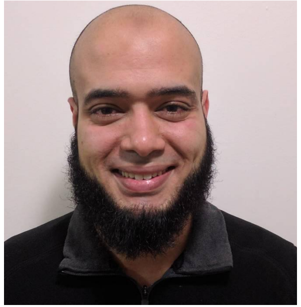
Data Structures

Infix, Postfix and Prefix Notation

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Infix/postfix/prefix notations

- This is not a data structure (sub)topic
- It is a popular application for using stacks
- In this lecture, we will understand these notations and how to evaluate
- Later: How to convert infix to postfix

Our regular math equations

- 1 + 2 ⇒ 3
- 1 + **2 * 5** ⇒ 1 + 10 ⇒ 11   (multiplication has higher priority)
  - As same as: 1 + (2 * 5)
- (**1 + 2**) * 5 ⇒ 3 * 5 ⇒ 15   : we used () to force a different order
- **12 / 3** * 10 / 2 ⇒ **4 * 10** / 2 ⇒ **40 / 2** ⇒ 20
  - Both / and * have same priority, so we evaluate left to right (LR Associativity)
- **1 + 2** - 5 + 9 ⇒ 7
  - Both + and - have the same priority, so we evaluate left to right
- So: {* /} are before {+ -}. If the same priority, do left to write
  - But () forces priority
- Confused? *Kindly review Programming operators topic: Precedence/Associativity*

Infix Notation

- Our regular math equations format is called **infix** notation
    - Infix = operators lie **between** the operands
    - <operand> <**operator**> <operand>
    - Example: 1 + 2 * 3   {* is between 2 and 3. + is between 1 and 2*3}
    - We just refreshed the evaluation precedence rules
- There are 2 other notations:
    - Postfix = operators is after the operands: 1 2 +
    - Prefix = operators is before the operands: + 1 2 +
    - **Why**? It is **easier** for the computer to **parse** them than the infix precedence/associativity

Let’s evaluate postfix expression

Tip *find num num* ***oper*** *and evaluate*

- 1 + 2 * 3 ⇒ 1 2 3 * +
  - 1 **2 3 *** + ⇒ 1 6 + ⇒ 7
- 1 + 2*3 -4 ⇒ 1 2 3 * + 4 -
  - 1 **2 3 *** + 4 - ⇒ **1 6 +** 4 - ⇒ **7** 4 - ⇒ 3
- (1+2)*(6-4) ⇒ 1 2 + 6 4 - *
  - **1 2 +** 6 4 - * ⇒ 3 **6 4 -** * ⇒ 3 **2** * ⇒ 6   [observe how we did not handle () priorities!
- 5 - 9 / (3*4 / 2) ⇒ 5 9 3 4 * 2 / / -
  - 5 9 **3 4 *** 2 / / - ⇒ 5 9 **12 2 /** / - ⇒ 5 **9 6 /** - ⇒ **5 1.5 -** ⇒ 3.5
- ((2+3)*4-(7-5))*(6+3) ⇒ 2 3 + 4 * 7 5 - - 6 3 + *
  - **2 3 +** 4 * 7 5 - - 6 3 + * ⇒ 5 4 * 7 5 - - 6 3 + * ⇒
  - 2 3 + 4 * **7 5 -** - 6 3 + * ⇒ 2 3 + 4 * 2 - 6 3 + * ⇒
  - 2 3 + 4 * 7 5 - - **6 3 +** * ⇒ 2 3 + 4 * 2 - 9 * ⇒

Prefix Evaluation

- Very similar logic: find **oper** num num and evaluate
- ((2+3)*4-(7-5))*(6+3) ⇒ * - * + 2 3 4 - 7 5 + 6 3
  - * - * **+ 2 3** 4 - 7 5 + 6 3
  - * - * 5 4 - 7 5 **+ 6 3**
  - * - * **5 4** - 7 5 9
  - * - 20 **- 7 5** 9 [- 7 5 = 7 - 5 = 2]
  - * **- 20 2** 9
  - * 18 9
  - **162**

Online Calculators

- You may google to find online calculator: Infix to Postfix/Prefix converter
  - But some might fail at tricky cases!
  - Tricky cases to try later on infix to prefix
    - 1-2+3 ⇒ +-123
    - 4^3^2 ⇒ ^4^32
- Example for site working fairly well
  - Step by step to postfix conversion / to prefix conversion

Think!

- Using a stack, simulate the **postfix evaluation** process
- Using a stack, how can we convert an infix to postfix?
    - Assume a string of single digit numbers and * + operators
        - E.g. 2 + 3 * 5

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”