Aritmetické výrazy I.

Základy algoritmizace – 12. cvičení

Zuzana Petříčková

21. května 2020

Aritmetické výrazy (zjednodušeně)

Budeme uvažovat zjednodušené aritmetické výrazy. Tvoří je:

- číselné konstanty, proměnné (reálná čísla)
  např. -0.5, 8, x, x1
- binární operátory: +, -, *, /
- unární minus $\sim$
- závorky (,)

abychom si zjednodušili zpracování výrazu, budeme unární minus značit jako ,,$\sim$" namísto ,,$-$" (znak ,,$-$" bude značit binární operátor minus)

Příklady:

- $6 * (\sim x) + 1.7$
- $((4 - x) + y)/ \sim (x * 2)$

Aritmetické výrazy (zjednodušeně)

Operátory a závorky

- asociativita operátoru
  - zleva: $a + b + c$ je totéž jako $((a + b) + c)$
    zprava: $a + b + c$ je totéž jako $(a + (b + c))$
  - my budeme uvažovat u operátoru $+, -, *, /$ **asociativitu zleva**
- priorita: čím nižší číslo, tím operátor více váže:

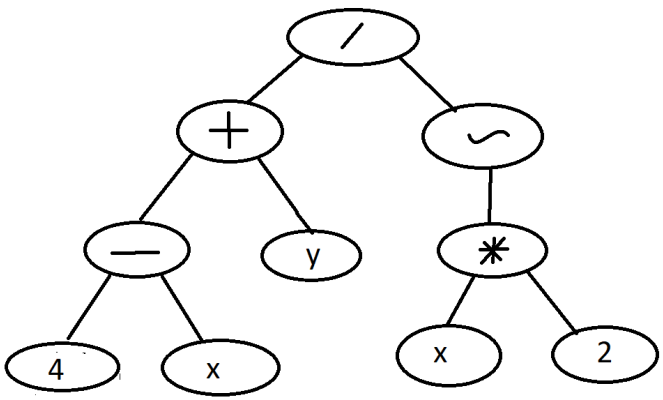
priorita	operátor
2	$\sim$
5	$* /$
2	$+ -$

  - $a * b + c$ je totéž jako $((a * b) + c)$
  - $\sim a + b - c$ je totéž jako $(((\sim a) + b) - c)$

Aritmetické výrazy (zjednodušeně)

Reprezentace aritmetických výrazů v programu:

- **textový řetězec** v infixové, prefixové (tzv. polské, PN) nebo postfixové (tzv. reverzní polské, RPN) notaci
- **(binární) strom**
  - vnitřní vrcholy ... operátory
  - listy ... číselné konstanty nebo proměnné
  - podstrom = podvýraz

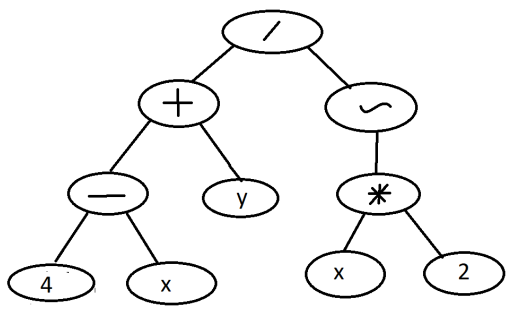


Aritmetické výrazy (zjednodušeně)

Reprezentace aritmetických výrazů pomocí textového řetězce:

- **infixová notace:**
  $((4 - x) + y)/ \sim (x * 2)$
- **prefixová notace:**
  / + - 4 x y ~ * x 2
- **postfixová notace:**
  4 x - y + x 2 * ~ /

- **strom pro uvedený výraz:**



Aritmetické výrazy (zjednodušeně)

Infixová notace

- **pro binární operátor:** nejprve levý operand, pak operátor, pak pravý operand, př. „$a + b$"
- **pro unární operátor:** nejprve operátor, pak operand, př. „$\sim b$"
- **příklad:** $((4 - x) + y) / \sim (x * 2)$
- potřebujeme závorky
- notace nejvíce přehledná pro člověka, ale nejhůře přehledná pro program (v programu se s ní pracuje nejhůře)

Aritmetické výrazy (zjednodušeně)

Prefixová (polská) notace (PN)

- **pro binární operátor:** nejprve operátor, pak levý operand, pak pravý operand, př. „$+\,a\,b$"
- **pro unární operátor:** nejprve operátor, pak operand, př. „$\sim\,b$"
- **příklad:** $/\;+\;-\,4\,x\,y\;\sim\;*\,x\,2$
- nepotřebujeme závorky (v případě, že pro unární minus máme „$\sim$" namísto „$-$")
  „$/\;+\;-\,4\,x\,y\;\sim\;*\,x\,2$" je totéž jako
  „$(/(+(-\,4\,x)\,y)(\sim\,(*\,x\,2)))$"
- notace nepřehledná pro člověka, ale výrazně snadnější zpracování programem

Aritmetické výrazy (zjednodušeně)

Postfixová (reverzní polská) notace (RPN)

- **pro binární operátor:** nejprve levý operand, pak pravý operand, nakonec operátor př. „$a\ b\ +$"
- **pro unární operátor:** nejprve operand, pak operátor, př. „$b\ \sim$"
- **příklad:** 4 x - y + x 2 * ~ /
- nepotřebujeme závorky (v případě, že pro unární minus máme „$\sim$" namísto „$-$")
  „4 x - y + x 2 * ~ /" je totéž jako
  „$((4\,x\,-)\,y\,+)((x\,2\,*)\ \sim)/$"
- notace nepřehledná pro člověka, ale super jednoduché zpracování programem

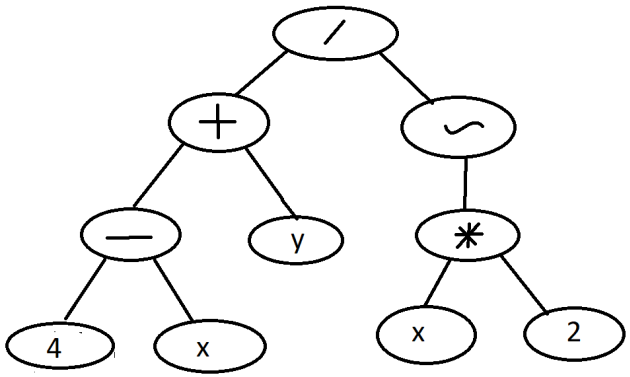
Aritmetické výrazy (zjednodušeně)

Typické operace s aritmetickými výrazy v programu

- převod mezi různými reprezentacemi
- vyhodnocení výrazu
- aritmetická kalkulačka

Binární strom aritmetického výrazu

- vnitřní vrcholy ... operátory
- listy ... číselné konstanty nebo proměnné
- podstrom = podvýraz



Vyhodnocení aritmetického výrazu uloženého ve stromě:

- to už umíme :-)
- metoda rozděl a panuj, viz metoda **zpracuj (POSTORDER)** pro BVS (probírali jsme na začátku semestru)
- lze řešit rekurzivně nebo pomocí zásobníku

Binární strom aritmetického výrazu

Převod stromu na řetězec v infixové, prefixové nebo postfixové notaci

- to už také umíme (použijeme stejný postup) :-)
- výpis stromu v pořadí INORDER (pro infixovou notaci), PREORDER (pro prefixovou notaci), popř. POSTORDER (pro postfixovou notaci)
- lze řešit rekurzivně nebo pomocí zásobníku (probírali jsme pro BVS na začátku semestru)

Časová složitost operací

- **vyhodnocení, převod na infixovou, prefixovou nebo postfixovou notaci:** $T(n) = O(n)$

Aritmetické výrazy (zjednodušeně)

- vyhodnotit nebo vypsat výraz uložený ve stromě je snadné
- ale: chtít po uživateli, aby vkládal do programu výraz jako strom není pohodlné

Jak ale vyhodnotit aritmetický výraz zadaný ve formě textového řetězce?

1. řetězec převedeme na strom a ten následně vyhodnotíme
2. vyhodnotíme výraz přímo (bez nutnosti převodu na strom)

- začneme s případem nejjednodušším pro zpracování: **řetězec v postfixové notaci**

Postfixová notace

Ukážeme si:

1. převod řetězce v postfixové notaci na strom
2. vyhodnocení aritmetického výrazu přímo (bez nutnosti převodu na strom)

POSTFIX $\rightarrow$ STROM

- nejprve implementujeme pomocnou funkci (popř. iterátor), která vrátí následující token řetězce v pořadí zleva doprava (token: operátor nebo operand)
- následně použijeme zásobník (pro uchování podstromů postupně vytvářeného stromu)

Postfixová notace (POSTFIX → STROM)

Základní myšlenka:

- výraz budeme procházet zleva doprava, postupně budeme odspoda vytvářet strom, mezivýsledky (podstromy) budeme ukládat na zásobník
- výraz je v postfixové notaci, nejprve proto přijdou operandy a pak teprve příslušný operátor

Postfixová notace (POSTFIX → STROM)

Použijeme: zásobník na vrcholy stromu (podstromy)

- Bereme tokeny jeden po druhém:
  - číslo, proměnná → vytvoříme odpovídající vrchol a vložíme ho na zásobník
  - binární operátor →
    1. na vrchu zásobníku jsou jeho operandy (na vrchu pravý, pod ním levý), vyjmeme je
    2. vytvoříme vrchol odpovídající operátoru, připojíme k němu operandy (jako syny) a vložíme ho na zásobník
  - unární operátor →
    1. na vrchu zásobníku je jeho operand, vyjmeme ho
    2. vytvoříme vrchol odpovídající operátoru, připojíme k němu operand (jako jedinného syna) a vložíme ho na zásobník
- na konci bude na zásobníku jedinný prvek = výsledný strom

Postfixová notace (POSTFIX → vyhodnoť přímo)

- algoritmus bude hodně podobný, jen místo vytváření stromu rovnou počítáme výsledek

Použijeme: zásobník na čísla

- Bereme tokeny jeden po druhém:
  - číslo → vložíme ho na zásobník
  - proměnná → vyčíslíme ji a číselnou hodnotu vložíme na zásobník
  - binární operátor →
    1. na vrchu zásobníku jsou jeho operandy (na vrchu pravý, pod ním levý), vyjmeme je
    2. spočteme výsledek operace a vložíme ho na zásobník
  - unární operátor →
    1. na vrchu zásobníku je jeho operand, vyjmeme ho
    2. spočteme výsledek operace a vložíme ho na zásobník
- na konci bude na zásobníku jedinný prvek = výsledek

Postfixová notace

Časová a prostorová složitost (POSTFIX $\rightarrow$ STROM, POSTFIX $\rightarrow$ vyhodnoť přímo)

- čas $T(n) = O(n)$
- prostor $S(n) = O(n)$ (zásobník)

Prefixová notace

Opět si ukážeme:

1. převod řetězce v prefixové notaci na strom
2. vyhodnocení aritmetického výrazu přímo (bez nutnosti převodu na strom)

PREFIX $\rightarrow$ STROM

- algoritmus bude o něco složitější než pro POSTFIX
- použijeme opět zásobník (pro uchování podstromů postupně vytvářeného stromu)
  - tentokrát nejprve přijde operátor a pak teprve jeho operandy

Prefixová notace (PREFIX → STROM)

Použijeme: zásobník na vrcholy stromu (podstromy)
Bereme tokeny jeden po druhém:

- operátor → vytvoříme odpovídající vrchol a vložíme ho na zásobník
- číslo, proměnná →
  - vytvoříme odpovídající vrchol v
  - **Cyklus:** podíváme se na vrchol zásobníku:

1. pokud je zásobník prázdný nebo je na vrchu zásobníku list odpovídající binárnímu operátoru, podstrom v vložíme na zásobník a ukončíme cyklus
2. pokud je na vrchu zásobníku list odpovídající unárnímu operátoru →
   - vyjmeme ho, připojíme k němu v jako jedinného syna a opakujeme pro výsledný podstrom (označíme ho jako v)
3. jinak (na vrchu zásobníku je větší podtrom) →
   - vyjmeme ze zásobníku nejprve podstrom u (levý operand) a pak list odpovídající binárnímu operátoru
   - připojíme k operátoru operandy (v bude pravým synem, u pravým) a opakujeme pro výsledný podstrom (označíme ho jako v)

na konci bude na zásobníku jedinný prvek = výsledný strom

Prefixová notace (PREFIX → vyhodnoť přímo)

- algoritmus bude hodně podobný jako algoritmus pro sestrojení stromu, jen místo vytváření stromu rovnou počítáme výsledek
- u POSTFIXu jsme si vystačili se zásobníkem na čísla
- tentokrát budeme pořebovat zásobník na tokeny (čísla nebo operátory)

Prefixová notace (PREFIX → vyhodnoť přímo)

Použijeme: zásobník na tokeny (čísla nebo operátory)

Bereme tokeny jeden po druhém:

- operátor → vložíme ho na zásobník
- číslo, proměnná →
  - proměnnou vyčíslíme → číslo v
  - **Cyklus:** podíváme se na vrchol zásobníku:

1. pokud je zásobník prázdný nebo je na vrchu zásobníku binární operátor, číslo v vložíme na zásobník a ukončíme cyklus
2. pokud je na vrchu zásobníku unární operátor →
   - vyjmeme ho, spočítáme výsledek operace, označíme ho jako v a opakujeme
3. jinak (na vrchu zásobníku je číslo) →
   - vyjmeme ze zásobníku nejprve číslo (levý operand) a pak binární operátor, v bude pravý operand
   - spočítáme výsledek operace, výsledek označíme jako v a opakujeme

na konci bude na zásobníku jedinný prvek = výsledek

Prefixová notace

Časová a prostorová složitost (PREFIX $\rightarrow$ STROM, PREFIX $\rightarrow$ vyhodnoť přímo)

- čas $T(n) = O(n)$
- prostor $S(n) = O(n)$ (zásobník)

Infixová notace

- pro uživatele je postfixová a prefixová notace poměrně nepřehledná, proto je dobré mu umožnit, aby mohl aritmetický výraz zadávat jako řetězec v infixové notaci
- zde bude situace o kus složitější než pro postfix či prefix, proto algoritmy necháme až na příští cvičení

Ukážeme si:

1. převod řetězce v infixové notaci na řetězec v postfixové notaci
2. převod řetězce v infixové notaci na strom
3. vyhodnocení aritmetického výrazu přímo (bez nutnosti převodu na postfixovou notaci nebo strom)