```cpp
#include <iostream>
#include <stack>
#include <string>

bool isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

int applyOperator(char op, int operand1, int operand2) {
    switch (op) {
        case '+':
            return operand1 + operand2;
        case '-':
            return operand1 - operand2;
        case '*':
            return operand1 * operand2;
        case '/':
            if (operand2 != 0) {
                return operand1 / operand2;
            } else {
                std::cerr << "Error: Division by zero!" << std::endl;
                exit(1);
            }
        default:
            std::cerr << "Error: Invalid operator!" << std::endl;
            exit(1);
    }
}

int evaluatePostfix(const std::string& postfixExpression) {
    std::stack<int> operandStack;

    for (char ch : postfixExpression) {
        if (isdigit(ch)) {
            operandStack.push(ch - '0');
        } else if (isOperator(ch)) {

            int operand2 = operandStack.top();
            operandStack.pop();

            int operand1 = operandStack.top();
            operandStack.pop();

            int result = applyOperator(ch, operand1, operand2);
            operandStack.push(result);
        }
    }

    if (!operandStack.empty()) {
```

```
50          return operandStack.top();
51      } else {
52          std::cerr << "Error: Invalid postfix expression!" << std::endl;
53          exit(1);
54      }
55  }
56
57  int main() {
58      std::string postfixExpression;
59
60
61      std::cout << "Enter a postfix expression: ";
62      std::getline(std::cin, postfixExpression);
63
64      int result = evaluatePostfix(postfixExpression);
65
66      std::cout << "Result: " << result << std::endl;
67
68      return 0;
69  }
70
```