Aqui considera-se os seguintes operadores e respectivas relações de precendência

Operador	Prioridade para execução	Ordem de avaliação em caso de empate
*, /	2	esquerda->direita
+,-	1	esquerda -> direita

Expressão em notação infixa: operadores separados pelos respectivos operandos
(5 + 9) * 2 + 6 * 5 = 58
(5 + 9) * ((2 + 6) * 5) = 560

Na notação infixa a mudança na ordem de precedência de operadores pode ser feita mediante o uso de parênteses.
Os poloneses desenvolveram métodos de notação que não necessitam de uso de parênteses para definir uma outra ordem de avaliação das operações nas expressões:

Expressão em notação pré fixa: operadores antecedem os seus operandos
+ * + 5 9 2 * 6 5 = 58
* + 5 9 * + 2 6 5 = 560
Expressão em notação pós fixa: os operadores sucedem seus respectivos operandos
5 9 + 2 * 6 5 * + = 58
5 9 + 2 6 + 5 * * = 560
Os algoritmos aqui propostos se referem a expressões sem parênteses e a números de um dígito, é, no entanto, possível adaptar estes algoritmos para que contemplem tais requisitos.

1) Pseudo código para um Conversor Infixo para Pós-fixo (Tenembaum)
Elementos envolvidos: string de entrada (notação infixa), uma PILHA para operadores, string de saída (notação pósfixa)

```
ENQUANTO houver caracter válido na entrada
      { charEntrada = lê caracter da string de entrada;
        SE charEntrada é operando
          coloque charEntrada direto na string de saída na mesma ordem que havia na string de entrada
        SENÃO
          ENQUANTO(  buscaNoTopo(pilha, &charTopo) == SUCESSO   &&
                                              (precedência(charTopo) >= precedência(charEntrada)) )
            { desempilhe um item e o coloca na saída }
        empilhe charEntrada /*a pilha é usada por operadores*/
        }
 ENQUANTO( pilha != vazia)
            { desempilhe um item e o coloque na saída}
```

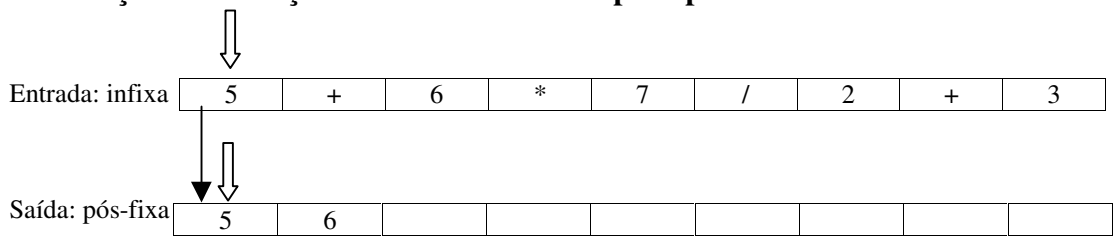
2) Pseudo código para avaliação de expressões em notação pós-fixa (Tenembaum)

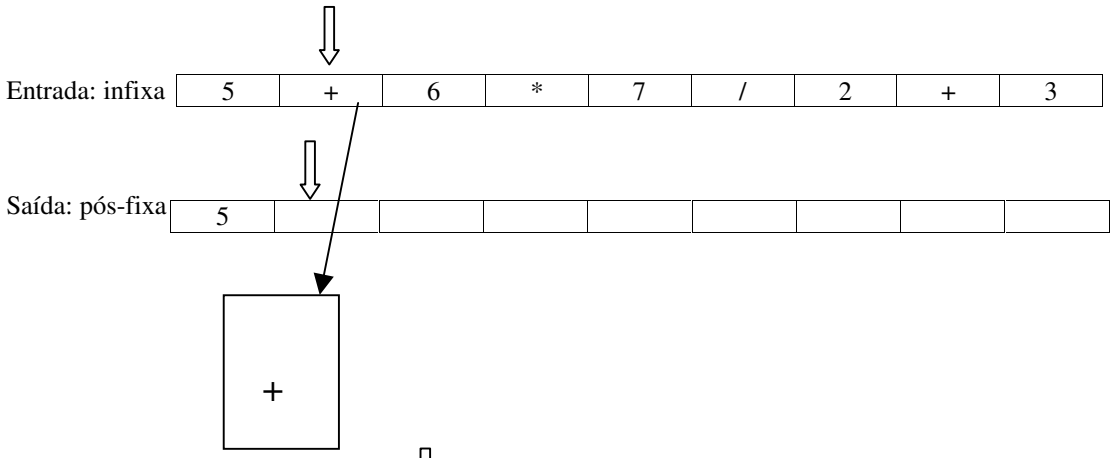
Entrada: cadeia de caracteres representando uma expressão em notação pós-fixa;
Saída: o valor da expressão;
Serão necessárias uma pilha para operandos

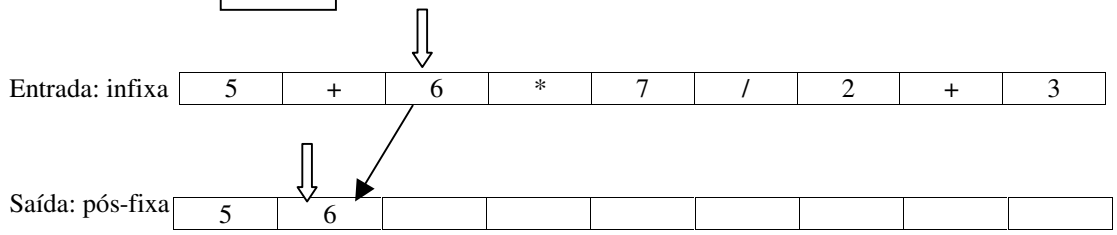
```
ENQUANTO(houver caractere valido na string de entrada)
{  leia um caracter da entrada
   SE(o caracter for operando)
         Empilhe o caracter
   SENAO
         { /*o caracter lido é operador */
           desempilhe dois operandos
           execute o cálculo aplicando o operador aos operandos
           empilhe o resultado do cálculo
         }
}
```

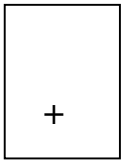
Ao final do processo a pilha guardará o resultado total da avaliação da expressão

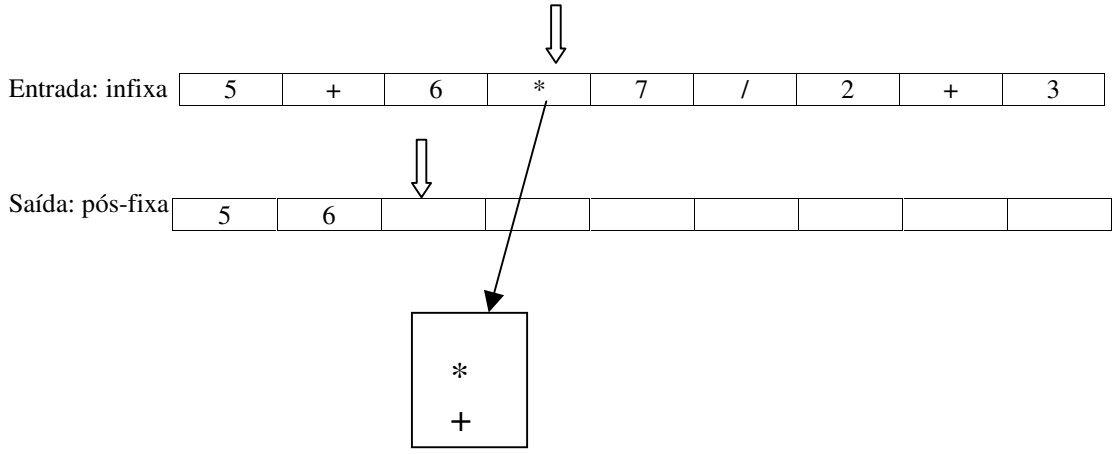
Simulação de execução da conversão infixo para pós-fixo



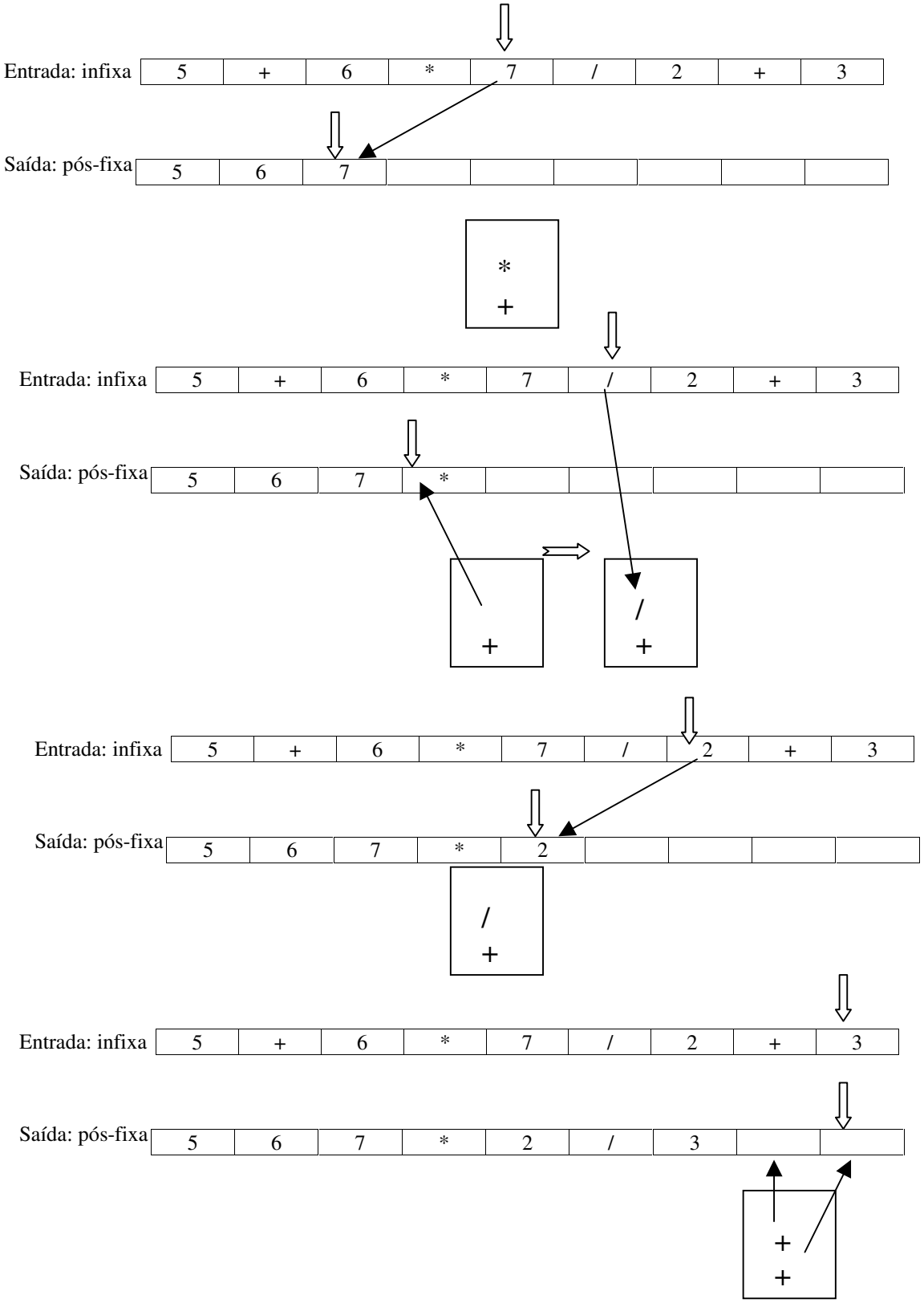








Continuação da simulação de execução da conversão infixo para pós-fixo



Avaliando a expressão pos fixa