# Calculation algorithm: Infix to Postfix

A general expression can contain many parentheses and operations with different priorities. It is difficult to calculate such expressions if you do not use special methods. Fortunately, there is a fairly effective and universal solution, using a stack, to calculate the most general expressions.

Earlier we processed expressions written in infix notation. This notation is not very convenient if an expression has operations with different priorities, especially when brackets are used. But we can use **postfix notation**, also known as **Reverse Polish notation (RPN)**. In this notation, operators follow their operands.

1. Add operands (numbers and variables) to the result (postfix notation) as they arrive.
2. If the stack is empty or contains a left parenthesis on top, push the incoming operator on the stack.
3. If the incoming operator has higher precedence than the top of the stack, push it on the stack.
4. If the precedence of the incoming operator is lower than or equal to that of the top of the stack, pop the stack and add operators to the result until you see an operator that has smaller precedence or a left parenthesis on the top of the stack; then add the incoming operator to the stack.
5. If the incoming element is a left parenthesis, push it on the stack.
6. If the incoming element is a right parenthesis, pop the stack and add operators to the result until you see a left parenthesis. Discard the pair of parentheses.
7. At the end of the expression, pop the stack and add all operators to the result.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stack top** | **Read object type** | **Condition** | **Action** |
|  | “(“ |  | Push object to stack |
|  | “)” |  | Pop operator to result until   * “(“ on Stack top   Discard “(“ and “)” |
|  | number, variable |  | Add object to result |
| [empty] or “(“ | operator |  | Push object to stack |
| operator | operator | Read object’s precedence is higher than Stack top’s | Push object to stack |
| operator | operator | Read object’s precedence is lower or equal than Stack top’s | Pop operator to result until   * “(“ on Stack top  or * lower precedence operator on Stack top   Push object to stack |
|  | [end of line] |  | Pop all operators to result |

When we have an expression in postfix notation, we can calculate it using another stack. To do that, scan the postfix expression from left to right:

* If the incoming element is a number, push it into the stack (the whole number, not a single digit!).
* If the incoming element is the name of a variable, push its value into the stack.
* If the incoming element is an operator, then pop twice to get two numbers and perform the operation; push the result on the stack.
* When the expression ends, the number on the top of the stack is a final result.