okně. Okno ponese název Gnuplot. Graf v programu je schopen dalších úprav, nastavení mřížky, ale taky okamžitého tisku. V případě použití příkazu *wxplot2d* dojde k vykreslení grafu přímo v programu. Ať už použijeme jeden nebo druhý příkaz, v obou případech lze graf uložit nebo kopírovat.

(%i2) wxplot2d([x^2], [x,−5,5],[y, 0,5])$



Obrázek 10 Funkce vykreslená pomocí wxplot2d

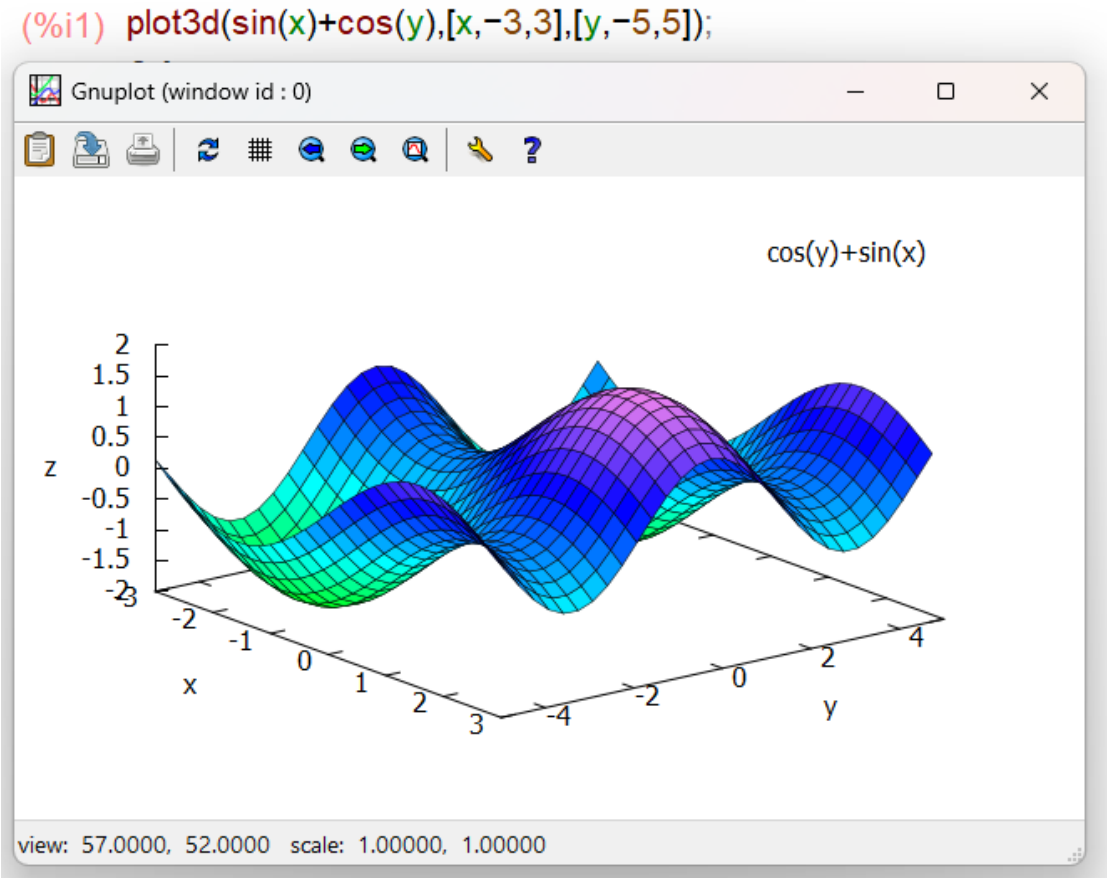
(%i3) plot2d([x^2], [x,−5,5],[y,0,5])$



Obrázek 11 Funkce vykreslená pomocí plot2d

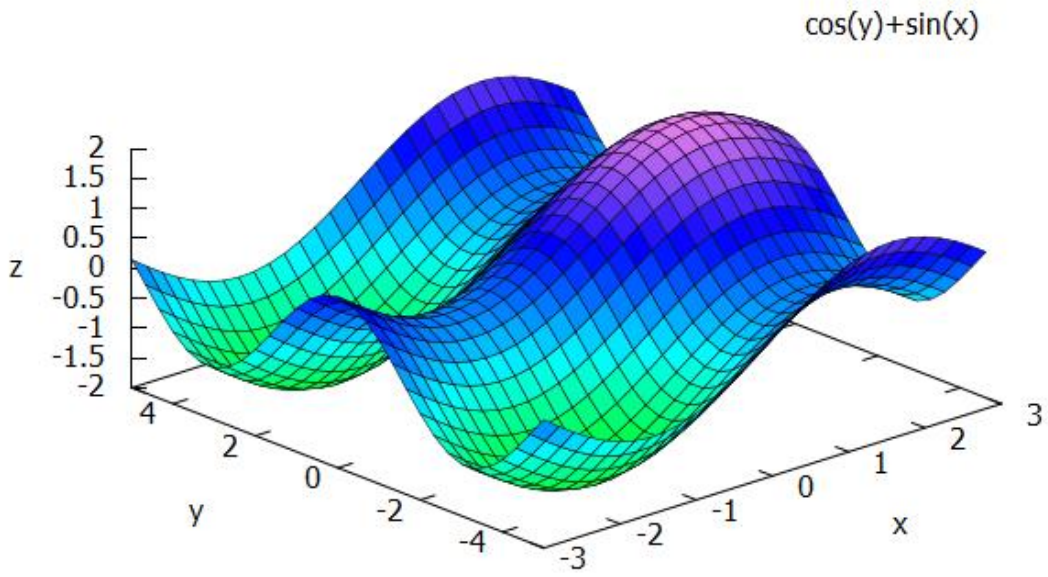
3D Graf

Jak jsme si již řekli, v programu je možné vykreslit funkce i ve 3D (Obrázek 11,12). Pro vykreslení funkcí ve 3D používáme příkazy *wxplot3d* a *plot3d*. V příkazech znovu zadáváme rozsah x a y osy. Funkci opět můžeme zobrazit pomocí funkce *Grafy*.



Obrázek 12 Funkce zobrazená v 3D

Ve 3D zobrazení můžeme použít dříve jmenované funkce jako je tisk, uložení grafu, kopírování grafu, zobrazení grafu s mřížkou. Ve 3D zobrazení, ale oproti 2D zobrazení můžeme využít posuvnou funkci a vykreslený graf si zobrazit z libovolného pohledu. To nám zdokonaluje schopnost zobrazovat funkce.



Obrázek 13 Zobrazení funkce z jiného pohledu

3 PRAKTICKÉ VYUŽITÍ PROGRAMU WXMAXIMA VE VÝUCE

Ve třetí kapitole se podíváme na praktické využití programu wxMaxima při výuce matematiky na druhém stupni základní školy. Ukážeme si využití v jednotlivých ročnících na druhém stupni. Budeme postupovat v jednotlivých ročnících a v jednotlivých tématech tak, aby na sebe navazovala. Nicméně budeme vybírat témata, tak aby jejich řešení v programu bylo užitečné a přínosné. V předchozí kapitole jsme se již zmínili o tom, že programy mohou být využity třemi způsoby. Žák může program využívat samostatně, skupinově nebo program může využívat učitel k demonstraci ukázek pro celou třídu. V celé kapitole budeme využívat knižní série učebnic matematiky pro 6.- 9. ročník základních škol.[20] Každá kapitola a podkapitola zahrnuje základní teorii a potřebné informace, další informace a detaily najdeme v učebnicích.

3.1 Šestý ročník

V matematice šestého ročníku si ze začátku zopakujeme znalosti z předchozích ročníku, a to především z pátého ročníku. Dále si společně probereme desetinná čísla, úhel, prvočísla, znaky dělitelnosti, nejmenší společný násobek a největší společný dělitel. Teorii a příklady jsme čerpali z učebnice matematiky pro 6. ročník základních škol.

3.1.1 Opakování z 5. ročníku

V rámci opakování přirozených čísel zde narazíme na zaokrouhlování přirozených čísel. Musíme si zde připomenout, že v případě zaokrouhlování se řídíme nejbližší číslicí nižšího řádu. Jestliže se za číslicí našeho řádu, na který chceme zaokrouhlit, nachází čísla 0,1,2,3 nebo 4, zaokrouhlujeme dolů a jestliže se ale za číslicí nachází 5,6,7,8,9, zaokrouhlujeme nahoru.

V programu wxMaxima k zaokrouhlování využíváme funkce *round* (Obrázek 14). Při zadání příkazu *round 48* nám program vygeneruje výsledek 48. Funkce *round* totiž primárně

[20] COUFALOVÁ, Jana. *Matematika pro 6. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 19.04.2023]. ISBN 978-80-7168-992-8.

COUFALOVÁ, Jana. *Matematika pro 7. ročník základní školy.* 3. vydání. Praha: Fortuna, 2017. [cit. 20.04.2023]. ISBN 978-80-7373-141-0.

COUFALOVÁ, Jana. *Matematika pro 8. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-994-2.

COUFALOVÁ, Jana. *Matematika pro 9. ročník základní školy.* 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-995-9.

slouží k zaokrouhlování na přirozená čísla, a tak samotný příkaz nám není schopen zaokrouhlit číslo na desítky. Nicméně existuje zde způsob, jak pomocí funkce *round* zaokrouhlit na desítky, stovky nebo jakýkoliv námi požadovaný řád. Jestliže bychom chtěli zaokrouhlit číslo 48 na desítky, museli bychom využít zápisu, kdy používáme násobení a dělení 10, zápis by vypadal následovně *round(48/10)*10*. Kdy pomocí dělení v závorce dostaneme číslo 4,8, takovéto číslo funkce *round* umí zaokrouhlit a výsledek bude 5. To ale není požadovaný výsledek a musíme číslo vynásobit 10. Následně dostaneme správný výsledek, a to 50. Musíme tedy číslo vždy vydělit a vynásobit 10, 100, 1 000. Záleží podle toho, na jaký řád chceme zaokrouhlit.

```
(%i1) round(48);
(%o1) 48

(%i2) round(48.45);
(%o2) 48

(%i3) round(48/10)·10;
(%o3) 50

(%i4) round(448/10)·10;
(%o4) 450

(%i15) round(448/100)·100;
(%o15) 400
```

Obrázek 14 Zaokrouhlování v programu

U dělení již známe způsob zápisu pomocí /. To nám umožňuje dělit jen čísla beze zbytku, jinak nám to napíše zlomek nebo při použití funkce *float* desetinné číslo.

Mám zadaný příklad 27: 10 =.Aby nám program vypsal výsledek a zbytek po dělení, využijeme zápis *floor(27/10)* pro vypsaní celočíselného výsledku a *mod(27/10)* pro výpis zbytku po dělení (Obrázek 15). Celý zápis tedy může vypadat *vysledek= floor(27/10); zbytek= mod(27,10).*

```
(%i17) vysledek= floor(27/10);
       zbytek= mod(27,10);
(%o16) vysledek =2
(%o17) zbytek =7
```

Obrázek 15 Celočíselné dělení se zbytkem

V prvním kroku dochází k výpočtu podílu (*floor*). Mohu se zeptat: „Kolikrát se nám vejde číslo 10 do 27?“ Tím získáváme svůj podíl. V druhém kroku vypočítáváme zbytek po dělení (*mod*) neboli jsme vypočítali, kolikrát se mi vleze 10 do 27 a teď počítáme, co nám po vydělení zbyde.

Dalším tématem, který bychom měli probrat v rámci opakování jsou zlomky, nicméně ty si ukážeme až je budeme detailně probírat v učebnici pro 7. ročník.

3.1.2 Desetinná čísla

U desetinných čísel si ukážeme, jak desetinná čísla sčítáme, odčítáme, násobíme a dělíme. Z minulých ročníků bychom měli mít znalost pojmenovat jednotlivé řády desetinných čísel a porovnávat desetinná čísla. Pro počítání s desetinnými čísli používáme námi již známé operátory. Při sčítání desetinných čísel nezáleží na pořadí sčítanců a postupujeme tak, že přičítáme čísla na jednotlivých řádech k sobě. Při odčítání postupujeme stejně, pouze si musíme dát pozor na to, abychom odčítali menší číslo od většího (Obrázek 16).

U násobení desetinných čísel postupujeme, že vynásobíme čísla jako čísla přirozená a v součinu oddělíme čárkou tolik desetinných míst, kolik jich má desetinné číslo.

Při dělení desetinného čísla desetinným číslem, uplatňujeme definici, že podíl dvou desetinných čísel se nezmění, když vynásobíme dělence i dělitele stejným číslem. Postupujeme tedy tak, že vynásobíme dělence i dělitele stejným číslem, tak abychom dělili přirozeným číslem.

```
(%i1) 6.304+0.016;
(%o1) 6.32

(%i2) 350.247-169.801;
(%o2) 180.446

(%i4) 2.1·5.4;
(%o4) 11.34

(%i12) 9.3/6.2;
(%o12) 1.5
```

Obrázek 16 Základní počty s desetinnými čísli

Uvedeme si zde zajímavou úlohu zaměřenou na desetinná čísla, kterou si společně vyřešíme.

Paní Půtová zaplatila v textilní galerii dvěma stokorunami. Kolik korun dostala nazpět, když koupila 4,5 m prádlové gumy, 0,75m bavlněné krajky, 3m stuhy a 12 knoflíků (Obrázek 17)? Cena prádlové guma za 1 m je 7,60 Kč, cena bavlněné krajky za 1 m je 12,10 Kč, cena stuhy za 1 m je 8,40 Kč a cena knoflíku za 1 kus je 3,80 Kč.

Pomocí znaku /* provedeme zápis slovní úlohy. Pak pomocí násobení a stanovení hodnoty vypočítáme částečné výpočty za jednotlivé položky, ty všechny sečteme a dostaneme hodnotu celkové útraty. Posledním krokem je výpočet peněz, které vrátí Paní Půtové po zaplacení s dvěma stokorunami.

```
(%i62) /* Určíme cenu za každou položku: */
       /* Cena prádlové gumy za 1 m = 7,60 Kč */
       /* Cena bavlněné krajky za 1 m = 12,10 Kč */
       /* Cena stuhy za 1 m = 8,40 Kč */
       /* Cena knoflíku za 1 kus = 3,80 Kč */

       /* Vypočítáme celkovou cenu za každou položku: */
       cena_prádlové_gumy: 4.5 · 7.60;
       cena_bavlněné_krajky: 0.75 · 12.10;
       cena_stuhy: 3 · 8.40;
       cena_knoflíků: 12 · 3.7;

       /* Spočítáme celkovou cenu nakupovaného zboží: */
       celková_cena: cena_prádlové_gumy + cena_bavlněné_krajky + cena_stuhy + cena_knoflíků;

       /* Vypočteme, kolik peněz paní Půtová dostala nazpět: */
       dostala_nazpět: 200 − celková_cena;

       /* Výsledek: */
       dostala_nazpět;

(%o56) 34.2
(%o57) 9.075
(%o58) 25.2
(%o59) 44.40000000000001
(%o60) 112.875
(%o61) 87.125
(%o62) 87.125
```

Obrázek 17 Výpočet slovní úlohy s desetinnými čísly

U desetinných čísel si ještě můžeme ukázat jejich porovnávání pomocí programu (Obrázek 18). Na obrázku můžeme vidět kód, který nám umožňuje porovnávat desetinná čísla. K porovnávání jsme využili funkce *if*, *elseif* a *then*. Funkce *if* má pro nás význam *pokud*, funkce *then* bychom významově přeložili *potom* a funkci *elseif bychom* přeložili jako *jinak pokud*. Významově bychom kód poskládali, *pokud* a <b *potom* vypsání textu a následuje funkce *jinak pokud*, která stanovuje novou podmínku. Kód tedy funguje tak, že do

proměnných *a, b* vložíme desetinná čísla. Následuje první krok porovnávaní hodnot, kód nejprve srovná číslo v proměnné *a* s číslem v proměnné *b* a jestliže je hodnota *b* větší, vypíše to, že „b je vetší než a“ v případě, že to není pravda, tak následuje druhý krok. Zde kód porovnává, zda hodnota *a* je větší než hodnota *b*, jestli je to pravda, program vypíše „a je větší než b“, v případě, že ani tohle není pravda, následuje poslední možnost a to, že se hodnoty v programu rovnají a program vypíše větu „a je stejné jako b“.

```
(%i21) /* zadej hodnoty a a b */
       a: 3.17$;
       b: 3.18$;

       /* porovnání hodnot */
       if a < b then "b je větší než a"
       elseif a > b then "a je větší než b"
       else "a je stejné jako b";

(%o21) b je větší než a
```

Obrázek 18 Porovnávání desetinných čísel

3.1.3 Úhel

S úhly se v učebnici seznamujeme jako s částí roviny. Učíme se úhly zapisovat a taky měřit. Dále se úhly učíme převádět, protože můžeme úhly zapsat ve stupních nebo v minutách. Nejvhodnější je zapisovat úhly ve stupních a v případě potřeby můžeme zbytek vyjádřit v minutách. K tomu, abychom to uměli, musíme vědět, že jeden stupeň se rovná šedesáti minutám a naopak šedesát minut se rovná jednomu stupni. Následně mohu libovolně převádět stupně na minuty a naopak.

```
(%i6)  /* zadání úhlu v stupních */
       (uhel_stupne: 4)$;

       /* převod na úhel v minutách */
       (uhel_minuty: uhel_stupne · 60)$;

       /* výsledek */
       uhel_minuty;
(%o6)  240
```

Obrázek 19 Převod stupňů na minuty

Zde máme ukázaný kód, kde *(%i6)* je náš vstup při převodu stupňů na minuty (Obrázek 19). Takže zadáme počet stupňů, které bychom chtěli převést na minuty a program nám to v kroku dva převede a v kroku tři už jen vypíše. Na výpisu tedy budeme mít *(%06)*, kde, již budeme mít převedený úhel v minutách.

```
(%i9)  /* zadání úhlu v minutách */
       (uhel_minuty: 180)$;

       /* převod na úhel ve stupních */
       (uhel_stupne: uhel_minuty / 60)$;

       /* výsledek */
       uhel_stupne;
(%o9)  3
```

Obrázek 20 Převod minut na stupně

Na Obrázku 20 můžeme vidět převod úhlu v minutách na stupně. Problémem by mohl nastat, kdyby v zadání například bylo, že máme převést 3 stupně a 15 minut na minuty nebo bychom převáděli takový úhel v minutách, který by nešel celý převést neboli nebyl by dělitelný 60 beze zbytku.

```
(%i52) /* zadání úhlu ve stupních a minutách */
       (uhel_stupne: 3)$;
       (uhel_minuty: 15)$;

       /* převod na úhel v minutách */
       (uhel_minuty1:  uhel_stupne· 60)$;
       (celkovy_pocet_minut: uhel_minuty + uhel_minuty1 )$;

       /* výsledek */
       celkovy_pocet_minut;
(%o52) 195
```

Obrázek 21 Převod úhlu na stupně včetně minut

Na Obrázku 21 vidíme možnost zadání úhlu ve stupních a minutách. Po zadání dojde k přepočtu stupňů na minuty, počet přepočtených minut se uloží do nové proměnné uhel_minuty1. Následně se k tomu připočítá počet minut, které jsme měli již v zadání. Výstup označený *(%o52)* nám již ukazuje konečný počet minut po převodu a součtu.

```
(%i10) /* zadání úhlu v minutách */
       (uhel_minuty: 190)$;

       /* převod na úhel ve stupních */
       (uhel_stupne: floor(uhel_minuty / 60))$;
       (zbytek_minut: mod(uhel_minuty, 60))$;

       /* výsledek */
       uhel_stupne;
       zbytek_minut;
(%o9) 3
(%o10) 10
```

Obrázek 22 Převod úhlu v minutách na stupně se zbytkem

Na Obrázku 22 můžeme naopak vidět neúplný převod úhlu zadaný v minutách na stupně. Využili jsme zde námi již známého dělení se zbytkem a první výstup označený *(%o3)* nám ukazuje, kolik vyšlo stupňů po převodu, druhý výstup označený *(%o10)* nám vypisuje, kolik minut nám zbylo po převodu na stupně. Při převodu 190 minut na stupně nám vyjde 3 stupně a 10 minut.

Mimo převodu úhlů můžeme úhly sčítat a odčítat. U sčítání a odčítání úhlů platí, že sčítáme a odčítáme stupně se stupněmi a minuty s minutami. Součtem dvou shodných úhlů dostaneme úhel dvojnásobný. U odčítání úhlů vždy musíme odčítat menší úhel od většího.

```
(%i14) /* zadej úhel A ve stupních */
       A_stupne: 30$;

       /* zadej úhel A v minutách */
       A_minuty: 15$;

       /* zadej úhel B ve stupních */
       B_stupne: 45$;

       /* zadej úhel B v minutách */
       B_minuty: 20$;

       /* sčítání úhlů stupně*/
       C_stupne: A_stupne + B_stupne$;

       /* sčítání úhlů minuty*/
       C_minuty: A_minuty + B_minuty$;


       /* výpis výsledku */
       [C_stupne, C_minuty];

(%o14) [75,35]
```

Obrázek 23 Součet úhlů

Kód v našem programu funguje tak, že prvně zadáme část prvního úhlu, která je zadaná ve stupních, následně zadáme počet minut (Obrázek 23). Následně zadáme počet stupňů a minut u druhého úhlu. Program nám zvlášť sečte stupně a minuty. Výsledky nakonec vypíše, přičemž první číslo je počet stupňů a druhé číslo je počet minut. Zde by mohlo dojít k námitce, že se děti snažíme naučit převodu minut na stupně, jestliže se náš počet minut ve výsledku je vyšší než 59. Nicméně i tenhle problém umíme pomocí programu vyřešit.

```
(%i48) /* zadej úhel A ve stupních */
       A_stupne: 2$;

       /* zadej úhel A v minutách */
       A_minuty: 1$;

       /* zadej úhel B ve stupních */
       B_stupne: 1$;

       /* zadej úhel B v minutách */
       B_minuty: 60$;

       /* sčítání úhlů stupně*/
       C_stupne: A_stupne + B_stupne$;

       /* sčítání úhlů minuty*/
       C_minuty: A_minuty + B_minuty$;

       /* převedení přebytečných minut na stupně */
       if (C_minuty >= 60) then (
          C_stupne : C_stupne + floor(C_minuty / 60),
          C_minuty : mod(C_minuty, 60)
       )$;

       /* výpis výsledku */
       [C_stupne, C_minuty];

(%o48) [4,1]
```

Obrázek 24 Součet úhlů s přepočtem minut na stupně

Kód na Obrázku 24 je stejný jako ten na Obrázku 23, jediný rozdíl je v automatickém přepočtení minut na stupně. V našem případě jsme zadali, že úhel A má velikost 2 stupně a 1 minutu, úhel B má velikost 1 stupně a 60 minut. Program v následujících dvou krocích sečte zvlášť stupně a minuty obou úhlů. V následujícím kroku se objevuje podmínka vytvořená funkcí *if*, jestliže dojde k tomu, že počet minut je větší nebo rovný 60, následuje dělení minut šedesáti a přičtení k počtu stupňů a funkce *mod* spočítá zbylé minuty a program počet stupňů a minut zobrazí. V našem případě po sečtení tedy máme 3 stupně a 61 minut. Program vyhodnotí, že máme více minut než 59 a pomocí dělení převede 60 minut na stupeň a následně ho přičte a spočítá zbývající minuty a následně vše vypíše. Program nám tedy vypíše výsledek, po sečtení dvou úhlů jsou to 4 stupně a 1 minuta.

3.1.4 Prvočísla

Prvočísla definujeme jako číslo, které má pouze dva různé dělitele, číslo jedna a samo sebe. V našem příkladě jsme ověřovali, zda číslo 7 je prvočíslem (Obrázek 25).

```
(%i13) /* Zde můžeš zadat číslo, které chceš ověřit */
       input_number: 7$

       /* Funkce pro zjištění, zda je číslo prvočíslo */
       is_prime(n) := block(
          if n < 2 then return(false),
          if n = 2 then return(true),
          if mod(n, 2) = 0 then return(false),
          for i from 3 while i^2 <= n step 2 do
             if mod(n, i) = 0 then return(false),
          return(true)
       )$

       /* Výsledek ověření */
       result: if is_prime(input_number) then "Zadane cislo je prvocislo." else "Zadane cislo neni prvocislo.";


(%o13) Zadane cislo je prvocislo.
```

Obrázek 25 Kód pro ověření prvočísla.

V tomto kódu je nejprve nastavena proměnná *input_number*, zde vložíme číslo, u kterého chceme ověřit, zda je prvočíslo. Pak je zde nastavená funkce *is_prime(n)*, která ověřuje, zda je naše číslo v proměnné prvočíslo. Tato funkce ověřuje několik podmínek. První podmínkou je, zda je číslo menší než 2. Kdyby to byla pravda program vypíše *false* (není prvočíslo), protože prvočíslo musí být větší než 1. Pokud je proměnná rovná 2, výsledek je, že proměnná je prvočíslo, ale jestli je proměnná sudé číslo větší než 2, tak program vypíše *false* (není prvočíslo), protože jiné sudé prvočíslo, než číslo dva není.

V druhé části program kontroluje, zda naše číslo vložené v proměnné *n* je dělitelné lichými čísly. Jestliže kód najde dalšího dělitele program vypíše, že číslo není prvočíslem. Jestliže kód dalšího dělitele nenašel, program vypíše, že zadané číslo je prvočíslo. Výsledek, zda námi zadané číslo je nebo není prvočíslo, program vypíše ve výstupu.

3.1.5 Znak dělitelnosti

Znak dělitelnosti nám pomáhá určit, zda dané číslo je dělitelné jiným číslem bez nutnosti provádět samotné dělení. Pro žáky je důležité naučit se jednotlivá pravidla, protože je mohou nadále využít například při krácení zlomků. Žáci se učí pravidla dělitelnosti pro čísla, která

lze dělit 2, 3, 4, 5, 6, 9, 10. Jednotlivá pravidla dělitelnosti jsou. Číslo je dělitelné 2, pokud je jeho poslední číslice 0, 2, 4, 6, 8. Číslo je dělitelné 3, pokud je součet jeho číslic dělitelný 3. Číslo je dělitelné 4, pokud je poslední dvojčíslí dělitelný 4. Číslo je dělitelné 5, pokud je poslední číslo 0 nebo 5. Číslo je dělitelné 6, pokud je číslo dělitelné 2 a 3. Číslo je dělitelné 9, pokud je součet jeho číslic dělitelný 9. Číslo je dělitelné 10, pokud je jeho poslední číslice 0.

V programu WxMaxima jsme napsali kód, který by tuto funkci vyhodnocení udělal za nás (Obrázek 26). Kód v programu ověřuje dělitelnost v rozsahu od 2 do 10. V druhém kroku můžeme vidět seznam dělitelů, kteří jsou uložení do proměnné *divisors*.

Následně program postupně prochází dělitele a to tak, že ověřuje, zda je zadané číslo dělitelné tímto dělitelem. K ověření dochází pomocí již známe funkce *mod*, která vrací zbytek po dělení. Pokud je zbytek roven 0, je zadané číslo dělitelné naším dělitelem. Program následně vypíše, že zadané číslo je dělitelné naším dělitelem nebo že není dělitelné naším dělitelem. Tento proces se opakuje pro každý dělitel v seznamu, a tím postupně vypisuje dělitelnost nebo nedělitelnost u všech dělitelů.

```
(%i3) /* Zde můžeš zadat číslo, které chceš ověřit */
      input_number: 42$

      /* Vytvoření seznamu dělitelů, které chceme zkontrolovat */
      divisors: [2, 3, 4, 5, 6, 7, 8, 9, 10]$

      /* Pro každý dělitel zkontroluj dělitelnost a vypiš výsledek */
      for divisor in divisors do (
        if mod(input_number, divisor) = 0 then
          print(sconcat("Číslo ", input_number, " je dělitelné ", divisor, "."))
        else
          print(sconcat("Číslo ", input_number, " není dělitelné ", divisor, "."))
      );

      Číslo 42 je dělitelné 2.
      Číslo 42 je dělitelné 3.
      Číslo 42 není dělitelné 4.
      Číslo 42 není dělitelné 5.
      Číslo 42 je dělitelné 6.
      Číslo 42 je dělitelné 7.
      Číslo 42 není dělitelné 8.
      Číslo 42 není dělitelné 9.
      Číslo 42 není dělitelné 10.
(%o3) done
```

Obrázek 26 Znak dělitelnosti

3.1.6 Nejmenší společný násobek

Nejmenší společný násobek dvou nebo více čísel je nejmenší číslo, které lze dělit těmito čísly.

K určení nejmenšího společného násobku můžeme přistoupit dvěma způsoby. Pokud bychom měli zadaná čísla 6, 8 a u nich bychom měli určit nejmenší společný násobek, jednodušším způsobem bychom postupovali tak, že bychom vyjmenovali násobky 8 a až bychom narazili na takový násobek, který je dělitelný číslem 6, našli bychom náš nejmenší společný násobek. Postup by vypadal takto: první násobek 8 je 8, číslo 8 není dělitelné 6; druhým násobkem je číslo 16 a ani toto číslo není dělitelné číslem 6; dalším násobkem je číslo 24 a to je dělitelné 6. Našli jsme nejmenší společný násobek. Tento způsob je výhodnější v případě, že máme určit nejmenší společný násobek u menších čísel.

V případě, že máme zadaná větší čísla a u nich máme určit nejmenší společný násobek, je lepší postupovat druhým způsobem. Druhým způsobem budeme postupovat tak, že si zadaná čísla rozložíme na součin prvočísel. Představme si, že nám byla zadána čísla 42 a 63. Čísla tedy rozložíme na součin prvočísel: 42 = 2 * 3 * 7 a 63 = 3 * 3 * 7. U prvního čísla vezmeme všechna prvočísla a u druhého vybereme pouze prvočísla, která jsme nevzali u prvního čísla. Takže u prvního čísla vezmeme prvočísla 2, 3, 7 a u druhého čísla vezmeme pouze jednu 3, zbylá čísla jsme již vybrali u prvního čísla. Z vybraných prvočísel u obou čísel uděláme součin *n* (42,63) = 2 * 3 * 7 * 3 = 126. Nejmenší společný násobek u čísel 42 a 63 je 126.

Na Obrázku 27 můžeme vidět ukázku výpočtu nejmenšího společného jmenovatele v programu wxMaxima. Nejprve uživatel do proměnné *number1* a *number2* zadá čísla, u kterých chce určit nejmenší společný jmenovatel. Poté kód použije vestavěnou funkci *lcm* (Least Common Multiple), která vypočítá nejmenší společný násobek těchto dvou čísel a výsledek uloží do proměnné *lcm_result*. Nakonec kód vypíše výsledek na obrazovku ve formátu "Nejmenší společný násobek čísel 42 a 63 je 126", kde 42 a 63 jsou hodnoty zadávaných čísel a 126 je vypočítaný nejmenší společný násobek.

```
(%i12) /* Zadej dvě čísla, pro která chceš nalézt nejmenší společný násobek */
       number1: 42$
       number2: 63$

       /* Použij funkci lcm pro nalezení nejmenšího společného násobku */
       lcm_result: lcm(number1, number2)$

       /* Vypiš výsledek */
       print("Nejmenší společný násobek čísel ", number1, " a ", number2, " je ", lcm_result, ".");

       Nejmenší společný násobek čísel  42  a  63  je  126 .
```

Obrázek 27 Výpočet nejmenšího společného násobku

3.1.7 Největší společný dělitel

Největší společný dělitel dvou či více čísel představuje nejvyšší číslo, kterým jsou námi zadaná čísla dělitelná.

Při výpočtu největšího společného dělitele budeme postupovat tak, že obě čísla opět rozložíme na prvočísla. Jako příklad použijeme čísla 42 a 63. Při rozkladu na prvočísla dostaneme 42 = 2*3*7 a 63 = 7 * 3 * 3 a využijeme pouze ta prvočísla, která se nachází v obou rozkladech. V našem případě to budou čísla 3 a 7, následně provedeme jejich součin, a tak dostaneme námi hledaný největší společný dělitel D (42, 63) = 3 * 7 = 21.

```
/* Zadej dvě čísla, pro která chceš nalézt největší společný dělitel */
number1: 42$
number2: 63$

/* Použití funkce gcd pro nalezení největšího společného dělitele */
gcd_result: gcd(number1, number2)$

/* Vypiš výsledek */
print("Největší společný dělitel čísel ", number1, " a ", number2, " je ", gcd_result, ".");

Největší společný dělitel čísel  42  a  63  je  21 .
```

Obrázek 28 Výpočet největšího společného dělitele

V kódu nejdříve definujeme dvě proměnné number1 a number2, do kterých uložíme dvě čísla, u kterých chceme najít největší společný dělitel (Obrázek 28). Poté využijeme vestavěnou funkci *gcd* (Greatest Common Divisor), která vypočítá největší společný dělitel dvou zadaných čísel. Výsledek funkce *gcd* je uložen do proměnné *gcd_result*. Na závěr je výsledek zobrazen pomocí funkce *print*, která vytvoří textový řetězec sestávající se z částí "Největší společný dělitel čísel", čísel v proměnných *number1* a *number2* a výsledku uloženého v proměnné *gcd_result*. Celý řetězec je poté zobrazen na obrazovce.

3.1.8 Shrnutí podkapitoly šestý ročník

V podkapitole se zaměřujeme na využití programu v hodinách matematiky v šesté třídě. Uvedli jsme využití programu pro zaokrouhlování čísel, kde jsme využili funkci *round*. Při dělení používáme funkci *floor* a *mod* pro celočíselné dělení se zbytkem. Dále se zabýváme sčítáním, odčítáním, násobením a dělením desetinných čísel. S desetinnými čísly také porovnáváme pomocí *if, then, elseif*. Ukazujeme si převod stupňů na minuty a obráceně. Počítáme nejmenší společný násobek pomocí funkce *lcm* a největší společný dělitel pomocí funkce *gcd*.

3.2 Sedmý ročník

V této kapitole se zaměříme na sedmý ročník a látku probíranou v něm. Ukážeme si výpočty u krácení zlomků, porovnávání zlomků, sčítání a odčítání zlomků, násobení a dělení zlomků, převod na smíšená čísla, přímé a nepřímé úměrnosti, procenta. Teorii a příklady jsme čerpali z učebnice matematiky pro 7. ročník základních škol.

3.2.1 Krácení zlomků

Zlomky krátíme tak, že čitatele i jmenovatele zlomku vydělíme jejich společným dělitelem. Zlomky, které již dále nemůžeme krátit, nazýváme zlomky v základním tvaru. Při počítání se snažíme zlomek zkrátit co největším číslem, aby pro nás byly další operace se zlomkem co nejjednodušší.

```
/* Získání zlomku od uživatele */
print("Zadej zlomek ve tvaru citatel/jmenovatel:")$;
fraction_input : ev(read());

/* Krácení zlomku na základní tvar */
gcd_fraction : gcd(numerator(fraction_input), denominator(fraction_input))$;
```

Zadej zlomek ve tvaru citatel/jmenovatel: 25/15;

$$\frac{5}{3}$$

Obrázek 29 Krácení zlomků

Kód slouží k získání zlomku od uživatele a následnému zkrácení na základní tvar (Obrázek 29). V první části kódu funkce *print* vypíše text „Zadej zlomek ve tvaru citatel/jmenovatel:". Následně uživatel zadává zlomek, který chce zkrátit. Následuje složená funkce *fraction_input: ev(read())*, která pomocí funkce *read* načte text, který jsme zadali a

následně funkce *ev* převede do výrazu, který následně program dokáže zpracovat. Následně je tento výraz uložen do proměnné *franction_input*

V druhém kroku funkce *gcd* vypočítá největší společný dělitel dvou čísel. Ta čísla nám zobrazí funkce *numerator* a *denominator,* kde funkce *numerator* nám vypíše čitatele ze zlomku a *denominator* nám vypíše jmenovatel ze zlomku. Takže máme vypočítaný největší společný dělitel dvou čísel, největší společný dělitel čitatele a jmenovatele. Následně pak dojde ke zkrácení čitatele a jmenovatele. Náš výsledek je uložen do hodnoty *gcd_fraction* a následně vypsán na obrazovku.

3.2.2 Porovnávání zlomků

Kód v programu umožňuje uživateli zadat dva zlomky a porovnat je (Obrázek 30). Na začátku kódu jsme pomocí funkce *print* vyzvání k zadání zlomků. Zlomek zadáváme ve stylu čitatel/jmenovatel. Pomocí funkcí *read* a *ev* dochází k přečtení a uložení zadaného zlomku do proměnné *fraction_1_input*. Následně nás program vyzve k zadání druhého zlomku, celá akce proběhne stejně jako v prvním příkladě, jen dojde k uložení do proměnné *fraction_2_input*.

Následuje druhá část kódu, kde dochází již k samotnému porovnávání zlomků. V kódu dochází k porovnávaní zlomků pomocí operátorů > a <, dále využíváme funkcí *if/elseif/else*. Kód tedy porovná zlomky a vrátí logickou hodnotu *true (pravda)* nebo *false (nepravda)*. V případě hodnoty *true* program pomocí funkce *print* vypíše „První zlomek je větší než druhý zlomek“. V případě hodnoty *false* program následně zkoumá, jestli je druhý zlomek větší než první a v případě, že je tato skutečnost pravdivá program nám vypíše „Druhý zlomek je větší než první zlomek“. V případě, že ani tento příklad není pravdivý, program automaticky vyhodnocuje, že jsou si zlomky rovny a program vypisuje „Zlomky jsou stejné“. Tohle jsou všechny možnosti při porovnávání zlomků. Jedinou zbývající možností je nesprávné zadání zlomku v první části kódu. V tomto případě by program vypsal chybu a zadávání zlomků bychom museli zopakovat.

```
/* Získání prvního zlomku od uživatele */
print("Zadej první zlomek ve tvaru citatel/jmenovatel:")$;
fraction_1_input : ev(read())$;

/* Získání druhého zlomku od uživatele */
print("Zadej druhý zlomek ve tvaru citatel/jmenovatel:")$;
fraction_2_input : ev(read())$;

/* Porovnání zlomků */
if fraction_1_input > fraction_2_input then (
    print("První zlomek je větší než druhý zlomek"),
    false
) elseif fraction_1_input < fraction_2_input then (
    print("Druhý zlomek je větší než první zlomek"),
    false
) else (
    print("Zlomky jsou stejné"),
    false
)$;
```

Zadej první zlomek ve tvaru citatel/jmenovatel: 22/2;
Zadej druhý zlomek ve tvaru citatel/jmenovatel: 22/3;
První zlomek je větší než druhý zlomek

Obrázek 30 Porovnávání zlomků

3.2.3 Sčítání a odčítání zlomků

Pokud mají zlomky stejný jmenovatel, můžeme je sčítat nebo odčítat jednoduše tím, že sečteme nebo odečteme jejich čitatele a jmenovatele opíšeme. V případě možnosti, upravíme zlomek do základního tvaru (Obrázek 31).

(%i5) 1/4+5/4;

$$(\%o5)\ \frac{3}{2}$$

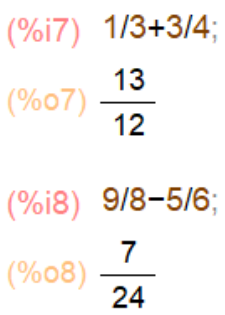
(%i6) 3/6−2/6;

$$(\%o6)\ \frac{1}{6}$$

Obrázek 31 Sčítání a odčítání zlomků se stejným jmenovatelem

U sčítání zlomků s různými jmenovateli, musíme zlomky nejprve převést na společného jmenovatele a poté sečíst čitatele rozšířených zlomků (Obrázek 32).

Podobně postupujeme i při odčítání zlomků, kdy opět zlomky převedeme na společného jmenovatele a následně odečteme čitatele.

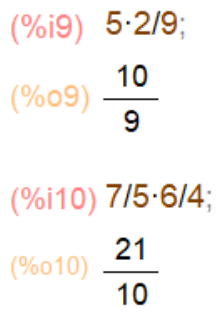


Obrázek 32 Sčítání a odčítání zlomků s různými jmenovateli

3.2.4 Násobení a dělení zlomků

U násobení zlomků přirozeným číslem postupujeme tak, že přirozeným číslem vynásobíme čitatele zlomku a jmenovatele ponecháme bez změny (Obrázek 33).

Zlomek násobíme zlomkem tak, že čitatele vynásobím čitatelem a jmenovatele vynásobím jmenovatelem a součin čitatelů zapíšu v novém zlomku na pozici čitatele a součin jmenovatelů zapíšu na pozici jmenovatele. U násobení zlomků platí, že v případě možnosti upravíme zlomek na základní tvar, ať už během násobení nebo až po vynásobení.



Obrázek 33 Násobení zlomků

U dělení přirozeného čísla zlomkem postupuji tak, že číslo vynásobím převráceným zlomkem. Takže první číslo opíšu, místo dělení dám násobení, zlomek převrátím a vypočítám součin.

Při dělení zlomků zlomkem postupujeme podobně jako při dělení přirozeného čísla zlomkem (Obrázek 34). První zlomek opíšu, místo dělení dám násobení a druhý zlomek převrátím. Následně vypočítám součin a popřípadě zkrátím.

```
(%i11) 9/ (2/7);
```

$$(\%o11)\ \frac{63}{2}$$

```
(%i12) (9/4)/(6/5);
```

$$(\%o12)\ \frac{15}{8}$$

Obrázek 34 Dělení zlomků

3.2.5 Převod na smíšená čísla

Při převodu zlomku na smíšené číslo budeme postupovat tak, že čitatele vydělíme jmenovatelem a výsledek, který nám vyjde po dělení představuje počet celků ve smíšeném čísle. Zbytek, který nám vyjde při neúplném podílu zapíšeme do čitatele a jmenovatel se nemění, tak ho pouze opíšeme. Při zápisu smíšených čísel, většinou neumísťujeme žádnou operaci mezi celé číslo a zlomek, ale můžeme zde umístit operaci sčítání.

Máme zadaný zlomek a chceme ho převést na smíšené číslo. V sešitě by zápis vypadal takto.

$\frac{23}{4}$ 23: 4= 5 (zb. 3) a zápis by vypadal $5\frac{3}{4}$

V programu, jsme provedli převod zlomku na smíšené číslo následujícím způsobem (Obrázek 35).

```
(%i62) /* Zadejte zlomek */
       vstup_zlomek : 23/4$;

       /* Funkce pro převod zlomku na smíšené číslo */
       zlomek_na_smisene_cislo(zlomek) := block(
          celociselna_cast : floor(zlomek),
          zlomek_cast : zlomek - celociselna_cast,
          sconcat(celociselna_cast, "+", zlomek_cast)
       )$;

       /* Zavolání funkce a výstup */
       smisene_cislo : zlomek_na_smisene_cislo(vstup_zlomek)$;
       vystup : sconcat("Smíšené číslo: ", smisene_cislo);
```

```
(%o62) Smíšené číslo: 5+3/4
```

Obrázek 35 Převod zlomku na smíšené číslo

V prvním kroku zadáváme zlomek jako podíl dvou celých čísel. Zlomek je v prvním kroku uložen jako hodnota do proměnné *vstup_zlomek*.

V druhém kroku kódu již převádíme zlomek na smíšené číslo pomocí funkce *floor* a vypočítáváme celočíselnou část smíšeného čísla a vypočtenou hodnotu ukládáme do proměnné *celociselna_cast*. Další proměnnou je *zlomek_cast*, kde vypočítáváme pomocí odčítání zbylou část smíšeného čísla. Tyto dvě proměnné jsou následně spojeny do řetězce pomocí funkce *sconcat*, tato funkce slouží ke spojení textových řetězců.

Poslední část kódu už jen spojuje jednotlivé části kódu a ukládá je do konečné proměnné *smisene_cislo*. Výstup kódu je pak vypsán pomocí posledního řádku, který k vypsání používá opět funkci *sconcat*, která spojí text „Smíšené číslo:“ a smíšené číslo dohromady. Smíšené číslo nám je zde vypsáno ve stylu $5 + \frac{3}{4}$, které odpovídá předešlému zápisu $5\frac{3}{4}$.

Převod smíšeného čísla na zlomek se na základních školách učí především způsobem, kdy celým číslem vynásobím jmenovatele a přičtu čitatele. Následně tento výsledek vložím do čitatele a jmenovatele opíšu. Pro smíšené číslo $7\frac{3}{5}$ by to vypadalo následovně.

$7 \times 5 + 3 = 38, \frac{38}{5}$

V programu bychom využili operace sčítání a jednoduše zapsali celočíselnou část smíšeného čísla plus zlomkovou část smíšeného čísla (Obrázek 36).

```
(%i1) 7+(3/5);
(%o1) 38/5
```

Obrázek 36 Převod smíšeného čísla na zlomek

3.2.6 Přímá úměrnost

Přímá úměrnost má rovnici y = k . x, kde ‚k‘ je koeficient přímé úměrnosti. Přímá úměrnost a její rovnice nám říká kolikrát se zvětší (zmenší) *x*, tolikrát se zvětší (zmenší) *y*. Proměnné *x* a *y* jsou přímo úměrné. Body v grafech přímé úměrnosti jsou umístěny na přímce, která prochází počátkem souřadnic.

Pomocí programu si ukážeme vyřešení jedné úlohy na přímou úměrnost, kde máme zadáno, že brigádník si za 8 hodin vydělal 288 Kč a podle smlouvy má odpracovat 120 hodin (Obrázek 37). Jaká bude jeho mzda?

```
(%i52)/* Zadejte hodnoty veličin */
      hodiny1 : 8$;
      castka1 : 288$;
      hodiny2 : 120$;

      /* Výpočet konstanty úměrnosti */
      k : castka1 / hodiny1$;

      /* Výpočet mzdy */
      castka2 : k · hodiny2$;

      /* Výstup */

      print(castka2);

      4320
```

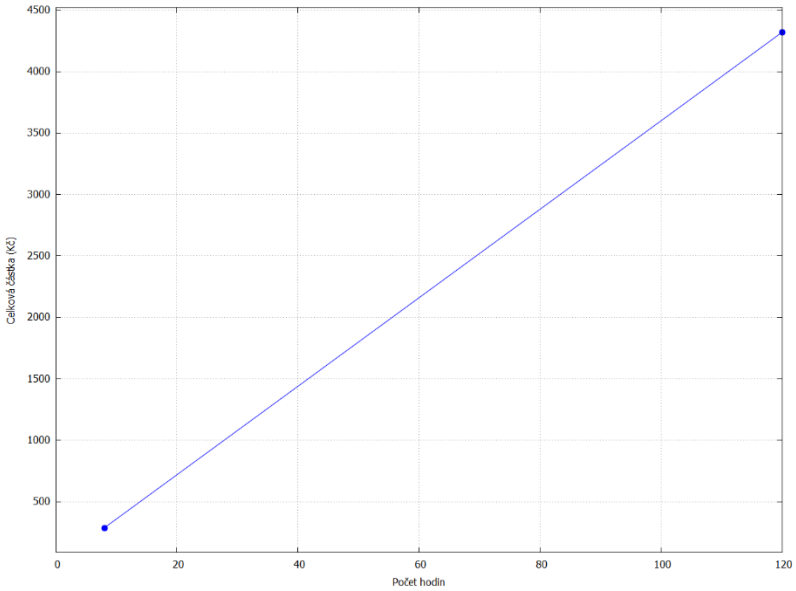
Obrázek 37 Výpočet přímá úměrnost

Kód v programu v prvním kroku vypočítá konstantu úměrnosti a to tak, že vydělí 288 počtem odpracovaných hodin, to je 8. Tak dostáváme konstantu úměrnosti neboli kolik peněz dostaneme za každou odpracovanou hodinu. Následně vynásobíme počet předpokládaných hodin a konstantu neboli počet peněz za hodinu, tuto hodnotu ukládáme do proměnné *castka2* a pomocí funkce *print* vypíšeme hodnotu mzdy po odpracování 120 hodin.

Následně mohu v programu provést vykreslení grafu přímé úměrnosti (Obrázek 39). Kde využijeme zadaných hodnot ve výpočtu. K vykreslení nám pomůže funkce *plot2d,* funkce využívá hodnoty, které jsme si vypočítali v předešlém kódu (Obrázek 38). Využíváme zde dvojici hodnot (hodiny1, castka1) a (hodiny2, castka2) (Obrázek 38). Zde využijeme dvou stylů zobrazení pomocí funkcí *discrete* a *linespoints*. Funkce *discrete* vytváři seznam hodnot, funkce *linepoints* nám v grafu vykresluje body a následně je pomocí linie propojuje. Nakonec zde přidáváme popisky os Počet hodin pro xlabel (osa x) a Celková částka (Kč) pro ylabel (osa y).

```
/* Vykreslení grafu */
plot2d(
   [discrete,[[hodiny1,castka1],[hodiny2,castka2]]],
   [style,linespoints],
   [xlabel,"Počet hodin"],
   [ylabel,"Celková částka (Kč)"]
);
```

Obrázek 38 Kód grafu přímé úměrnosti



Obrázek 39 Graf přímé úměrnosti

3.2.7 Nepřímá úměrnost

Přímá úměrnost má rovnici y = k /x, kde ‚k' je koeficient nepřímé úměrnosti. Nepřímá úměrnost a její rovnice nám říká kolikrát se zvětší (zmenší) *x*, tolikrát se zmenší (zvětší) *y*. Proměnné *x* a *y* jsou nepřímo úměrné. Graf nepřímé úměrnosti tvoří body ležící na křivce. Křivka se nazývá hyperbola.

Ukážeme si zde úlohu a pak pomocí programu vypočítáme. Tři zedníci staví zídku 8 hodin, jak dlouho by stavěli zídku 2 zedníci (Obrázek 40)?

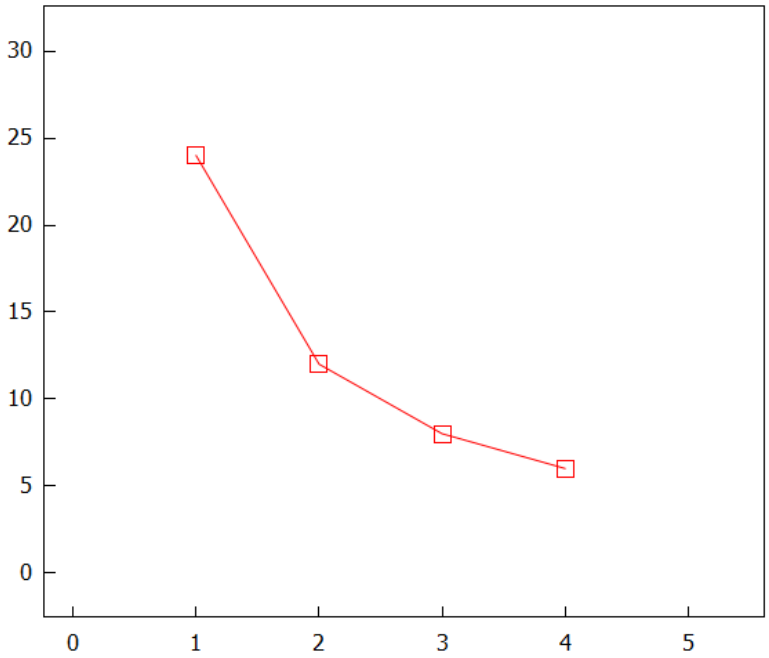
```
(%i4) /* Zadejte hodnoty veličin */
      x1 : 3$; /* počet zedníků */
      y1 : 8$; /* počet hodin pro stavbu zdi s 3 zedníky */
      x2 : 2$; /* počet zedníků */


      /* Výpočet hodnoty neznámé veličiny y */
      solve(x1 · y1 = x2 · y2, y2);
(%o4) [y2=12]
```

Obrázek 40 Výpočet nepřímé úměrnosti

Tento kód slouží k výpočtu nepřímé úměrnosti mezi počtem zedníků a počtem hodin, které potřebují k postavení zdi. Pro zadanou hodnotu počtu zedníků *x1* a počtu hodin *y1*, které potřebují k postavení zdi a pro druhou hodnotu počtu zedníků *x2* hledáme neznámou hodnotu počtu hodin *y2*.

Funkci *solve* použijeme k vyřešení rovnice pro neznámou veličinu *y2*. V tomto případě máme rovnici *x1 * y1 = x2 * y2*, kde *y2* je neznámou veličinou. Kód používá funkci *solve* k nalezení hodnoty *y2*. Následně je vypsaná vypočítaná hodnota, která nám vyjde 12 a graf bude vypadat následovně (Obrázek 41).



Obrázek 41 Vykreslení grafu nepřímé úměrnosti

3.2.8 Procenta

Naší další kapitolou jsou procenta a úroky. Setinu celku můžeme zapsat několika způsoby: $\frac{1}{100} = 0{,}01 = 1\ \%$. Jeden celek má vždy 100 %. Jestliže chceme určit 1 % ze základu, základ vydělíme 100. Máme stanovený tenhle zápis 1 % z 300 = 3, kde 1 % je počet procent, 300 je základ a 3 je procentová část. Zápis bychom přečetli jako: jedno procento z tři set je tři.

Výpočet procentové části ze základu. Jako první krok určíme 1 % ze základu neboli vydělíme 100. Následně získané číslo (1 %) vynásobíme počtem procent.

Příklad: 5 % z čísla 300

```
(%i11) /* Definice proměnných */
       zaklad: 300$;
       pocet_procent: 5$;

       /* Výpočet procenta z části */
       vysledek: zaklad · pocet_procent / 100;
```

```
(%o11) 15
```

Obrázek 42 Výpočet procentové části

V programu výpočet provádíme tak, že v prvním kroku definujeme počet procent a základ, ze kterého následně budeme vypočítávat procentovou část (Obrázek 42). V druhém kroku již vypočítáváme výpočet procentové části. Tady se způsob výpočtu neliší od způsobu, který se žáci učí ve škole. Následně dojde k vypsaní hodnoty.

Zde vypočítáváme základ z procentové části a procent (Obrázek 43). V prvním kroku bychom vypočítali opět jedno procento a to tak, že vydělíme procentovou část počtem procent. Následně získané číslo vynásobíme 100.

```
(%i22) /* Definice proměnných */
       procenta: 4$;
       procentova_cast: 16$;

       /* Výpočet základu */
       základ: procentova_cast · 100 / procenta;
```

```
(%o22) 400
```

Obrázek 43 Výpočet základu

V programu výpočet základu provedeme tak, že opět definujeme proměnné. V tomhle případě definujeme procenta a procentovou část. Následně výpočet základu provedeme tak, že procentovou část vynásobíme stovkou a vydělíme počtem procent. Znovu se postup výpočtu nijak neliší od výpočtu bez využití programu.

3.2.9 Shrnutí podkapitoly sedmý ročník

Text v podkapitole se zaměřuje na matematiku sedmém ročníku. Ukazujeme si využití programu u krácení zlomků, porovnávání zlomků, sčítání a odčítání zlomků, násobení a dělení zlomků, převodu zlomků na smíšená čísla a přímé úměrnosti. U vybraných témat jsme využili již známé funkce, ale také nové. Nejdůležitější novou funkcí je *solve,* která se používá k řešení matematických rovnic.

3.3 Osmý ročník

V této kapitole se zaměříme na látku probíranou v osmém ročníku. Ukážeme si výpočty mocnin a odmocnin, pomocí Pythagorové věty, obvodu kruhu a délku kružnice, výrazů a lineární rovnice. Teorii a příklady jsme čerpali z učebnice matematiky pro 8. ročník základních škol.

3.3.1 Mocniny a odmocniny

V seznámení s programem jsme si již ukázali, jak provádět mocninu a odmocninu, nicméně si zkusíme ukázat jiný způsob výpočtu něž pouhé zadání dvou čísel a operátora. O druhé mocnině platí, že je součinem dvou stejných čísel. Platí tedy $3 . 3 = 3^2$. Kde v 3^2 je 3 základ mocniny a 2 je mocnitel.

```
/* Zadání mocněnce a mocnitele */
zaklad: 3$;
mocnitel: 3$;


/* Definice funkce pro výpočet mocniny */
mocnina: zaklad^mocnitel$;


/* Výpis výsledku */
mocnina;
```

27

Obrázek 44 Výpočet mocniny

V programu v prvním kroku kódu zadáváme dvě proměnné (Obrázek 44). Proměnná *zaklad* nám umožní vložit číslo, které budeme umocňovat, v našem případě je to číslo 3. Proměnná *mocnitel* nám umožňuje zadat číslo, kterým budeme umocňovat, v našem případě jsme zadali číslo 3.

V druhém kroku dojde k samotnému výpočtu, podle vzoru, který jsme si ukázali v seznámení s programem. Výpočet tedy proběhne, takže využijeme operace ‚^' a následně dojde k uložení výpočtu do proměnné *mocnina*. V posledním kroku dojde k vypsání našeho výsledku.

Při výpočtu druhé odmocniny z kladného čísla platí $b^2 = a$, po úpravě $\sqrt{a} = b$. V příkladě $\sqrt{100} = 10$, kde 100 je základ odmocniny a $\sqrt{}$ je odmocnitel. V případě, že u odmocniny není napsaný exponent, jedná se o odmocninu dvěma.

```
/* Zadání základu */
zaklad: 16$;

/* Zadání odmocnitele */
odmocnitel: 2$;

/* Výpočet odmocniny */
odmocnina: zaklad^(1/odmocnitel)$;

/* Výpis výsledku */
odmocnina;
4
```

Obrázek 45 Výpočet odmocniny.

V prvním kroku kód vyžaduje zadání základu pro odmocnění (Obrázek 45). V našem příkladě jsme zadali číslo 16 a hodnotu uložili do proměnné *zaklad*. V druhém kroku jsme zadali číslo odmocnitele, pro nás číslo 2 a hodnotu jsme uložili do proměnné *odmocnitel*. Ve třetím kroku již dochází k samotnému výpočtu, odmocnina je zde zapsaná ve stylu zlomku v exponentu, protože odmocninu vyššího řádu nemůžeme vypočítat jiným způsobem. Následně je výpočet uložen do proměnné odmocnina a ta je vypsána v posledním kroku.

3.3.2 Pythagorova věta

Pythagorova věta nám říká, že je-li trojúhelník pravoúhlý, pak je součet druhých mocnin odvěsen roven druhé mocnině přepony. Když tuto větu zapíšeme pomocí vzorečku vypadá to následovně $a^2 + b^2 = c^2$.

Ukážeme si příklad na výpočet pomocí Pythagorovy věty. Ve cvičení máme zadáno, že máme vypočítat délku přepony pravoúhlého trojúhelníka ABC s odvěsnami a=15 cm, b=8 cm.

```
/* Zadání délek stran */
a: 15$;
b: 8$;

/* Výpočet délky přepony */
c: sqrt(a^2 + b^2)$;

/* Výpis výsledku */
c;
```

17

Obrázek 46 Výpočet pomocí Pythagorovy věty

Výpočet v kódu probíhá tak, že v prvním kroku do proměnných *a, b* stanovíme hodnoty 15 a 8, protože to jsou délky odvěsen, které nám byly zadány (Obrázek 46). V druhém kroku dojde k výpočtu pomocí Pythagorova vzorce. Kdy dojde k umocnění hodnot v proměnných *a, b* a potom následuje sečtení a k odmocnění dvěma. Výsledek je uložen do proměnné *c*. V posledním kroku nám kód vypíše námi hledanou neznámou neboli nám to vypíše délku přepony v trojúhelníku.

Převod do dvojkové soustavy – doplňkové téma.

Když jsme si ukázali možnost využití mocnin, můžeme si ukázat další využití, a to převod z desítkové soustavy do dvojkové soustavy (binární soustavy). Binární kód slouží k zápisu dat nebo informací, pomocí dvou symbolů 0 a 1. Zápis používáme například v počítačích. Převod z desítkové do dvojkové soustavy je založen na dělení se zbytkem (Obrázek 47). Kdy zadané číslo dělíme dvěma, zapíšeme celočíselný podíl a zbytek zapíšeme vedle. Dále postupujeme tak, že podělené číslo dále vydělíme dvěma a opět si zapíšeme zbytek. Opakujeme tento krok, dokud nedostaneme podíl roven 0. Zbytky, které jsme získali při dělení zapíšeme od spodního k vrchnímu. Zápis zbytků zapsaný od spodního k vrchnímu číslu je číslo, které jsme převedli do dvojkové soustavy.

```
(%i70) /* Zadání čísla */
       cislo: 45$;

       /* Funkce pro převod čísla do dvojkové soustavy */
       binaryConversion: block(
          binary: "",
          while cislo > 0 do (
             remainder: mod(cislo, 2),
             binary: concat(remainder, binary),
             cislo: floor(cislo / 2)
          ),
          binary
       )$;

       /* Převod čísla do dvojkové soustavy */
       dvojkova_soustava: binaryConversion;


(%o70) 101101
```

Obrázek 47 Převod čísla do dvojkové soustavy

V první části kódu dochází k zadání čísla, které chceme převést do dvojkové soustavy. Následně zde využíváme funkce *binaryCorversion*, která se použije pro převod do dvojkové soustavy. Uvnitř funkce najdeme jednotlivé kroky, které nám pomáhají k převodu. Funkce *binary* zde slouží k postupnému sestavování binárního zápisu. Následuje funkce *while*, která nám stanovuje podmínku k dalším krokům. Jestli je proměnná *cislo* větší, než nula dochází k dalším krokům, jestli proměnná není větší než nula k dalším krokům nedochází. Jedná se o funkci tvořící cykly. Tedy jestli jsme připuštěni k dalším krokům pomocí funkce *mod* dojde k vypočítání zbytku po dělení. Zbytky po dělení jsou přidány do binárního kódu pomocí funkce *concat*, a tím se postupně sestavuje celý binární zápis čísla. Hodnota uložená v proměnné *cislo* se aktualizuje pomocí celočíselně dělící funkce *floor*. Tato funkce nám hodnotu v proměnné postupně snižuje až dojde k tomu, že naše hodnota v proměnné *cislo* není větší než 0. Následně dochází k poskládání binárního kódu, uložení do proměnné *dvojkova_soustava* a samotnému vypsání binárního kódu.

3.3.3 Obvod kruhu a délka kružnice

Obvodem kruhu je délka jeho hraniční kružnice. Pro výpočet obvodu kruhu používáme vzorec $o = \pi \cdot 2 \cdot r$ nebo $o = \pi \cdot d$. Kde proměnná r je poloměr a proměnná d je průměr, pro ně platí $d = 2 \cdot r$. Proměnná π je Ludolfovo číslo, jehož hodnotu vyjádříme jako $\pi \doteq 3{,}14$.

```
(%i35) /* Definujeme proměnnou pro poloměr kruhu */
       r: 5$;

       /* Výpočet obvodu kruhu */
       obvod: 2 · %pi · r$;

       /* Převedeme na číselnou hodnotu */
       round(float(obvod));

(%o35) 31
```

Obrázek 48 Výpočet obvodu kruhu

V první části kódu definujeme proměnnou *r* a přiřadíme jí hodnotu (Obrázek 48). V našem případě jí přiřadíme hodnotu 5. Jak jsme si již řekli, tato hodnota představuje poloměr kruhu, kdybychom měli zadaný průměr, museli bychom to vydělit dvěma. V druhém kroku probíhá již samotný výpočet, kdy dojde k dosazení proměnné do vzorce a vyvolání proměnné π pomocí zápisu *%pi*. Následně je výpočet uložen do proměnné obvod. Ve třetím kroku využijeme funkce *float*, aby náš výsledek byl převeden na číselnou hodnotu, jinak by nám vyšel výsledek 10 π. Jako poslední funkce je použitá funkce *round*, které nám zajistí, že výsledek vyjde v zaokrouhleném celočíselném tvaru. Následně je tento výsledek vypsán jako výstup.

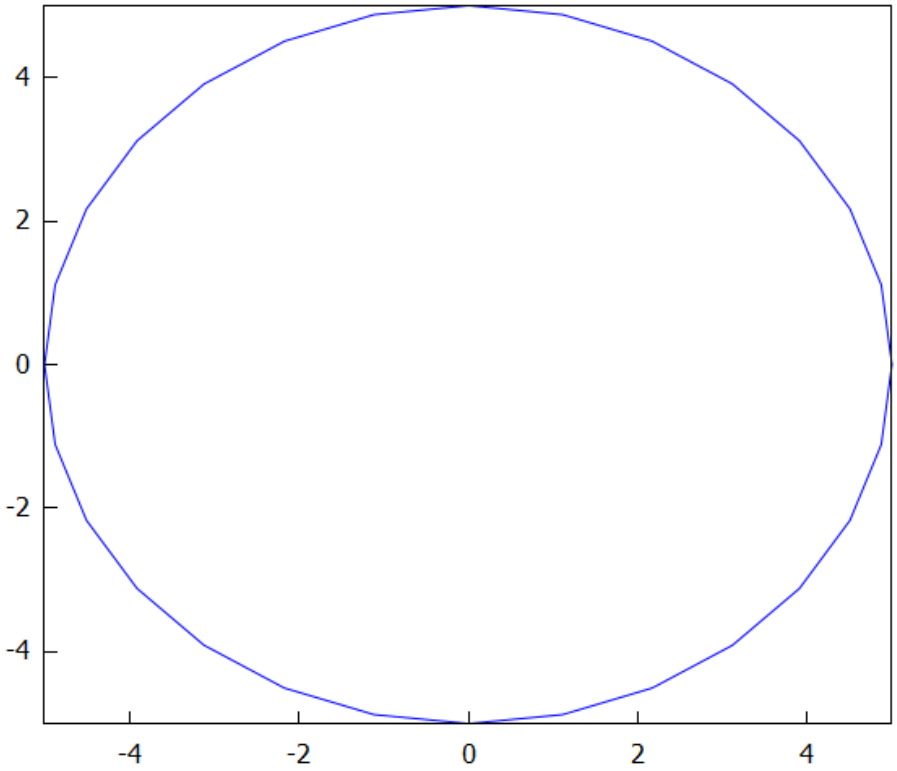
V programu můžeme vykreslit kružnici o zadaném poloměru (Obrázek 50).

```
(%i6) load(draw);
      draw2d(polar(5, theta, 0, 2·%pi));

(%o5) C:/maxima-5.46.0/share/maxima/5.46.0/share/draw/draw.lisp
(%o6) [gr2d(polar)]
```

Obrázek 49 Kód pro vykreslení kružnice

Kód v první části využije funkci *load(draw)*, která slouží k načtení balíčku *draw* (Obrázek 49). Balíček *draw* obsahuje grafické funkce pro vykreslování v programu. Funkce *draw2d* vytváří dvourozměrný graf a přijímá grafické objekty, které mají být vykresleny. V našem případě použijeme funkci *polar,* která vykresluje polární grafy. Parametr 5 je poloměr kružnice a *theta* je proměnná, která reprezentuje úhel. Proměnné 0 a *2*%pi* stanovují rozsah úhlu.



Obrázek 50 Vykreslení kružnice

Pro výpočet obsahu kruhu využijeme vzoreček $S = \pi \, . \, r^2$. V případě zadání průměru místo poloměru by upravený vzorec vypadal takhle $S = \pi \, . (\frac{d}{2})^2$.

```
(%i9)  /* Definujeme proměnnou pro poloměr kruhu */
       r: 5$;

       /* Výpočet obsahu kruhu */
       obsah: %pi · r^2$;

       /* Převedeme na číselnou hodnotu */
       round(float(obsah));

(%o9)  79
```

Obrázek 51 Výpočet obsahu kruhu

V kódu prvně stanovíme zadaný poloměr a hodnota poloměru je uložená do proměnné *r* (Obrázek 51). V druhé části kódu ji využijeme v našem vzorci, kdy ji umocníme dvěma a následně ji vynásobíme Ludolfovým číslem. Výsledek je uložen do proměnné *obsah*. Hodnota v proměnné je v posledním kroku vyjádřena v číselném tvaru a zaokrouhlena.

3.3.4 Výrazy

V kapitole si prvně zopakujeme počítání s číselnými výrazy (Obrázek 52). Výpočet se závorkami a výpočet bez závorek.

```
(%i1) (5−3)·(7+3);
(%o1) 20

(%i2) 2·4+7·3−24/2;
(%o2) 17
```

Obrázek 52 Číselné výrazy

Jestliže v číselném výrazu využijeme místo číslic písmena dostaneme výrazy s proměnnými (Obrázek 53). V případě, že za všechny proměnné dosadíme konkrétní čísla, dostaneme hodnotu výrazu pro dané proměnné.

```
(%i22) a: 5;
       b: 3;

       sqrt((a+b)^2 − (a^2 + b^2));

(%o20) 5
(%o21) 3
(%o22) √30
```

Obrázek 53 Výraz s proměnnými

V kódu stanovujeme hodnoty a ukládáme je do proměnných a, b. Následně tyto hodnoty jsou uložené do našeho výrazu, $\sqrt{(a+b)^2-(a^2+b^2)}$. V případě změny hodnoty dojde k přepočítání výrazu.

Při zápisu výrazu si musíme dát pozor a výrazy zapisovat stylem $5\,.\,a^2$, tento výraz nejde v programu zapsat ve zkráceném zápisu. Nemůžeme tedy napsat $5a^2$.

```
(%i2) 2·a²+5·a²;
(%o2) 7 a^2

(%i3) (3·x+2) + (5−2·x);
(%o3) x +7

(%i4) (3·x+2) − (5−2·x);
(%o4) 5 x −3

(%i5) 2·a²+5·a³;
(%o5) 5 a^3 +2 a^2
```

Obrázek 54 Sčítání a odčítání výrazů

Při sčítání a odčítání jednočlenů platí pravidlo, že sčítáme a odčítáme jednočleny, které mají stejnou proměnnou se stejným exponentem (Obrázek 54). Sečteme nebo odečteme koeficienty a mocninu proměnné opíšeme. Sčítání a odčítání mnohočlenů děláme tak, že odstraníme závorky a následně jednočleny sečteme nebo odečteme.

U násobení jednočlenů vynásobíme koeficienty a vynásobíme mocniny se stejným základem. Dělení jednočlenů provádíme tak, že vydělíme koeficienty a vydělíme mocniny se stejným základem (Obrázek 55).

```
(%i6) 4·5·k;
(%o6) 20 k

(%i7) 8·x²/4;
(%o7) 2 x^2
```

Obrázek 55 Násobení a dělení jednočlenů

Při násobení mnohočlenu jednočlenem – jednočlenem vynásobíme všechny členy mnohočlenu a výsledné součiny sečteme (Obrázek 56). Dělení mnohočlenů jednočlenem – jednočlenem vydělíme každý člen mnohočlenu a výsledné podíly sečteme.

```
(%i1) expand(3·(a+b));
(%o1) 3 b+3 a

(%i3) expand((10·a + 5)/5);
(%o3) 2 a+1
```

Obrázek 56 Násobení a dělení mnohočlenů jednočlenem

V kódu využijeme funkce *expand*, která se využívá k rozložení a rozepsání výrazů. Funkce v našem případě slouží k násobení a dělení mnohočlenů jednočlenem.

Násobení mnohočlenu mnohočlenem provedeme tak, že prvním členem prvního mnohočlenu vynásobíme každý člen druhého mnohočlenu (Obrázek 57). Následně druhým členem prvního mnohočlenu vynásobíme každý člen druhého mnohočlenu. Vzniklé součiny v případě možnosti sečteme.

```
(%i7) expand((a+b)·(c+d));
(%o7) b d+a d+b c+a c

(%i9) expand((x+y)·(2·x+1));
(%o9) 2 x y+y+2 x^2+x
```

Obrázek 57 Násobení mnohočlenů mnohočlenem

V kódu jsme znovu využili funkce *expand*, která nám umožňuje roznásobení mnohočlenů mnohočlenem.

Při úpravě výrazů můžeme využít tři vzorce $(a+b)^2 = a^2 + 2ab + b^2$, $(a-b)^2 = a^2 - 2ab + b^2$, $a^2 - b^2 = (a+b)(a-b)$ (Obrázek 58).

```
(%i2) expand((x+y)²);
(%o2) y^2+2 x y+x^2

(%i3) expand((q−3)²);
(%o3) q^2−6 q+9

(%i6) expand((y−2)·(y+2));
(%o6) y^2−4
```

Obrázek 58 Využití vzorců pro úpravu výrazů

V kódu jsme využili znovu funkci *expand*, dále program využil vzorce pro úpravu výrazů a výrazy upravil.

3.3.5 Lineární rovnice

Rovnice se skládají z výrazů s proměnnou. Proměnou v naší rovnici označujeme písmenem a nazývá se neznámá. Vyřešit rovnice znamená nalézt všechna čísla, která při dosazení za neznámou způsobí, že se pravá i levá strana rovnice budou rovnat. Úpravy, které používám k vyřešení rovnice nazýváme ekvivalentní úpravy.

Pro úpravu rovnic využíváme sčítání, odčítání, násobení a dělení. Při provedení těchto operací na obou stranách rovnice nedojde ke změně kořenů rovnice (Obrázek 59).

```
(%i3) solve(3·x−4=5);
(%o3) [x = 3]

(%i4) solve(3·(2·a+1)=(a−3)·4−5);
(%o4) [a = −10]
```

Obrázek 59 Výpočet rovnic

V programu vyřešíme rovnice pomocí funkce *solve*. Funkce hledá hodnotu pro naši neznámou neboli proměnou, která by splňovala danou rovnici. Výsledek neboli hodnota proměnné je následně vypsána.

Při řešení rovnice se zlomky v prvním kroku odstraníme zlomky (Obrázek 60). Zlomky odstraníme tak, že obě strany rovnice vynásobí nejmenším společným jmenovatelem všech zlomků. Nejmenším společným jmenovatel nevynásobíme pouze zlomky, ale taky všechny členy rovnice.

```
(%i5) solve(2·x+1/2=2/3);
(%o5) [x = 1/12]
```

Obrázek 60 Výpočet rovnice se zlomkem

Vyřešení rovnice se zlomky v programu uděláme stejně, jak v předchozím případě. K vyřešení znovu využijeme funkce *solve*.

```
(%i1) solve(2·x+2·x−2=4·x+1);
(%o1) []

(%i2) solve(2·x+2·x−2=4·x−2);
(%o2) all
```

Obrázek 61 Speciální případy řešení rovnic

V programu jsme si zkusili vyřešit dvě rovnice. Zadání první rovnice je $2\,.\,x + 2\,.\,x - 2 = 4\,.\,x + 1$ (Obrázek 61). Po upravení rovnice bychom měli dostat $0\,.\,x = 3$, to nám říká, že jestliže za x dosadíme libovolné číslo, vždy nám bude vyjde neplatná rovnost $0 = 3$. Na základě výsledku můžeme prohlásit, že rovnice nemá řešení.

Zadání druhé rovnice vypadá následovně $2.x + 2.x - 2 = 4.x - 2$. Po úpravě jsme dostali $0.x = 0$. To nám říká, že můžeme za x dosadit libovolné číslo a vždy dostaneme platnou rovnici. Tím pádem má rovnice nekonečně mnoho řešení.

Lineární rovnice tedy mohou mít jedno řešení, žádné řešení nebo nekonečně mnoho řešení.

V programu si ukážeme ještě jeden kód, který nám pomůže vyřešit lineární rovnice (Obrázek 62). V prvním kroku nám kód pomocí funkce *print* vypíše text "Zadejte rovnici:". Po zadání rovnice nám funkce *read* a *ev* slouží k přečtení a uložení rovnice do proměnné rovnice. Následně je vyraz pomocí funkce *solve* vyřešen a řešení je uloženo do proměnné *reseni*. V posledním kroku kódu dojde k vypsaní textu „Řešení je“ a vypsání samotného výsledku. V případě že výraz má jedno řešení: kód vypíše "Řešení je[$x = výsledek\ výrazu$] „, v případě, že výraz nemá žádné řešení: kód vypíše "$Řešení\ je$ [] " anebo "$Řešení\ je\ all$ " v případě, že máme nekonečně mnoho řešení.

```
/* Zeptejte se uživatele na rovnici */
print("Zadejte rovnici:")$;
rovnice: ev(read())$;

/* Vyřešení rovnice */
reseni: solve(rovnice)$;

/* Výstup řešení */
print("Řešení je ", reseni)$;
```

```
Zadejte rovnici: x+2=x+2;
Řešení je   all
```

Obrázek 62 Řešení výrazů s výpisem

3.3.6 Shrnutí podkapitoly osmý ročník

V osmém ročníku se zaměříme na mocniny, odmocniny, Pythagorovu větu, obvod a délku kruhu, výrazy a lineární rovnice. Při výpočtu příkladů používáme programové funkce. Po seznámení s mocninami jsme si vyzkoušeli doplňkové téma, a to převod do dvojkové soustavy. Novou funkci v kapitole byla funkce *expand*, která slouží k rozšíření a zjednodušení výrazů.

3.4 Devátý ročník

V této kapitole se zaměříme na látku probíranou v matematice v devátém ročníku. Ukážeme si výpočty lomených výrazů, lineárních rovnic s neznámou ve jmenovateli, soustav lineárních rovnic se dvěma neznámými, výpočty a grafické zobrazení funkcí. Teorii a příklady jsme čerpali z učebnice matematiky pro 9. ročník základních škol.

3.4.1 Lomené výrazy

V případě, že máme zapsaný podíl dvou výrazů, říkáme tomu lomený výraz. V našem příkladu máme zlomek a v čitateli máme první výraz a ve jmenovateli máme druhý výraz $\frac{x+5}{3+x}$.

U lomených výrazů se jmenovatel nesmí rovnat nule. Při práci s lomenými výrazy tuto podmínku vždy uvádíme.

Krácení lomených výrazů

Při krácení lomených výrazů musíme čitatele i jmenovatele vydělit stejným nenulovým výrazem nebo číslem.

Krácení výrazů si ukážeme na zadaném příkladu, kdy máme zadáno $\frac{9x^2y}{6x}$, $6x \neq 0$. Hledáme tedy výraz různý od nuly, kterým je dělitelný čitatel i jmenovatel. Čitatel i jmenovatel lze dělit 3, ale i *x*. Takže jmenovatel i čitatel vydělíme *3x*. Po zkrácení nám zbude $\frac{3xy}{2}$.

```
(%i25) /* Zadání lomeného výrazu */
       zadany_vyraz: (9·x^2·y) / (6·x)$;

       /* Zjednodušení lomeného výrazu */
       zkraceny_vyraz: ratsimp(zadany_vyraz)$;

       /* Výpis zjednodušeného lomeného výrazu */
       zkraceny_vyraz;

(%o25)  3 x y
        -----
          2
```

Obrázek 63 Krácení lomených výrazů

V prvním kroku zadáváme náš lomený výraz (Obrázek 63). Zde pro správné zadání musíme použít závorky. Námi zadaný lomený výraz uložíme do proměnné *zadany_vyraz*. Ke

zkrácení neboli zjednodušení výrazu využijeme funkce *ratsimp*. Tato funkce upraví zlomek na jeho nejjednodušší tvar a následně se zjednodušený lomený výraz uloží do proměnné *zkraceny_vyraz*. V posledním kroku dochází k vypsání zkráceného výrazu.

Sčítání a odčítání lomených výrazů

Při sčítání a odčítání lomených výrazů mohou nastat dvě situace. Můžeme sčítat nebo odčítat výrazy se stejnými jmenovateli nebo s různými jmenovateli.

V případě, že sčítáme nebo odčítáme výrazy se stejnými jmenovateli, jmenovatele opíšeme a následně sečteme nebo odečteme čitatele.

V situaci, kdy sčítáme nebo odčítáme výrazy s různými jmenovateli, nejdříve převedeme výrazy na společného jmenovatele. U převodu dvou lomených výrazu na společného jmenovatele musíme nejdříve zkontrolovat, zda nemůžeme zadané jmenovatele nějakým způsobem upravit, ať už pomocí vzorců nebo vytknutí. Následně společný jmenovatel bude součin našich jmenovatelů a čitatele prvního lomeného výrazu rozšíříme o nespolečnou část jmenovatele druhého lomeného výrazu. V dalším kroku provedeme stejný krok s čitatelem druhého lomeného výrazu a jmenovatelem prvního lomeného výrazu. Následně sečtu nebo odečtu rozšířené čitatele.

```
(%i24) /* Zadání prvního lomenného výrazu */
       Prvni_lom: (2) / (x + 3)$;

       /* Zadání druhého lomenného výrazu */
       Druhy_lom: (x) / (x − 1)$;

       /* Sčítání lomenných výrazů */
       soucet: ratsimp(Prvni_lom + Druhy_lom)$;

       /* Výpis výsledku */
       soucet;

(%o24)  (x^2 + 5 x − 2) / (x^2 + 2 x − 3)
```

Obrázek 64 Sčítání lomených výrazů

V programu nejdříve zapíšeme první lomený výraz a uložíme ho do proměnné *První_lom*, následně zapíšeme druhý lomený výraz a uložíme ho do proměnné *Druhy_lom* (Obrázek 64). Následně proběhne součet lomených výrazů a pomocí funkce *ratsimp* dojde k provedení

součtů zadaných výrazů. V posledním kroku již vypíšeme samotný výsledek. V případě že bychom chtěli vypočítat rozdíl dvou lomených výrazů, kód v programu by byl stejný až na rozdílný operátor a správně přejmenované proměnné (Obrázek 65). Nakonec nesmíme zapomenout na stanovení podmínek $x \neq 1$ a $x \neq -3$.

```
(%i40) /* Zadání prvního lomenného výrazu */
       Prvni_lom: (x) / (x + 3)$;

       /* Zadání druhého lomenného výrazu */
       Druhy_lom: (4) / (x − 1)$;

       /* Odčítání lomenných výrazů */
       rozdil: ratsimp(Prvni_lom − Druhy_lom)$;

       /* Výpis výsledku */
       rozdil;
```

$$(\%o40)\quad \frac{x^2-5x-12}{x^2+2x-3}$$

Obrázek 65 Odčítání lomených výrazů

NÁSOBENÍ LOMENÝCH VÝRAZŮ

Násobení lomených výrazů je podobné jako násobení zlomků. V prvním kroku se podíváme, jestli není možné provést krácení do kříže. Následně provedeme součin čitatelů, který lomíme součinem jmenovatelů.

```
(%i64) /* Zadání prvního lomenného výrazu */
       lom_vyraz1: 2 / (x + 3)$;

       /* Zadání druhého lomenného výrazu */
       lom_vyraz2: x / (x − 1)$;

       /* Násobení lomenných výrazů */
       soucin: ratsimp(lom_vyraz1 · lom_vyraz2)$;

       /* Výpis výsledku */
       soucin;
```

$$(\%o64)\quad \frac{2x}{x^2+2x-3}$$

Obrázek 66 Násobení lomeného výrazu

Kód začíná vepsáním prvního lomeného výrazu a uložení do proměnné *lom_vyraz1*, následuje vepsáním druhého lomeného výrazu a uložení do proměnné *lom_vyraz2* (Obrázek 66). Ve třetím kroku proběhne samotné násobení, znovu jsme zde využili funkce *ratsimp* v případě, kdybychom nevyužili funkce *ratsimp*, program by vypsal výraz bez roznásobených jmenovatelů a vypsal by $\frac{2x}{(x+3)(x-1)}$. V posledním kroku dochází již k samotnému vypsaní. Opět si zde stanovíme podmínky, za kterým mají výrazy smysl $x \neq -3, x \neq 1$.

DĚLENÍ LOMENÝCH VÝRAZŮ

Při dělení lomených výrazů postupujeme podobně jako při dělení zlomků. První výraz opíšeme, místo operace dělení napíšeme násobení a druhý výraz převrátíme.

```
(%i84) /* Zadání prvního lomenného výrazu */
       lom1: 2 / (x + 3)$;

       /* Zadání druhého lomenného výrazu */
       lom2: x / (x - 1)$;

       /* Dělení lomenných výrazů */
       podil:ratsimp( lom1 / lom2)$;

       /* Výpis výsledku */
       podil;
```

$$(\%o84)\quad \frac{2x-2}{x^2+3x}$$

Obrázek 67 Dělení lomených výrazů

V programu opět zadáme lomené výrazy a uložíme je do proměnných *lom1* a *lom2* (Obrázek 67). V dalším kroku provedeme dělení dvou lomených výrazů a pomocí funkce *ratsimp* dojde k úpravě výrazů do požadované podoby. Výsledný výraz je uložen do proměnné podíl a ten je v posledním kroku vypsán. Na konci stanovujeme podmínky, za kterých mají výrazy smysl, je důležité stanovit podmínky všech jmenovatelů v příkladu, $x \neq 1, x \neq -3, x \neq 0$.

3.4.2 Lineární rovnice s neznámou ve jmenovateli

Již jsme si ukázali, jak můžeme řešit lineární rovnice pomocí programu wxMaxima a zde si ukážeme, jak řešit lineární rovnice s neznámou ve jmenovateli.

Máme zadanou rovnici $5 = \frac{10}{x}$. Aby lomený výraz měl smysl, musíme stanovit podmínku $x \neq 0$. Abychom odstranili zlomek vynásobíme obě strany rovnice proměnnou x. Po vynásobení a upravení rovnice budeme mít $5x = 10$. Po vyřešení rovnice najdeme kořen $x = 2$, který splňuje námi stanovenou podmínku.

```
(%i3)
        /* Zadání rovnice */
        rovnice: 5 = 10 / x$;

        /* Vyřešení rovnice */
        reseni: solve(rovnice)$;

        /* Výpis řešení */
        reseni;
(%o3) [x =2]
```

Obrázek 68 Výpočet lineární rovnice s neznámou ve jmenovateli

V prvním kroku zadáme rovnici a uložíme ji do proměnné rovnice (Obrázek 68). V druhém kroku dojde k vyřešení zadané rovnice pomocí funkce *solve* a uložení do proměnné *reseni*. V posledním kroku dojde k vypsání výsledku, a to $x = 2$.

V případě, že bychom zadali rovnici, která nemá řešení, program by vypsal: [].

3.4.3 Soustavy Lineárních rovnic se dvěma neznámými

Rovnice, která má tvar $ax + by = c$, kde proměnné *a, b, c* jsou reálná čísla, pro které platí $a \neq 0, b \neq 0, c \neq 0$ a proměnné *x, y* jsou naše neznámé, nazýváme tuto rovnici lineární rovnicí se dvěma neznámými.

Řešením soustavy dvou lineární rovnic se dvěma neznámými můžou být tři. Soustava rovnic má jedno řešení a výsledem je uspořádaná dvojice čísel nebo soustava nemá žádné řešení anebo soustava rovnic má nekonečně mnoho řešení.

Vyřešení soustavy lineárních rovnic se dvěma neznámými, můžeme provést dvěma způsoby. Metodu dosazovací (substituční) nebo metodu sčítací (adiční). Pro lineární rovnice se dvěma neznámými aplikujeme stejné ekvivalentní úpravy jako u lineárních rovnic s jednou neznámou.

Při využití dosazovací metody postupujeme tak, že z jedné rovnice vyjádříme jednu neznámou, a to to vyjádření neznáme dosadíme do druhé rovnice. Dostáváme rovnici o jedné

neznáme, kterou vyřešíme. Vyřešením dostáváme hodnotu první neznámé, tuto hodnotu dosadíme tak, abychom dostali hodnotu druhé neznámé.

Metodu sčítací provedeme tak, že jednu nebo obě rovnice soustavy upravujeme tak, abychom po sečtení upravených rovnic dostali pouze rovnici s jednou neznámou.

```
(%i5) /* Zadání dvou lineárních rovnic se dvěma neznámými */
      rovnice1: x + 5·y = 23$;
      rovnice2: 5·x + 2·y = 23$;

      /* Vyřešení soustavy rovnic */
      reseni: solve([rovnice1, rovnice2], [x, y])$;

      /* Výpis řešení */
      reseni
      ;
(%o5) [[x = 3, y = 4]]
```

Obrázek 69 Řešení dvou lineárních rovnic se dvěma neznámými

V prvním kroku zadáme rovnice do programu a uložíme je do proměnných *rovnice1* a *rovnice2* (Obrázek 69). V druhém kroku dochází k vyřešení rovnic pomocí funkce *solve*. Ve funkci máme vepsány seznam rovnic [rovnice1, rovnice2] a seznam neznámých [x, y]. Výsledek neboli hodnota neznámých je uložena do proměnné *reseni*. V posledním kroku dochází již k samotnému vypsání neznámých a jejich hodnot.

3.4.4 Funkce

Funkci f definujeme jako předpis, který každému prvku množiny D(f) přiřadí jedno číslo z množiny reálných čísel. Množina D(f) je definiční obor funkce f. Množinu reálný čísel, která jsou přiřazená pomocí funkce k definičnímu oboru, nazýváme oborem hodnot funkce a značíme ji H(f).

Funkci můžeme určit pomocí rovnice, grafem nebo tabulkou.

V případě určení funkce pomocí rovnice, může zápis rovnice vypadat následovně: $y = 3x - 1, x \in R$. Zápis $3x - 1$ je předpis funkce a x ∈ R je náš definiční obor. Zápis může vypadat i takhle y = f(x), x ∈ $D(f)$.

Grafem funkce $f: y = f(x), x \in D(f)$ v Kartézské soustavě souřadnic je množina všech bodů, které mají souřadnice [x, y]. Přičemž mluvíme o grafu funkce, jestliže ke každé hodnotě proměnné x máme v grafu přiřazenou nejvýše jednu hodnotu proměnné y.

Lineární funkce je každá funkce zadaná ve tvaru $y = ax + b$. Proměnné a, b jsou reálná čísla. Lineární funkce je vykreslována jako přímka. Funkci definujeme jako rostoucí v případě, že $a > 0$ a naopak je funkce klesající v případě, že $a < 0$. V případě, že proměnná $a = 0$, nazýváme funkci konstantní a přímka je rovnoběžná s osou x.

V zadání příkladu máme sestrojit graf funkce $y = 3x - 2$, $x \in \{-1,3\}$. Graf funkce sestavíme pomocí tabulky tak, že budeme dosazovat do funkce za proměnou *x*. Díky dosazení dojde k výpočtu hodnoty *y* a následně můžeme tyto body o souřadnicích [x, y] zaznamenat do Kartézského systému souřadnic. Podle definičního oboru jsme zjistili, že grafem není množina izolovaných bodů, a tedy můžeme body spojit souvislou čarou.

x	-1	0	1	2	3
y	-5	-2	1	4	7

Tabulka 1: Seznam hodnot a proměnných

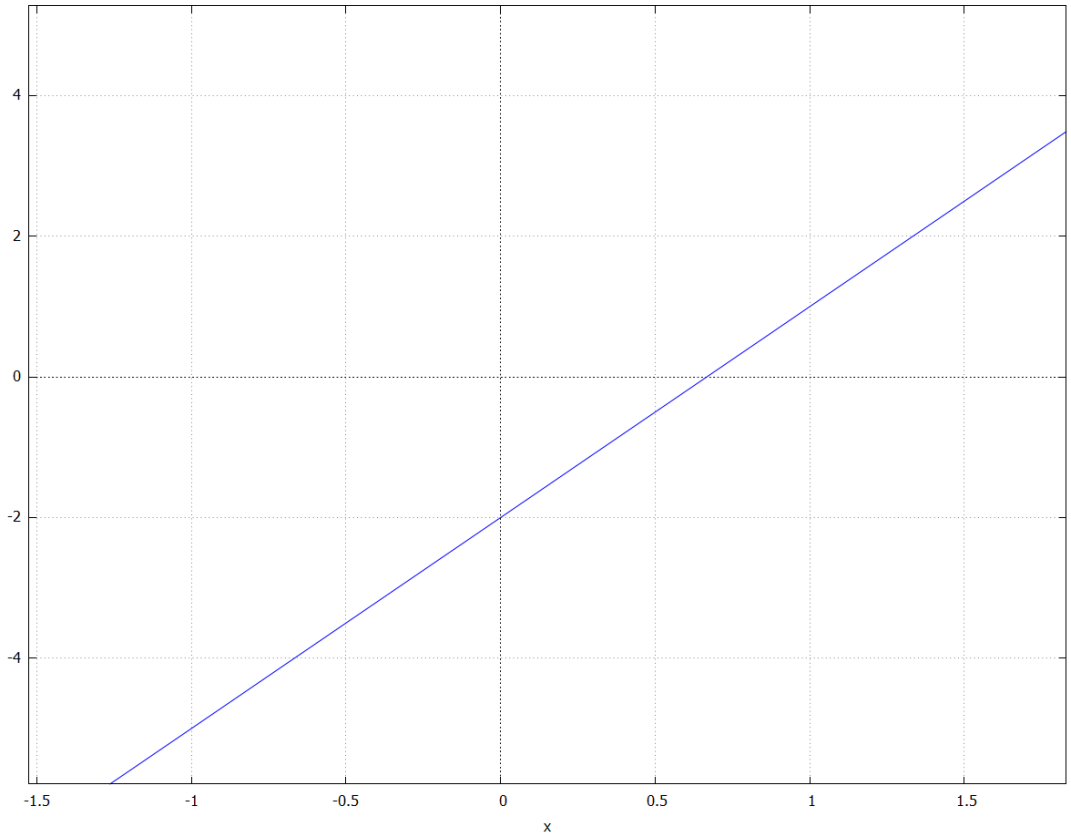
V programu vykreslíme graf tak, že v prvním kroku definujeme výraz *3x -2*, který uložíme do proměné *f(x)* (Obrázek 70). Tím jsme vytvořili matematickou funkci, která přiřazuje hodnotu *f(x)* pro každou hodnotu *x*.

V druhém kroku dochází k vykreslení grafu pomocí funkce *plot2d* (Obrázek 71). Ve funkci máme vloženou naši proměnnou *f(x)*, kterou chceme vykreslit. Druhým argumentem v zadané funkci je [x, -10, 10], který nám stanovuje rozsah osy x. V našem případě máme nastavený rozsah od -10 do 10.

```
(%i16)/* Definice funkce */
      f(x) := 3·x − 2;

      /* Vykreslení grafu */
      plot2d(f(x), [x, −10, 10])$;
```

Obrázek 70 Kód pro vykreslení grafu



Obrázek 71 Graf zadané funkce

3.4.5 Shrnutí podkapitoly devátý ročník

Zaměřujeme se na využití programu pro matematiku devátého ročníku. Podkapitola zahrnuje výpočty lomených výrazů, lineárních rovnic, soustavy lineárních rovnic a grafické zobrazení funkcí. Řešíme i rovnice s neznámou ve jmenovateli. Při řešení soustav používáme metody dosazovací a sčítací. Ukázali jsme si vykreslení funkce pomocí programu. Nově jsme si ukázali funkci *ratsimp,* která slouží k zjednodušení racionální výrazů.

VYUŽITÍ PROGRAMU WXMAXIMA V HODINĚ MATEMATIKY

V rámci hodiny matematiky v sedmé třídě jsem se rozhodl nezávazně využít program wxMaxima jako pomůcku při výuce. Chtěl jsem žákům představit možnosti programu a ukázal jsem jim tak možnost využití programů v hodině matematiky celkově. Nechtěl jsem žáky nijak přespříliš zatěžovat, abych v nich nevyvolal odpor, ale chtěl jsem je motivovat a podpořit je k další možné práci v tomto programu. Zaměřil jsem se především na práci se zlomky.

Na začátku jsem stručně představil program wxMaxima a stručně žákům ukázal prostředí programu wxMaxima. Také jsem ukázal zadávání různorodých matematické příkladů do programu. Vysvětlil jsem jim zápis v programu u sčítání, odčítání, násobení a dělení zlomků. Poté jsme se společně pokusili provést výpočet zlomkových příkladů a porovnat rychlost výpočtu s manuálním počítáním na tabuli. Porovnání proběhlo způsobem soutěže. Žáci si chtěli vyzkoušet, zda jejich manuální vypočet je rychlejší než výpočet spolužáka, který seděl za počítačem. Následně si chtěli role vyměnit a vyzkoušet si také druhý způsob. Samozřejmě se rychlost výpočtu lišila v závislosti na schopnostech jednotlivých žáků.

Následně někteří žáci začali uvažovat, k čemu všemu se dá program využít. Ukázal jsem žákům další možné výpočty pomocí programu. Vybral jsem látku, kterou jsme měli již probranou, aby dokázali alespoň částečně zhodnotit přínos využití programu. S žáky jsme tedy zkoušeli další různé matematické modelování a příklady již pro ně známé látky. Žáci si uvědomili, že program je velmi užitečný nástroj pro zkoumání a porozumění matematických konceptů.

Jedním z důvodů zájmu žáků o program wxMaxima mohlo být jeho intuitivní rozhraní a jednoduchost použití. Díky přehlednému uživatelskému prostředí byli žáci schopni rychle seznámit se s programem a začít ho využívat k řešení matematických problémů. To bylo pro ně důležité, protože se tak cítili samostatnější a jistější ve svých schopnostech v matematice.

Dalším faktorem mohl být přínos, který program wxMaxima nabízí v rámci matematických výpočtů. Žáci si uvědomili, že pomocí programu mohou provádět složité matematické operace a výpočty s větší přesností a efektivitou. To jim umožnilo rychlejší a efektivnější řešení matematických úloh, které jsem jim předložil.

I když nemám konkrétní data, která by nám umožnila vyvodit závěry, tak usuzuji, že zájem žáků o využití programu wxMaxima během hodiny matematiky byl jasně patrný. Žáci projevovali velké zaujetí a spolupráci a bylo jasně patrné, že byli nadšeni z možnosti využití

programu wxMaxima během hodiny matematiky. Tento zájem byl pravděpodobně podpořen i intuitivním rozhraním programu, jeho přínosem při matematických výpočtech a rozmanitostí matematických funkcí, které program nabízí.

ZÁVĚR

Cílem práce bylo zkoumat a navrhnout možnost využití programu wxMaxima ve výuce na druhém stupni základních škol v České republice. V práci jsme si představili, jakým způsobem dochází k vzdělávání matematiky na druhém stupni základních škol, ať už s využitím technologií nebo bez nich. Dále jsme si nastínily, jaké podmínky musí být dodrženy, aby vzdělávání s využitím počítače nebo jiných technologií, bylo přínosné.

Také jsme si představili program wxMaxima, jako takový. Popsali jsme si jeho historii a základní ovládání v programu.

K tvorbě bakalářské práce jsme využili učebnice matematiky pro 6-9. ročník základních škol od Jany Coufalové a kolektivu. Do práce jsme vybrali kapitoly, u kterých by využití programu ve výuce bylo užitečné a přínosné. Při vypracování práce jsme si zároveň ukázali potenciál využití programu ve výuce, ať už pedagogem nebo žáky. Naše práce tedy může posloužit jako návod či manuál pro správné ovládání programu. Výhody využití programu je možnost vizualizace matematických výpočtů a některých grafických objektů. Program může dále posloužit, jako uvedení do světa programování a techniky. Účinné využití programu ve výuce matematiky by mělo být sloučeno se správnou didaktickou metodikou a pedagogickým přístupem.

Celkově lze stanovit, že program wxMaxima i díky jeho volné dostupnosti, představuje užitečný software pro výuku nebo pro dotváření výuky matematiky na druhém stupni základních škol v České republice. Práce může přispět k rozvoji vztahu mezi žákem a matematikou a technologiemi, dále by mohla posloužit k inspiraci pedagogů k novodobému přístupu ve výuce.

V práci se nám podařilo splnit všechny stanovené cíle práce, díky postupnému vypracování jednotlivých kapitol jsme dokázali navrhnout, jak lze program wxMaxima uplatnit ve výuce matematiky na druhém stupni základnách škol v České republice.

SEZNAM POUŽITÉ LITERATURY

[1] BESHENOV, Lyosha. *Maxima a Computer Algebra System*.[online]. [cit. 21.04.2023]. Dostupné z: http://maxima.sourceforge.net

[2] BUŠA, Ján. MAXIMA: *Open source systém počítačovej algebry.* [online]. [cit. 21.04.2023]. 2006. Dostupné z: http://sccg.sk/~batorova/UPG/priruckaSK.pdf

[3] COUFALOVÁ, Jana. *Matematika pro 6. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 19.04.2023]. ISBN 978-80-7168-992-8.

[4] COUFALOVÁ, Jana. *Matematika pro 7. ročník základní školy.* 3. vydání. Praha: Fortuna, 2017. [cit. 20.04.2023]. ISBN 978-80-7373-141-0.

[5] COUFALOVÁ, Jana. *Matematika pro 8. ročník základní školy*. 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-994-2.

[6] COUFALOVÁ, Jana. *Matematika pro 9. ročník základní školy.* 2., upr. vyd. Praha: Fortuna, 2007. [cit. 30.04.2023]. ISBN 978-80-7168-995-9.

[7] *Didaktika matematiky v 21. století* [online]. Copyright © [cit. 22.04.2023]. Dostupné z: https://mfi.upol.cz/files/29/2904/mfi_2904_256_276.pdf

[8] Edwin L. Woollett. *Maxima by Example*. [online]. California State University, Long Beach. [cit.26.01.2021]. Dostupné z: https://web.csulb.edu/~woollett/mbe.html

[9] FEDRIANI, Eugenio M., MOYANO, Rafael. *Using MAXIMA in the Mathematics Classroom.* 23th April, 2010. [online].[cit. 21.04.2023]. Dostupné z: https://www.researchgate.net/publication/265102308_Using_MAXIMA_in_the_Mathematics_Classroom

[10] HAŠEK, Roman, NORULÁKOVÁ, Michaela. *Program wxMaxima ve výuce matematiky* [online]. Jihočeská Univerzita v Českých Budějovicích, [cit. 7.04.2023]. Dostupné z: http://home.pf.jcu.cz/~hasek/VTM1/wxMaxima_ve_vyuce.pdf

[11] *Komenský rozpracoval základní didaktické zásady - Psychologie, pedagogika.* Psychologie, pedagogika - Vše co student potřebuje vědět [online]. [cit. 22.04.2023].

Dostupné z: https://psychologie-pedagogika.studentske.cz/2008/06/komensk-rozpracoval-zkladn-didaktick.html

[12] KUSÁK, Radim. *Matematické programy* [online]. 2010 [cit. 21.04.2023]. Dostupné z: http://utf.mff.cuni.cz/~kusak/math.php?zalozka=4

[13] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003. [cit. 22.04.2023]. ISBN 80-7315-039-5.

[14] MAXIMA. *GPL CAS based on DOE-MACSYMA - Browse /Maxima-Windows* [online].[cit. 05.03.2023]. Dostupné z: https://sourceforge.net/projects/maxima/files/Maxima-Windows/.

[15] OPENAI. *ChatGPT-4* [AI program]. OpenAI, 2023 [cit. 30.04.2023]. Dostupný z: https://openai.com/blog/chatgpt

[16] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky*. Olomouc: Univerzita Palackého, 2002. ISBN 80-244-0534-2.

[17] RŮŽIČKOVÁ, Bronislava. *Didaktika matematiky 2*. Olomouc: Univerzita Palackého v Olomouci, 2004. Skripta / Univerzita Palackého. Pedagogická fakulta. [cit. 22.04.2023]. ISBN 80-244-0815-5. Dostupné také z: http://www.digitalniknihovna.cz/mzk/uuid/uuid:890c38b0-027b-11e4-89c6-005056827e51

[18] Souza, Paulo & Fateman, Richard & Moses, Joel & Yapp, Cliff.The Maxima Book. [online]. 2004.[cit.26.01.2021]. Dostupné z: https://maxima.sourceforge.io/docs/maximabook/maximabook-19-Sept-2004.pdf

[19] WOSSALA, Jan. *Využití ICT ve výuce matematiky*. [online]. [cit. 23.04.2023]. Dostupné z: https://www.kr-kralovehradecky.cz/assets/rozvoj-kraje/evropska-unie-EHP/krajsky-akcni-plan-rozvoje-vzdelavani/aktualni-informace/vyuziti_ict_ve_vyuce_matematiky.pdf

[20] wxMaxima Developers. *wxMaxima* [online]. [cit. 15.03.2023]. Dostupné z: https://wxMaxima-developers.github.io/wxMaxima/wxMaxima.pdf

[21] 5.2 Vzdělávací oblast - *Matematika a její aplikace* - úvod - DIGIFOLIO. Domů - DIGIFOLIO [online]. [cit. 22.04.2023]. Dostupné z: https://digifolio.rvp.cz/view/view.php?id=10289

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

ICT	Informační a komunikační technologie
GUI	Grafické uživatelské rozhraní