# Aritmetické výrazy II.

## Základy algoritmizace – 13. cvičení

Zuzana Petříčková

12. května 2020

# Aritmetické výrazy II

- dnes dokončíme látku z minulého cvičení

**Uvažujeme 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)

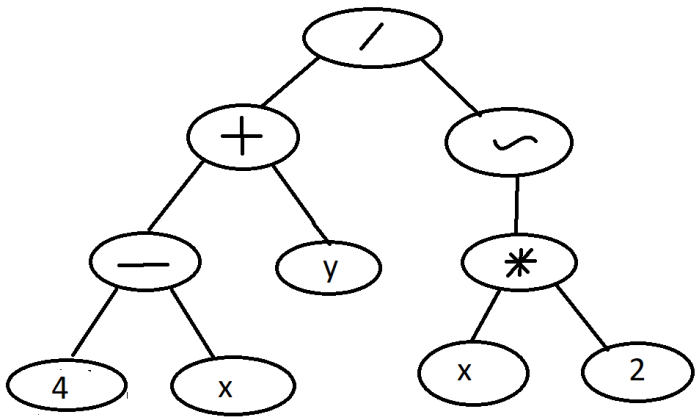
**Příklady:**

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

# Aritmetické výrazy (zjednodušeně)

**Reprezentace aritmetických výrazů v programu:**

- **(binární) strom**
  - vnitřní vrcholy ... operátory
  - listy ... číselné konstanty nebo proměnné
  - podstrom = podvýraz
- **textový řetězec**
  - **infixová notace:** $((4 - x) + y) / \sim (x * 2)$
  - **prefixová notace:** / + - 4 x y ~ * x 2
  - **postfixová notace:** 4 x - y + x 2 * ~ /



# Aritmetické výrazy (zjednodušeně)

**Minule jsme se seznámili s typickými operacemi s aritmetickými výrazy v programu**

- převod mezi různými reprezentacemi:
  1. STROM $\rightarrow$ POSTFIX, PREFIX, INFIX
  2. POSTFIX $\rightarrow$ STROM
  3. PREFIX $\rightarrow$ STROM
- přímé vyhodnocení výrazu zadaného jako
  1. STROM
  2. POSTFIX
  3. PREFIX

všechny minule probrané algoritmy měly lineární časovou i prostorovou složitost, $T(n) = S(n) = O(n)$

# Infixová notace

- tentokrát se zaměříme na infixovou notaci

**Syntaxe**

- **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)

# 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 si algoritmus více rozkouskujeme

**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)

# Infixová notace (INFIX → POSTFIX)

- nejprve si ukážeme převod výrazu z infixové na postfixovou notaci
- použijeme zásobník pro uchování tokenů (operátorů a levých závorek)

**Značení:**

- LZ ... levá závorka
- PZ ... pravá závorka

# Infixová notace (INFIX → POSTFIX)

**Použijeme:** zásobník na tokeny (operátory a LZ)

- Bereme tokeny jeden po druhém:
  - číslo, proměnná → vypíšeme ji
  - levá závorka (LZ) → vložíme ji na zásobník
  - pravá závorka (PZ) → vyjmeme ze zásobníku všechny tokeny až po LZ a vypíšeme je, vyjmeme i LZ (tu nevypisujeme)
  - operátor o →
    - **cyklus:** dokud je na vrchu zásobníku operátor se stejnou nebo vyšší prioritou než o → vyjmeme ho ze zásobníku a vypíšeme
    - vložíme operátor o na zásobník
  - konec výrazu → vyjmeme ze zásobníku všechny tokeny (operátory) a vypíšeme je (LZ ignorujeme)

# Infixová notace (INFIX → STROM)

- už umíme převést INFIX na POSTFIX a pak POSTFIX na STROM
  - každý z algoritmů využíval zásobník
- nyní oba postupy spojíme dohromady
- **použijeme dva zásobníky:**
  - Z na tokeny (operátory a LZ)
  - $Z_h$ pro uchování podstromů postupně vytvářeného stromu

# Infixová notace (INFIX → STROM)

**Použijeme dva zásobníky:** $Z$ na tokeny (operátory a LZ) a $Z_h$ na podstromy
Bereme tokeny jeden po druhém:

- číslo, proměnná → vytvoříme odpovídající vrchol a vložíme ho na $Z_h$
- levá závorka (LZ) → vložíme ji na $Z$
- pravá závorka (PZ) →
  - **cyklus:** dokud je na vrchu $Z$ operátor (ne LZ), **zpracujeme ho**
  - ze $Z$ vyjmeme i LZ
- operátor o →
  - **cyklus:** dokud je na vrchu $Z$ operátor se stejnou nebo vyšší prioritou než o, **zpracujeme ho**
  - vložíme operátor o na $Z$
- konec výrazu → **zpracujeme** všechny zbylé operátory v $Z$ (LZ ignorujeme)

Výsledný strom je v $Z_h$ jako jedinný prvek

**Zpracování operátoru:**

- vyjmeme operátor ze $Z$, vyjmeme jeho operandy (jeden nebo dva) z vrchu $Z_h$, vytvoříme vrchol odpovídající operátoru, připojíme k němu jeho operandy jako syny a výsledný podtrom vložíme na $Z_h$

# Infixová notace (INFIX → vyhodnoť přímo)

- algoritmus bude hodně podobný jako předchozí, jen místo vytváření stromu rovnou počítáme výsledek
- **použijeme dva zásobníky:**
  - Z na tokeny (operátory a LZ)
  - $Z_h$ na čísla (mezivýsledky)

# Infixová notace (INFIX → vyhodnoť přímo)

**Použijeme dva zásobníky:** $Z$ na tokeny (operátory a LZ) a $Z_h$ na čísla
Bereme tokeny jeden po druhém:

- číslo → vložíme ho na $Z_h$
- proměnná → vyčíslíme ji a číselnou hodnotu vložíme na $Z_h$
- levá závorka (LZ) → vložíme ji na $Z$
- pravá závorka (PZ) →
  - **cyklus:** dokud je na vrchu $Z$ operátor (ne LZ), **zpracujeme ho**
  - ze $Z$ vyjmeme i LZ
- operátor o →
  - **cyklus:** dokud je na vrchu $Z$ operátor se stejnou nebo vyšší prioritou než o, **zpracujeme ho**
  - vložíme operátor o na $Z$
- konec výrazu → **zpracujeme** všechny zbylé operátory v $Z$ (LZ ignorujeme)

Výsledek je v $Z_h$ jako jedinný prvek

**Zpracování operátoru:**

- vyjmeme operátor ze $Z$, vyjmeme jeho operandy (jeden nebo dva) ze $Z_h$, spočítáme výsledek operace a vložíme ho na $Z_h$

# Infixová notace

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

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

# Další náměty k samostudiu

**Aritmetické výrazy ve skriptech**

- kapitola 4.3 **Syntaktická analýza**, **Vyhodnocení aritmetického výrazu**, str. 113-116 (řešení pomocí dvojité rekurze)
  - doporučuji nastudovat práci s tokeny (stejný postup je použit v příkladu za domácí úkol)
- kapitola 6.3 **Zpracování aritmetického výrazu** (str. 200-204)

# Další náměty k samostudiu

**Kapitoly ve skriptech, které jsme na cvičení neprobírali**
(ale stálo by za to si je nastudovat nejen ke zkoušce)

- kapitola 2: iterátor, tabulka symbolů (hešovací tabulka), B-strom
- některé třídící algoritmy v kap. 5
- kapitola 6: Vyvážené a AVL stromy
- kapitola 7: seminumerické algoritmy
- kapitola 8: další algoritmy: tranzitivní uzávěr grafu, výpočet hodnoty polynomu, generování permutací, test prvočíselnosti, efektivní násobení matic