

Detaily implementace řešení výpočtu matematických výrazů

Roland Schulz

February 11, 2020

- 1 Řetězcové zpracování výrazu
 - Stromové zpracování výrazu
 - Tokenizace infixové notace
 - Parsování infixové notace
 - Precedentní umísťování do AST - názorně
 - Obecně potřebné struktury pro řešení řetězcového zpracování
- 2 Přímý výpočet pomocí průběžných výpočtů

Stromové zpracování výrazu (Expression calculator)

- Lexikální analýza (Tokenizace)
 - Gramatika a prostředí
- Syntaktická analýza (Parsování)
 - Zásobník
 - Strom
- Evaluace AST pomocí DFS Post-order (LRN)

- Regex metoda
- Využití označování z GUI
 - Samostatné značkovací pole
 - Využití nezobrazených znaků(whitespace, separator)

- Tabulka symbolů (operandy, fce, konstanty, proměnné)
- Precedenční tabulka početních operací

Precedence:

Pole množin označující precedenci operátorů nebo typů lexémů (tokenů) kde pozice označuje precedenci. Motivace umístit přednostní operátory a tokeny co nejnižší pro korektní aplikaci DFS.

(,)	+, -	<i>negative</i>	*, /	<i>pow</i>	<i>fact</i>	<i>const, fce</i>
1	2	3	4	5	6	<i>max</i>

Algoritmus výpočtu podle Shunting yard algoritmu

zjednodušeně infix to RPN: eval = interpretace stacku RPN

Vstup: $3 + 4 * 2 / (1 - 5) ^ 2 ^ 3$

Token	Akce	Výstup (v RPN)	Zásobník operátorů	Poznámky
3	Přidej token do výstupu	3		
+	Přidej token na zásobník	3	+	
4	Přidej token do výstupu	3 4	+	
*	Přidej token na zásobník	3 4	+ *	* má větší prioritu než +
2	Přidej token do výstupu	3 4 2	+ *	
/	Vyjmi ze zásobníku a vypiš	3 4 2 *	+	/ a * mají stejnou prioritu
	Přidej token na zásobník	3 4 2 *	+ /	/ má větší prioritu než +
(Přidej token na zásobník	3 4 2 *	+ / (
1	Přidej token do výstupu	3 4 2 * 1	+ / (
-	Přidej token na zásobník	3 4 2 * 1	+ / (-	
5	Přidej token do výstupu	3 4 2 * 1 5	+ / (-	
)	Vyjmi ze zásobníku a vypiš	3 4 2 * 1 5 -	+ / (Opakuj dokud nenajdeš "("
	Vyjmi ze zásobníku	3 4 2 * 1 5 -	+ /	Zahod' nalezenou závorku
^	Přidej token na zásobník	3 4 2 * 1 5 -	+ / ^	^ má větší prioritu než /
2	Přidej token do výstupu	3 4 2 * 1 5 - 2	+ / ^	
^	Přidej token na zásobník	3 4 2 * 1 5 - 2	+ / ^ ^	^ je vyhodnoceno zprava doleva
3	Přidej token do výstupu	3 4 2 * 1 5 - 2 3	+ / ^ ^	
konec	Vypiš zbytek zásobníku	3 4 2 * 1 5 - 2 3 ^ ^ / +		

Obecně potřebné struktury pro řešení řetězcového zpracování

- Strom, při použití infix to ast algoritmu
- Stack, při použití Shunting yard algoritmu

Přímý výpočet (Immediate execution calculator)

2 registry:

1. pro předešlý výsledek - ANS
2. pro uchování zadaného čísla uživatelem

Operace budou zakotveny do tlačítek GUI a budou při klikání na operandy počítat průběžné výsledky.

Veškerá logika programu se přesouvá do GUI frontendu.

Při implementaci více funkcí/konstant univerza by mohlo dojít k znepráhlednění kódu a ke zhoršení přehlednosti frontendu aplikace.

Reference:

infix to AST: <https://www.rhyscitlema.com/algorithms/expression-parsing-algorithm/>