|  |  |
| --- | --- |
| **Prefix to Postfix in C++** | |
| #include <iostream>  #include <stack>  #include <string>  using namespace std;  // Function to convert a prefix expression to a postfix expression.  string preToPost(string exp) {  stack<string> op;  int n = exp.length();  for (int i = n - 1; i >= 0; i--) {  char ch = exp[i];  if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {  string val1 = op.top();  op.pop();  string val2 = op.top();  op.pop();  string cal = val1 + val2 + ch;  op.push(cal);  } else {  op.push(string(1, ch));  }  }  return op.top();  }  int main() {  string prefix1 = "\*+AB-CDE";  cout << "Prefix: " << prefix1 << " -> Postfix: " << preToPost(prefix1) << endl; // Expected: "ABC+DE-\*"  string prefix2 = "\*-A/BC-/DEFG";  cout << "Prefix: " << prefix2 << " -> Postfix: " << preToPost(prefix2) << endl; // Expected: "ABC/-DE/FG-\*"  // Add more test cases as needed  return 0;  } | 📋 Dry Run Table:  | **i (index)** | **ch** | **Stack Before** | **Action** | **Stack After** | | --- | --- | --- | --- | --- | | 7 | 'E' | [] | Operand → push "E" | ["E"] | | 6 | 'D' | ["E"] | Operand → push "D" | ["E", "D"] | | 5 | 'C' | ["E", "D"] | Operand → push "C" | ["E", "D", "C"] | | 4 | '-' | ["E", "D", "C"] | Operator → pop "C" & "D" → "CD-" | ["E", "CD-"] | | 3 | 'B' | ["E", "CD-"] | Operand → push "B" | ["E", "CD-", "B"] | | 2 | 'A' | ["E", "CD-", "B"] | Operand → push "A" | ["E", "CD-", "B", "A"] | | 1 | '+' | ["E", "CD-", "B", "A"] | Operator → pop "A" & "B" → "AB+" | ["E", "CD-", "AB+"] | | 0 | '\*' | ["E", "CD-", "AB+"] | Operator → pop "AB+" & "CD-" → "AB+CD-\*" | ["AB+CD-\*"] |  Final Result: Top of the stack: **"AB+CD-\*"** |
| Prefix: \*+AB-CDE -> Postfix: AB+CD-\*  Prefix: \*-A/BC-/DEFG -> Postfix: ABC/-DE/F-\* | |