Prefix to Infix Conversion

Problem Statement

Given a prefix expression, convert it into an infix expression using a stack-based approach.

Prefix expressions have operators before operands. The task is to correctly format the expression into infix notation.

Logic Behind the Code

1. Traverse the prefix expression from right to left.
2. If the character is an operand (1-9), push it onto the stack.
3. If the character is an operator, pop the top two operands from the stack.
4. Merge them into an infix expression without brackets and push the result back to the stack.
5. Continue until the full expression is traversed. The final result in the stack is the infix expression.

C++ Code

```
#include<iostream>
#include<stack>
#include<string>
using namespace std;

string solve(string v1, string v2, char ch) {
  string s = "";
  s += v1;
  s.push_back(ch);
  s += v2;
  return s;
}

int main() {
  string s = "-+1/*+26483";
  stack<string> st;

  for(int i = s.length() - 1; i >= 0; i--) {
```

```
    if(s[i] >= '1' && s[i] <= '9') {
      st.push(to_string(s[i] - '0'));
    } else {
      string v1 = st.top(); st.pop();
      string v2 = st.top(); st.pop();
      char ch = s[i];
      string ans = solve(v1, v2, ch);
      st.push(ans);
    }
  }

  cout << st.top();
  return 0;
}
```

Dry Run Example

Expression: -+1/*+26483

Step-by-step Execution (Without Brackets):

1. Read '3' -> Push "3"
2. Read '8' -> Push "8"
3. Read '4' -> Push "4"
4. Read '6' -> Push "6"
5. Read '2' -> Push "2"
6. Read '+' -> Pop "2", "6", merge -> Push "2+6"
7. Read '/' -> Pop "2+6", "4", merge -> Push "2+6/4"
8. Read '*' -> Pop "2+6/4", "8", merge -> Push "2+6/4*8"
9. Read '1' -> Push "1"
10. Read '+' -> Pop "1", "2+6/4*8", merge -> Push "1+2+6/4*8"
11. Read '-' -> Pop "1+2+6/4*8", "3", merge -> Push "1+2+6*4/8-3"

Final Output: 1+2+6*4/8-3

Conclusion

This program successfully converts a given prefix expression into an infix expression using a stack. It efficiently processes the expression by traversing from right to left and using a stack to manage operands and operators.

Short Notes

- Prefix expressions have operators before operands.
- We process the string from right to left using a stack.
- If operand, push onto stack. If operator, pop top two, merge, and push back.
- The final expression in the stack is the required infix expression.