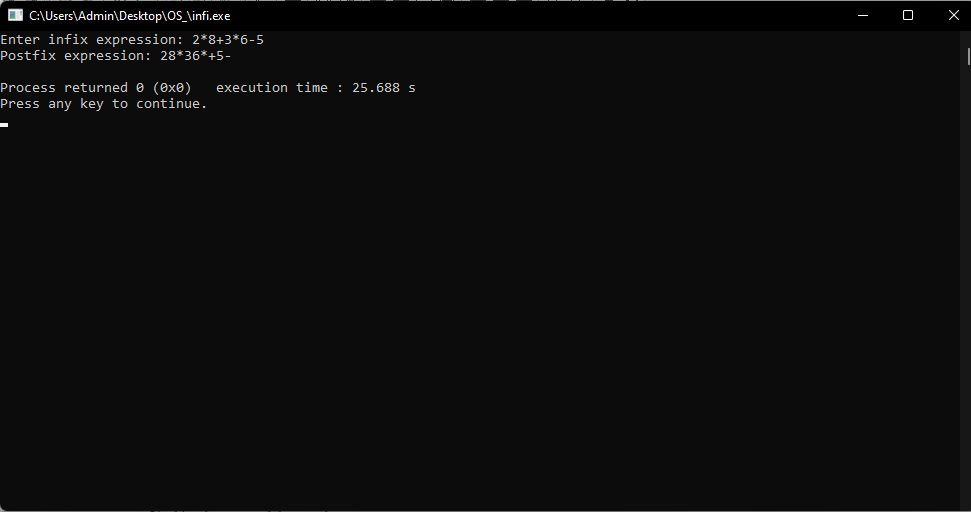
1. #include <stdio.h>  
#include <stdlib.h>  
  
#define MAX\_SIZE 100  
  
int isOperator(char ch) {  
    return (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%');  
}  
  
int precedence(char operator) {  
    if (operator == '+' || operator == '-')  
        return 1;  
    if (operator == '\*' || operator == '/' || operator == '%')  
        return 2;  
    return 0;  
}  
  
void infixToPostfix(char infix[], char postfix[]) {  
    char stack[MAX\_SIZE];  
    int top = -1;  
    int i, j;  
  
    for (i = 0, j = 0; infix[i] != '\0'; i++) {  
        if (infix[i] >= '0' && infix[i] <= '9') {  
            postfix[j++] = infix[i];  
        } else if (isOperator(infix[i])) {  
            while (top >= 0 && precedence(stack[top]) >= precedence(infix[i])) {  
                postfix[j++] = stack[top--];  
            }  
            stack[++top] = infix[i];  
        } else if (infix[i] == '(') {  
            stack[++top] = infix[i];  
        } else if (infix[i] == ')') {  
            while (top >= 0 && stack[top] != '(') {  
                postfix[j++] = stack[top--];  
            }  
            if (top >= 0 && stack[top] == '(') {  
                top--;  
            }  
        }  
    }  
  
    while (top >= 0) {  
        postfix[j++] = stack[top--];  
    }  
  
    postfix[j] = '\0';  
}  
int main() {  
    char infix[MAX\_SIZE], postfix[MAX\_SIZE];  
  
    printf("Enter infix expression: ");  
    scanf("%s", infix);  
  
    infixToPostfix(infix, postfix);  
  
    printf("Postfix expression: %s\n", postfix);  
  
    return 0;  
}



2. #include <stdio.h>  
#include <stdlib.h>  
#include <ctype.h>  
  
#define MAX\_STACK\_SIZE 100  
  
int stack[MAX\_STACK\_SIZE];  
int top = -1;  
  
void push(int item) {  
    if (top == MAX\_STACK\_SIZE - 1) {  
        printf("Stack Overflow\n");  
        exit(EXIT\_FAILURE);  
    }  
    stack[++top] = item;  
}  
  
int pop() {  
    if (top == -1) {  
        printf("Stack Underflow\n");  
        exit(EXIT\_FAILURE);  
    }  
    return stack[top--];  
}  
  
int isOperator(char ch) {  
    return (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '%');  
}  
  
int evaluatePostfix(char postfix[]) {  
    int i = 0;  
    while (postfix[i] != '\0') {  
        char currentSymbol = postfix[i];  
        if (isdigit(currentSymbol)) {  
            push(currentSymbol - '0');  
        } else if (isOperator(currentSymbol)) {  
            int operand2 = pop();  
            int operand1 = pop();  
            switch (currentSymbol) {  
                case '+':  
                    push(operand1 + operand2);  
                    break;  
                case '-':  
                    push(operand1 - operand2);  
                    break;  
                case '\*':  
                    push(operand1 \* operand2);  
                    break;  
                case '/':  
                    push(operand1 / operand2);  
                    break;  
                case '%':  
                    push(operand1 % operand2);  
                    break;  
            }  
        }  
        i++;  
    }  
    return pop();  
}  
  
int main() {  
    char postfixExpression[100];  
  
    printf("Enter postfix expression: ");  
    scanf("%s", postfixExpression);  
  
    int result = evaluatePostfix(postfixExpression);  
  
    printf("Result: %d\n", result);  
  
    return 0;  
}

