Gymnázium Arabská, Praha 6, Arabská 16
Předmět Programování, vyučující Jan Lána

ISAAC

Ročníkový projekt

Petr Salavec, 4.E

Duben 2020

Obsah

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené.
Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Anotace

Česky:

Mým cílem bylo vytvořit kalkulačku, která by zvládala více, než jenom prosté operace s přirozenými čísly. Má kalkulačka měla zvládat věci jako lineární a kvadratické rovnice, rovnice s vyššími mocninami, nebo práce s logaritmy. Dále měla být schopná vykreslovat grafy, popřípadě i pracovat s rovnicemi s více neznámými a umět si poradit s odmocninami.

English:

My goal was to create a calculator that would be capable of handling more than just simple mathematical operation with natural numbers. My calculator was supposed to handle things such as linear and quadratic equations, higher degree equations, or work with logarithms. Furthermore it was meant to be able to draw graphs, even work with equations with more variables and be capable of dealing with roots.

Zadání projektu:

Mojí ročníkovou prací bude udělat super chytrou kalkulačku, která bude umět počítat rovnice, nerovnice, rovnice s více neznámými, rovnice s imaginárními čísly, rovnice s goniometrickými funkcemi a podobně. Bude také umět počítat a vykreslovat grafy. Zkrátka by měla umět většinu středoškolské matiky a zároveň by také měla ukázat postup řešení. [DOPLNĚNÍ]: Rovnice s jednou neznámou, rovnice s více neznámými (alespoň 2), kvadratické rovnice. Dokážu si představit i rovnice s neznámou s vyšší mocninou a logaritmy. Lehčí goniometrické rovnice, které např. nebudou mít vložené funkce v jiných funkcích, by asi také byly možné.

Úvod

V tomto dokumentu popíši jakým způsobem jsem tuto ročníkovou práci vytvořil, jaké matematické a programovací metody jsem použil na řešení rozličných problémů a zdůvodním proč jsem je použil. Také chci znázornit proč jsem někde použil méně efektivní (i efektní) řešení a demonstruji výhody i nevýhody mých řešení. Na konci samozřejmě zmíním mé osobní zhodnocení své práce a uvedu, kde si myslím, že se nalézají slabé a silné stránky mého projektu. Také se chci jenom velmi stručně věnovat práci na takovýchto projektech.

Toto téma jsem si nezvolil náhodně. V programování mě vždy fascinovaly řešení různých matematických problémů, navíc jsem vždy obdivoval aplikace jako PhotoMath a Desmos, které mi byly v tomto projektu velkou inspirací.

Technologie

Projekt je celý napsán v Javě, pro grafické prostředí jsem použil Java FXML. Jelikož jádro mého projektu neleží v grafické stránce, či uživatelském rozhraní, jako FXML editor jsem použil open-source verzi Scene Builderu. Jakožto IDE jsem využil NetBeans 8.2.
Důvod využití Javy je čistě subjektivní preference jazyka. Samozřejmě jsem zvážil využití jazyků jako Python, nebo C#, ale shledal jsem, že by to mému projektu nepřineslo žádné viditelné výhody.

Rovnice

Lineární a kvadratické rovnice

Algoritmus pro vyhodnocování lineárních a kvadratických rovnic je poměrně prostý. Zakládá se na předpokladu, že všechny lineární a kvadratické rovnice se dají upravit do tvaru: $ax^2 + bx + c = 0$; kde je x neznámá. Z toho vyplývá, že klíčové je zde získat tři hodnoty a,b,c, v momentě kdy je známe, je nalezení řešení poměrně prostou záležitostí.

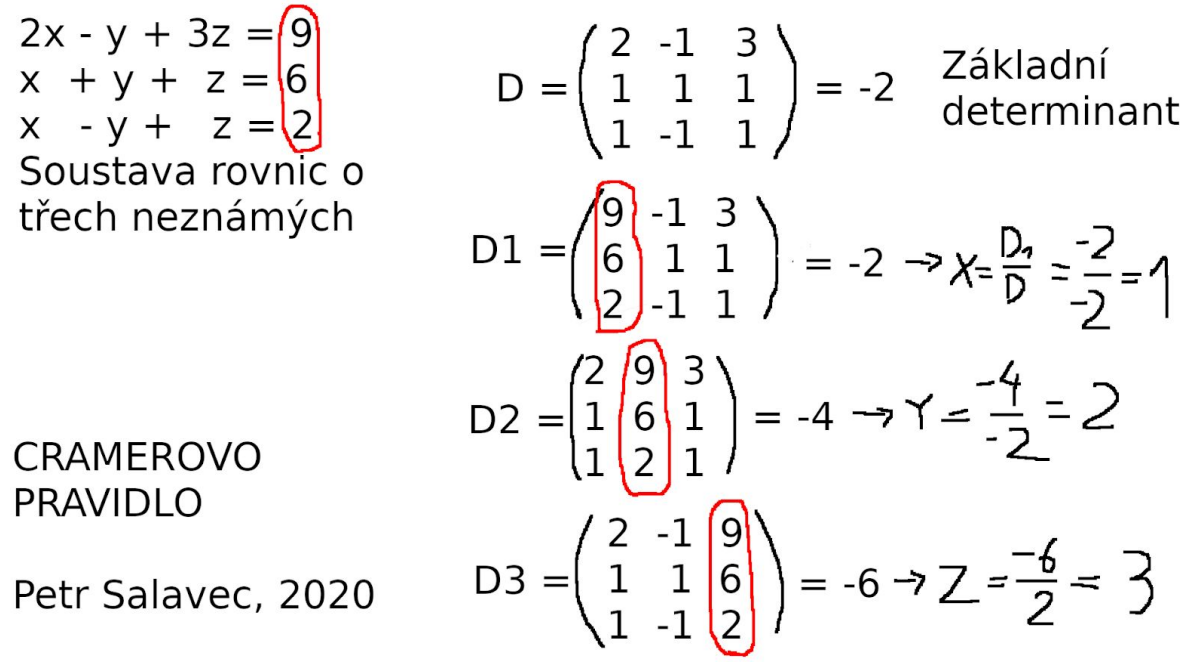
Proces hledání těchto třech hodnot je určitě nejtěžší částí celého mého projektu, neboť způsobů, jak lze zapsat kvadratickou, nebo lineární rovnici existuje mnoho a s nimi existuje i spouta výjimek a zvláštních zápisů. Jako první řešení jsem použil uložení celé rovnice do stromové datové struktury a postupné vyhodnocování rovnice na základě znamének. Důvod bylo, že se velmi podobně řeší normální matematické výpočty a operace. Ovšem problémem se ukázalo to, že to co funguje velmi dobře na normální matematické operace, nefunguje příliš dobře, když se zakomponují do těchto operací neznámé. Příkladem nechť výraz: 3(9/3+3) - zde je patrné, že se nejdříve provedou operace v závorce a s výsledkem závorky se potom pracuje v operacích mimo závorku. To ale není možné u výrazu 3(9/x+3), kde se musí závorka vlastně "zachovat" a celá roznásobit. Takovýchto problémů se samozřejmě vyskytuje mnohem více a stromová datová struktura není na jejich řešení příliš praktická.

Jako druhý způsob řešení jsem zvolil postupné upravování rovnice v poli, až by se došlo k požadovanému tvaru. Podstatou je, že se v tomto poli rovnice postupně upravuje jak to jen jde a části rovnice, které již nejsou na ničem vázané (tudíž nejsou v závorkách, v násobení apod.) se postupně vyškrtávají a jejich hodnoty se ukládají do globálních proměnných. Tím se ušetří čas a obejdou se problémy se závorkami, neboť se vyřeší jako první a provedou se s nimi jejich příbuzné operace. Limitací, kterou jsem ovšem sám a vědomě do programu zakomponoval, je, že není možné provést s jedním členem více než jednu vyšší operaci (násobení, dělení) - tudíž není možné napsat: 3x/3. Důvody pro toto omezení jsou dva - zaprvé by bylo velmi obtížné vůbec ověřit legitimitu takovýchto zápisů a za druhé se takovéto zápisy příliš nevyskytují (nikdy se v matematické učebnici nevyskytuje $3\times3\times x$, toto je rovnou napsáno jako 9x) a jejich vyhodnocování by program ohromě zatížilo.
Po získání těchto tří hodnot se vyhodnotí zdali je rovnice a lineární, nebo kvadratická a na základě toho se vypočítají řešení, buď prostřednictvím diskriminantu nebo dělením lineární rovnice.

Rovnice s více neznámými

Na výpočet rovnic s dvěma a třemi neznámými jsem použil Cramerovo pravidlo. Ačkoliv je možné Cramerovo pravidlo použít na výpočet rovnice o jakémkoli počtu neznámých, od třetího stupně rapidně roste jeho časová náročnost a navíc se rovnice o čtyřech a více neznámých příliš nevyskytují.

Cramerovo pravidlo pracuje s n+1 determinanty rovnic s n neznámými. Základní determinant, se spočítá ze standardní matice sestavené z rovnic. Dále se spočítají zbývající determinanty z matic, ve kterých se vždy nahrazuje n sloupec pravou stranou rovnic. Nakonec se řešení získají dělením těchto determinantů původním determinantem.



Standardní implementací Cramerova pravidla je Leverrier-Faddeevův, nebo Samuelson-Berkowitzův algoritmus, ale jelikož jsem svoje řešení limitoval na rovnice o třech neznámých, mohl jsem implementovat výpočet bez použití rekurzí a cyklů. To sice není tak efektní, ale enormně to zrychlilo výpočet determinantů, který je normálně n^4+O(n^3).

Rovnice vyšších řádů

Na rovnice vyšších řádů se v matematice standartně používá metoda tečen (také Newtonova metoda), tedy hledání derivace a s její pomocí s postupné propracování se k řešení. Toto jsem ale nemohl použít, neboť hledání derivací by bylo velmi náročné, a i kdyby byly nalezeny, neměli bychom garantováno, že se nám podaří dohled kořeny. Proto jsem namísto metody tečen použil trochu zjednodušenou metodu sečen. Ta spočívá v tom, že se zkusí několik "možných" řešení a od nich se potom postupně konverguje ke kořenům rovnice. Výhody metody sečen jsou, že je velmi jednoduché ji implementovat a většinou jí stačí méně iterací než u Newtonovy metody.

Pro mé potřeby jsem určil, že je nepravděpodobné, že by se řešení nacházelo mimo interval (-100;100). Díky tomuto je pak implementace prostá - postupně se zkouší všechna čísla z tohoto intervalu a buď se tak najde řešení rovnou, nebo se najde alespoň číslo, které je k řešení blízko a od toho se potom k řešení konverguje.
Přesněji - pokud se najde číslo, které po dosazení za neznámou do rovnice vrátí výsledek v intervalu (-5;5), tak se pro takové číslo začnou kontrolovat i jeho setinny. Pokud je pak výsledek v intervalu přípustné tolerance chyby (-0,02;0,02), desetinné číslo je považováno za kořen a je zaokrouhleno na jedno desetinné místo.,

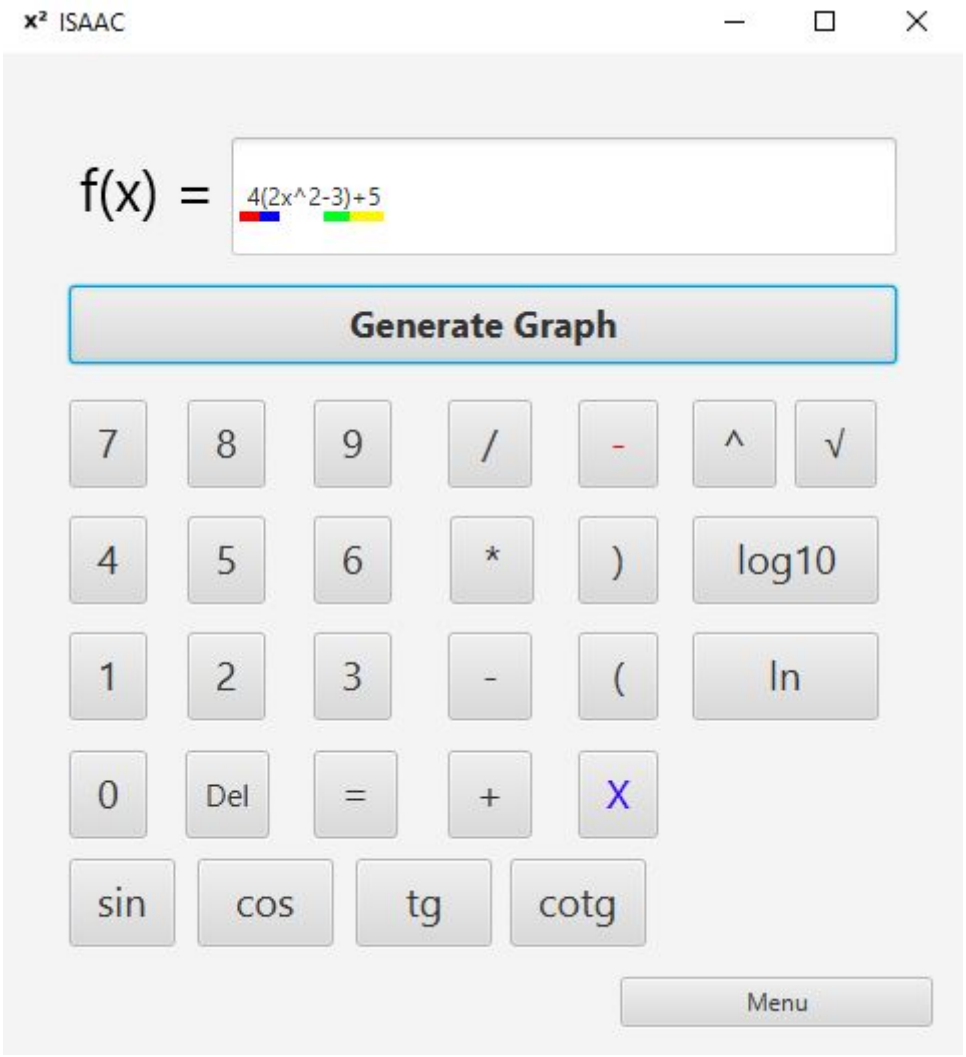
Tento algoritmus funguje na rovnice jakýchkoli řádů, ale z důvodu limitace uživatelského rozhraní jsem ve svém projektu nastavil jako maximální řád pátou mocninu. Druhým důvodem tohoto omezení je to, že u rovnic vyšších řádů se většinou vyskytují v množinách řešení komplexní čísla, se kterými si tento algoritmus nedokáže poradit.

Grafy

Vyhodnocování funkce

Ve funkci je klíčové především určit její typ, tedy zdali je goniometrická, kvadratická, lineární apod. Za účelem zjednodušení jsem pracoval s tím, že se tyto typy nebudou prolínat, tudíž že se nebude třeba v goniometrické funkci vyskytovat vložená funkce kvadratická. Také nepředpokládáme, že se bude rozdílné typy funkcí násobit, nebo dělit. Toto určení typu funkce je důležité především pro její pozdější nákres.

U každé funkce pak předpokládáme maximálně 4 komponenty (5 - pokud počítáme i její typ) - a to absolutní člen přičítán k proměnné(c) a celé funkci(d), násobek proměnné(b) a celé funkce(a) = [a × sin(b × x + c) + d]. V případě funkce: $f(x) = 4(x^2 + 3)$ jsou tyto hodnoty: a - 4; b - 1; c - 3; d - 0. Pomocí těchto čtyř hodnot je pak možné dopočítat jakýkoliv bod na funkci f.

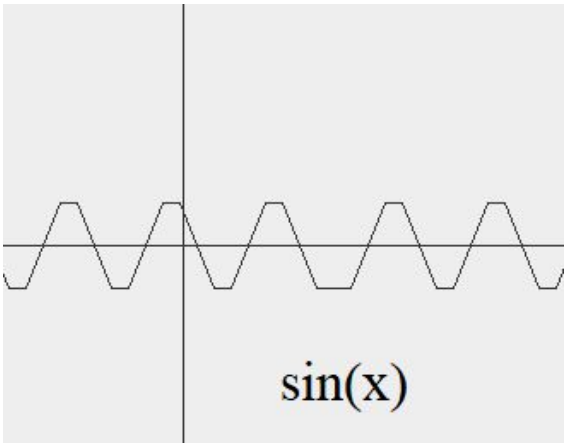


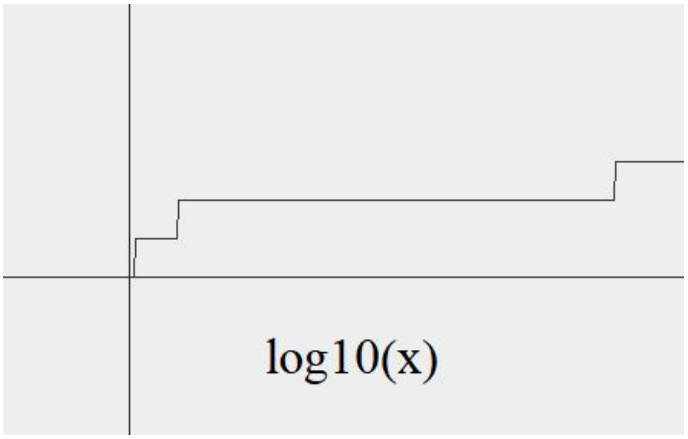
- hodnota násobící funkci
- hodnota násobící proměnnou
- hodnota přičítaná k proměnné
- hodnota přičítaná k funkci

Vykreslování funkce

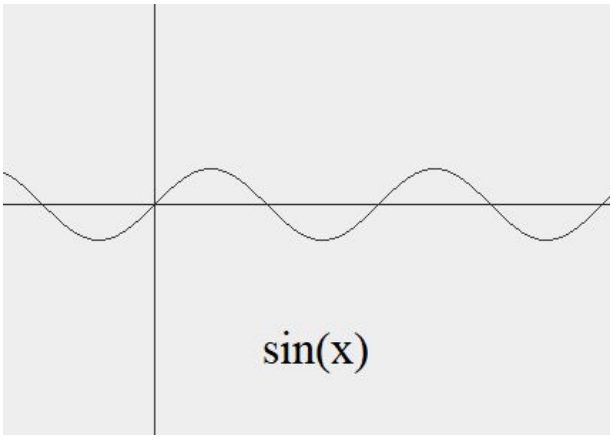
Pro samotné vykreslení funkce používám třídu Polygon z knihovny awt (java.awt.Polygon), neboť ta umožňuje konstrukci křivky na základě souřadnic v dvourozměrné soustavě souřadnic. Pro každou funkci pak postupuji po ose x a dopočítávám bod y, na základně typu funkce a jejích čtyřech komponentů.

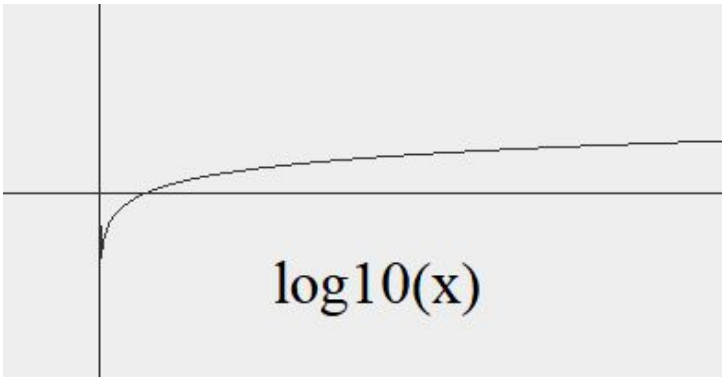
Naprosto klíčový je zde systém škálování funkce, jelikož jsou souřadnice v polygonu uloženy ve formě integerů. To je ale u značného počtu funkcí problematické, neboť třeba jenom takový sinus nabývá na ose y jenom tří celých čísel (-1,0,1) - což znamená, že se křivka tvoří jenom mezi těmito třemi body na ose y.





Řešení, které jsem na použil bylo zmíněné “škálování” funkce, tedy vlastně “roztažení” funkce na ose y, abychom na ni mohli postupovat po menších a přesnějších intervalech. Důležité je zároveň pracovat s osou x, aby funkce potom nebyla mnohem vyšší, než by měla být. Proto se musí body na ose y počítat s již škálováným x, v mém případě jsem po testování použil škálu 25. Díky tomuto se všechny funkce nejen zpřesní, ale hlavně se “vyhladí”.





Práce s čísly

Zpracovávání vstupu

Správné zpracování vstupu je především v části s rovnicemi naprosto klíčové. Prvním problémem je samozřejmě ukládání a "poskládání" čísel a neznámých, neboť čísla se neukládají jako celá čísla, nýbrž jenom jako číslovky. Samotné složení čísla z číslovek je triviální záležitostí, zajímavý problém je spíše kdy tuto operaci provést a v jakém formátu čísla ukládat. Jako přímočarým řešením se zdá býti provést tyto operace po skončení vstupu a čísla ukládat jako integery. To je ovšem problematické z toho důvodu, že potom není možné rozumně pracovat v jedné datové struktuře jak s čísly, tak se znaménky a neznámými. Jedna možnost nápravy tohoto problému je tedy neměnit čísla rovnou, ale změnit je, až když je to třeba. To ale zaprve není moc praktické z hlediska přehlednosti kódu, ani to není příliš efektivní.
Proto je dobré namísto integrů ukládat čísla (a celý zbytek rovnice) ve formě stringů. To sice ztěžuje samotné početní operace, ale velmi to usnadňuje, jak práci s polem, ve kterém je funkce uložena, tak reprezentaci např. desetinných čísel.

Možnosti zápisu

Zajímavou část správné analýzy rovnic, tvoří předvídání různých matematických zápisů stejné věci a vyrovnávání se s nimi. Kupříkladu výraz $4 \times x$, se dá také napsat jako 4x, druhá verze je dokonce běžnější a proto je potřeba se s ní vyrovnat. Zde jsem ale musel zabudovat další limitaci do mého programu, neboť ačkoliv je zápis a x4, také teoreticky možný, je poměrně obtížné s ním pracovat a většinou se takto matematicky nezapisuje, proto takové zápisy nejsou brány jako legitimní v mém programu.
Mezi další zajímavé matematické výkyvy v zápisech patří například fakt, že zatímco pokud výraz začíná záporným členem, mínus před něj samozřejmě píšeme, před kladný člen se na začátku nepíše. To se zdá býti banalitou, ale v programu, který je založen na vyhodnocování členů rovnice z velké části podle jejich znamének je toto důležitá výjimka.

Závěr

Je na první pohled zřejmé, že se mi nepodařilo do projektu zakomponovat všechny věci ze zadání. U něčeho to bylo naprosto vědomě, jako třeba u ukázání postupu řešení. Ačkoliv jsem se o to ze začátku pokoušel (kalkulačka lineárních a kvadratických rovnic je napsána tak trochu s touto funkcionalitou na mysli)., zjistil jsem, že to zkrátka není v mých silách. Stejně třeba u komplexních čísel, které jsem sice dostal do poměrně dobrého stavu, ale bohužel jenom poměrně, takže jsem je nakonec z finálního výsledku vyškrtl. U jiných menších věcí to bylo spíše nevědomě, jako třeba u goniometrických rovnic a nerovnic, na které jsem úplně zapomněl.

Na druhou stranu si myslím, že hlavní body zadání - a to rovnice lineární, kvadratické, s více neznámými, s vyššími mocninami a vykreslování grafů, jsem splnil a myslím, že u některých jsem přišel i s poměrně elegantní řešeními.

Pokud bych měl zmínit jeden přínos tohoto projektu pro mě, tak by to určitě byla schopnost rozumného rozdělování a nakládání s časem. Jelikož je toto poměrně velký projekt, bylo důležité v určitých bodech říci, že i když třeba ještě nejsem s něčím naprosto spokojen, je čas se zaměřit na věc druhou. Zvláště pak u věcí, jako třeba grafy, nebo funkce, které se vlastně dají vylepšovat donekonečna.

Celkově jsem se svým ročníkovým projektem spokojen, nezastírám jeho omezení a mouchy, ale nestydím se za ně.

Zdroje a použitá literatura

Determinanty a jejich výpočet: - https://mathworld.wolfram.com/Determinant.html
- https://matematika.cz/determinanty
Cramerovo pravidlo - https://cs.wikipedia.org/wiki/Cramerovo_pravidlo
- https://matematika.cz/cramerovo-pravidlo
Metoda tečen - https://cs.wikipedia.org/wiki/Metoda_te%C4%8Den
Metoda sečen - https://cs.wikipedia.org/wiki/Metoda_se%C4%8Den
Všechny obrázky a ikony v programu jsou vytvořeny mnou.