**СУ “Св. Климент Охридски”,**

**ФМИ – Софтуерно инженерство**

**Курсов проект по Структури от данни и алгоритми**

**MiniMathematica**

Радка Георгиева Дичева, Факултетен № 61666

Съдържание

[1. Въведение 2](#_Toc409876355)

[2. Описание на приложените алгоритми 2](#_Toc409876356)

[3. Описание на програмния код 4](#_Toc409876357)

[4. Използвани технологии 4](#_Toc409876358)

[5. Инсталация и настройки 5](#_Toc409876359)

[6. Приложимост на проекта 5](#_Toc409876360)

# Въведение

Курсовият проект по Структури от данни и алгоритми, който си избрах и написах е MiniMathematica или много прост и елементарен калкулатор. Този калкулатор поддържа основните аритметични операции - събиране, изваждане, умножение, деление, а също така и отчита въвеждането на отрицателни числа. Друго, което може да пресмята този калкулатор е натурален логаритъм, n-ти корен и повдигане на степен. Работещият с калкулатора може да използва константите пи (π) и Неперово число (e), а също така и тригонометричните функции – синус, косинус, тангенс и котангенс. Всъщност това бяха изискванията в условието на курсовия проект и се стремях да изпълня тях.

# Описание на приложените алгоритми

В условието на курсовия проект освен самото задание какво да прави програмата, примерния вход и изход имаше и предложени алгоритми, които да бъдат използвани. Затова реших да се придържам към съвета и избрах да прочета, науча и имплементирам посочените алгоритми. А именно “Shunting-yard” алгоритъм и Reverse Polish Notation или обратен полски запис. Изключително голяма помощ се оказаха няколко клипчета обясняващи алгоритъма и нотацията, както и любимият на всички Google.

Рецепта за “Shunting-yard” алгоритъм:

Докато има елементи (tokens) за четене:

Четене на елемент (token).

Ако елементът (token) е число, то се добавя на опашката.

Ако елементът (token) е функция, то се добавя на стека.

Ако елементът (token) е разделител в аргумент на функция (например запетайка):

Докато елементът на върха на стека стане лява (отваряща) скоба, се махат оператори от стека и се добавят в опашката. Ако не се срещне лява (отваряща) скоба, или разделителят не е бил на мястото си, или скобите са несъответстващи.

Ако елементът (token) е оператор o1, тогава:

Докато друг елемент (token) - оператор o2 е на върха на стека, и приоритета на o1 е по-малък или равен на този на o2 ,тогава o2 се маха от стека и се добавя на опашката.

o1 се добавя на стека.

Ако елементът (token) е лява (отваряща) скоба, тогава се добавя на стека.

Ако елементът (token) е дясна (затваряща) скоба:

Докато елементът (token) на върха на стека стане лява (отваряща) скоба, махане на оператори от стека и добавянето им на опашката.

Махане на лявата (отваряща) скоба от стека, но без да се добавя на опашката.

Ако елементът (token) на върха на стека е функция, махане от стека и добавяне на опашката.

Ако няма лява (отваряща) скоба в стека, значи има несъответстващи скоби.

Когато няма повече елементи (tokens) за четене:

Докато има оператори на стека:

Ако операторът на върха на стека е скоба значи има несъответстващи си скоби.

Махане на операторите от стека и добавянето им на опашката.

Край.

Опашката съдържа резултата, който ни е нужен.

Рецепта за обратен полски запис:

Докато остават входни елементи (tokens):

Четене на следващия елемент (token) от входа.

Ако елементът (token) е стойност:

Добавяне на стека.

Ако не, елементът (token) е оператор (или функция):

Знае се кой оператор колко аргумента изисква.

Ако има по-малко стойности на стека, то:

Грешка, няма достатъчно стойности в израза на потребителя.

Иначе, махане на нужните стойности от стека.

Извършване на дейност със стойностите и оператора.

Добавяне на резултата на стека.

Ако има само една стойност на стека:

Тази стойност е резултатът от пресмятането.

Ако не, има повече стойности в стека:

Грешка, потребителския вход има прекалено много стойности.

Край.

# Описание на програмния код

Програмата ми е разделена на два класа. Единият съдържа самия код на курсовия проект, докато другият е класът Test, където и от името му става ясно, че се използва за да се разбере правилно ли функционира програмата и прави ли това което се изисква. Този клас съдържа main метода, в който е кодът, позволяващ на потребителят да въведе свой вход и също така се извикват останалите методи на програмата, които имплементират гореспоменатите алгоритми и се грижат за вярното пресмятане на потребителския вход, ако разбира се потребителят е въвел коректни данни.

По-големият и сложен клас е Project, където от името му става ясно, че тук се съдържа всъщност същинската програма или имплементацията на самите алгоритми. В класа има няколко метода, всеки от които играе своята роля. Но първо се намират имплементирани ArrayList-ове от String-ове, които пазят, функциите (логаритъм, корен квадратен, степенуване, синус, косинус, тангенс, котангенс), аритметичните оператори (плюс, минус, умножение, деление) и резултат. Също така имаме изброим тип за операторите както и HashMap, които служат на метода isHigherPrecedence, който приема два параметъра от тип String и връща резултат от булев тип. Този метод определя приоритета на операторите (както знаем умножението и делението са с по-висок приоритет от събирането и изваждането, и трябва да се изпълняват първи). Следва методът filler, който нито приема параметри нито връща резултат. Той, както е ясно от името, пълни гореспоменатите ArrayList-ове със съответните String-ове.

Идва ред на по-сложните методи, първият от които е getTokens. Този метод приема параметър от тип String и връща като резултат List<String>. Функционалността на този метод е да вземе потребителския вход, който е от тип String, и да го разбие до елементи (tokens), така че всяко число, оператор или функция да е отделен елемент (token) и тях да върне като резултат. Следващият метод е fromInfixToReversePolishNotation, който приема като параметър List<String> и връща резултат Queue<String>. Функционалността на този метод е да вземе вече разбития на елементи (tokens) потребителски вход, да приложи гореописания “Shunting-yard” алгоритъм и да върне опашката, съдържаща израза в Postfix запис. Последният метод е reversePolishNotation, който приема като параметър Queue<String> и връща резултат от тип double. Функционалността на този метод е да вземе опашката, съдържаща израза в Postfix запис, да приложи гореописаният алгоритъм за обратен полски запис, за да върне резултата, който всъщност представлява и резултата на израза, въведен от потребителя.

# Използвани технологии

Единствената използвана технология е IDE-то, което е необходимост за всеки програмист.

# Инсталация и настройки

Потребителят трябва да има компютър. На този компютър е необходимо да е инсталирано някое IDE (например Eclipse, NetBeans), поддържащо Java.

# Приложимост на проекта

Основното приложение на тази програма е като курсов проект по Структури от данни и алгоритми, защото вече съществуват множество програми и калкулатори, поддържащи много по-голям набор от функционалности, които са се доказали в практиката и се използват отдавна. Все пак курсовият проект спомогна за научаването на алгоритмите, за имплементирането им и за разбирането как работят и какво представляват те.