Reverse Polish notation

> Infix notation: operators are in between their operands
>
> Postfix notation / Reverse Polish notation: Operators are written after operands
>
> Prefix notation / Polish notation: operators are written before operands

---

Why RPN can be used to evaluate expressions:

- Reverse polish notation provides and unambiguous method of representing an expression
- Reading from left to right
- Without need to use bracket
- With no need for rules of precedence

Explain how RPN is used by an interpreter to evaluate expression:

- Expressions are always evaluated from left to right
- Each operator uses two previous values on the stack
- Pushing and popping identifiers on a stack

Data structure to evaluate an expression in RPN

- Stack
    - The operands are popped from the stack in reverse order to how they were pushed
- Binary tree
    - Allows both infix and postfix to be evaluated (tree traversal)

Describe the main steps in the evaluation of a Reverse Polish Notation(RPN) expression using a stack

- Working from left to right in the expression
- If element is a number, push that number into stack
- If element is an operator, then pop the first two numbers on the stack
- … perform that operations on these two numbers
- PUSH result back into stack
- END once the last item in the expression has been dealt with

One advantage of the use of RPN is that evaluation of expressions does not require rules of precedence; Explain this statement

- Rules of precedence means different operators have different priorities; for example multiplication is done before addition
- In RPN, evaluation of expression is from left to right // operator are used in the sequence in which they were read
- No need for brackets // infix may require brackets

Write the Reverse Polish Notation for the following expressions

1. (A+B)*(C-D)

   - AB+
   - CD-*

   Full answer: AB+CD-*
2. ((-A/B)*4)/(C-D)

   - A-
   - B/4*
   - CD-/

   Full answer: A-B/4*CD-/

Evaluation of RPN

Evaluating an RPN expression

A stack can be used to evaluate an RPN expression. Let's consider the execution of the following RPN expression when x has the value 3 and y has the value 4:

```
x 2 * y 3 * + 6 /
```

The rules followed here are that the values are added to the stack in turn. The process is interrupted if the next item in the RPN expression is an operator. This causes the top two items to be popped from the stack. Then the operator is used to create a new value from these two and the new value is added to the stack. The process then continues. Figure 20.13 shows the successive contents of the stack with an indication of when an operator has been used. The intermediate states of the stack when two values have been popped are not shown.

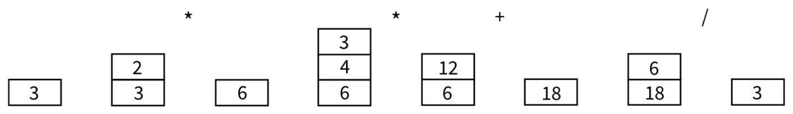

Figure 20.13 Evaluating a Reverse Polish expression using a stack