|  |  |
| --- | --- |
| **Postfix 2 Prefix in C++** | |
| #include <iostream>  #include <stack>  using namespace std;  // Function to convert a postfix expression to a prefix expression.  string postToPre(string exp) {  stack<string> op;  for (int i = 0; i < exp.length(); i++) {  char ch = exp[i];  if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {  string val2 = op.top();  op.pop();  string val1 = op.top();  op.pop();  string cal = ch + val1 + val2;  op.push(cal);  } else {  op.push(string(1, ch));  }  }  return op.top();  }  int main() {  string postfix1 = "ab+c\*";  cout << "Postfix: " << postfix1 << " -> Prefix: " << postToPre(postfix1) << endl; // Expected: "\*+abc"  return 0;  } | Input: Postfix Expression = "ab+c\*"  Expected Prefix = "\*+abc" 📋 Dry Run Table:  | **i** | **ch** | **Stack Before** | **Action** | **Stack After** | | --- | --- | --- | --- | --- | | 0 | 'a' | [] | Operand → push "a" | ["a"] | | 1 | 'b' | ["a"] | Operand → push "b" | ["a", "b"] | | 2 | '+' | ["a", "b"] | Operator → pop "b", "a" → form +ab, push it | ["+ab"] | | 3 | 'c' | ["+ab"] | Operand → push "c" | ["+ab", "c"] | | 4 | '\*' | ["+ab", "c"] | Operator → pop "c", "+ab" → form \*+abc, push it | ["\*+abc"] |  ✅ Final Output: Prefix: \*+abc |
| Postfix: ab+c\* -> Prefix: \*+abc | |