# Algoritmi i strukture podataka

- predavanja -

6. Stog

### Stog

- struktura podataka kod koje se posljednji pohranjeni podatak prvi uzima u obradu (Last In First Out – LIFO)
- potrebne operacije:
  - dodavanje (push) elemenata na vrh stoga
  - brisanje (pop) elemenata s vrha stoga
  - inicijalizacija praznog stoga
  - po potrebi
    - uvid u sadržaj vrha stoga (peek)
- pojedina operacija push ili pop zahtijeva jednako vremena bez obzira na broj pohranjenih podataka
- može se realizirati statičkom strukturom podataka

#### StackStatic.cpp

- najbrža realizacija: O(1)
- postoji mogućnost prepunjenja

### Stog

Realizacija dinamičkim poljem

#### StackDynamic.cpp

- Vjerojatnost prepunjenja vrlo mala
- Može se dogoditi O(N)

# Stog

Realizacija jednostruko povezanom listom

#### StackList.cpp

- Vjerojatnost prepunjenja vrlo mala
- Uvijek O(1)
- Primjer primjene stoga:
  - Evaluacija izraza napisanog u RPN

### Evaluacija izraza u postfix notaciji

■ izraz u *infix* notaciji:

može se pretvoriti u postfix notaciju (ili Reverse Polish Notation -RPN):

- kod RPN notacije operandi se pišu prije operatora, a operator se primjenjuje na operande koji se nalaze točno ispred njega
- prednost postfix notacije nemamo zagrade, jednostavna evaluacija izraza korištenjem stoga

### Ručna pretvorba infix → postfix

dodati sve implicitne zagrade u infix izraz:

$$8*(9+3)/16$$
 ((8\*(9+3))/16)

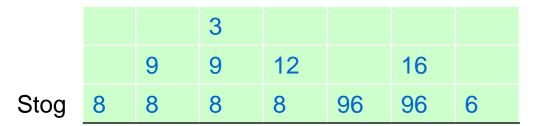
 krenuvši od unutrašnje zagrade, slijed operand – operator – operand zamijeniti u operand – operand – operator

$$((8*(9+3))/16)$$
  
 $((8*93+)/16)$   
 $((8*93+)/16)$   
 $(893+*/16)$   
 $(893+*/16)$   
 $(893+*/16)$   
 $(893+*/16)$ 

Strojni algoritam: Shunting Yard (kasnije, potreban red)

# Evaluacija izraza u postfix notaciji

- algoritam evaluacije izraza u postfix notaciji korištenjem stoga:
  - izraz se evaluira s lijeva na desno, token po token
  - ako je token broj, stavi se na stog (push)
  - ako je token operator, skinu se zadnja dva broja sa stoga (pop), na njih se primijeni taj operator, a rezultat se stavlja natrag na stog (push)
- Primjer:  $8*(9+3)/16=6 \rightarrow 893+*16/$ Ulaz 8 9 3 + \* 16/



RPN.cpp

#### Primjer: izravna evaluacija infix izraza

- Dva stoga:
  - stog operatora
  - stog vrijednosti
- Pojedinačna evaluacija:
  - uzmi operator sa stoga operatora
  - uzmi operande sa stoga vrijednosti
  - primijeni operator nad operandima
  - stavi rezultat na stog vrijednosti

#### Primjer: izravna evaluacija infix izraza

#### Algoritam:

- dok ima znakova na ulazu
  - ako slijedi vrijednost, stavi je na stog vrijednosti
  - ako slijedi otvorena zagrada, stavi je na stog operatora
  - ako slijedi zatvorena zagrada
    - dok na vrhu stoga operatora nije otvorena zagrada
      - obavi pojedinačnu evaluaciju
    - skini otvorenu zagradu sa stoga operanada
  - ako slijedi operator
    - dok ima operatora na stogu koji su većeg ili jednakog prioriteta
      - obavi pojedinačnu evaluaciju
    - stavi novi operator na stog operatora
- dok ima operatora na stogu
  - obavi pojedinačnu evaluaciju
- uzmi rezultat sa stoga vrijednosti

InfixEvaluation.cpp