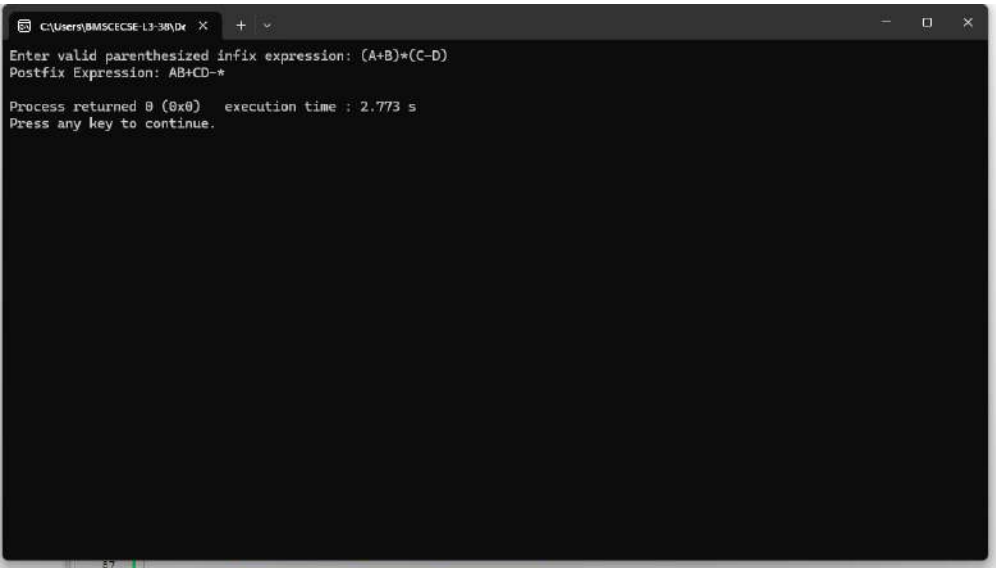
```
}

void infixToPostfix(char* infix, char* postfix) {
    int i, j = 0;
    char item, x;

    for (i = 0; infix[i] != '\0'; i++) {
        item = infix[i];

        if (item == ' ' || item == '\t')
            continue;

        if (isalnum(item)) {
            postfix[j++] = item;
        }

        else if (item == '(') {
            push(item);
        }

        else if (item == ')') {
            while ((x = pop()) != '(' && x != -1) {
                postfix[j++] = x;
            }
        }

        else if (isOperator(item)) {
            while (top != -1 && precedence(peek()) >= precedence(item)) {
                postfix[j++] = pop();
            }
            push(item);
        }
    }

    while (top != -1) {
        postfix[j++] = pop();
    }

    postfix[j] = '\0';
}


int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter valid parenthesized infix expression: ");
    fgets(infix, sizeof(infix), stdin);

    infixToPostfix(infix, postfix);
```




```
    printf("Enter valid parenthesized infix expression: ");
    fgets(infix, sizeof(infix), stdin);

    infixToPostfix(infix, postfix);

    printf("Postfix Expression: %s\n", postfix);

    return 0;
}
```

Logs & others

38\Desktop\p22.c | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 97, Col 1, Pos 1771 | Insert | Read/Write | default

Search

ENG IN 11:50:40 29-09-2025

```
        item = infix[i];

        if (item == ' ' || item == '\n')
            continue;

        if (isalnum(item)) {
            postfix[j++] = item;
        }

        else if (item == '(') {
            push(item);
        }

        else if (item == ')') {
            while ((x = pop()) != '(' && x != -1) {
                postfix[j++] = x;
            }
        }

        else if (isOperator(item)) {
            while (top != -1 && precedence(peek()) >= precedence(item)) {
                postfix[j++] = pop();
            }
            push(item);
        }
    }

    while (top != -1) {
        postfix[j++] = pop();
    }

    postfix[j] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter valid parenthesized infix expression: ");
    fgets(infix, sizeof(infix), stdin);

    infixToPostfix(infix, postfix);

    printf("Postfix Expression: %s\n", postfix);

    return 0;
}
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 97, Col 1, Pos 1771 | Insert | Read/Write | default

Search

ENG IN 11:49:46 29-09-2025

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX 100

char stack[MAX];
int top = -1;


void push(char c) {
    if (top == MAX - 1)
        printf("\nStack Overflow\n");
    else
        stack[++top] = c;
}


char pop() {
    if (top == -1)
        return -1;
    else
        return stack[top--];
}

char peek() {
    if (top == -1)
        return -1;
    else
        return stack[top];
}

int precedence(char op) {
    if (op == '+' || op == '-')
        return 1;
    if (op == '*' || op == '/')
        return 2;
    return 0;
}

int isOperator(char c) {
    return (c=='+'||c=='-'||c=='*'||c=='/');
}

void infixToPostfix(char* infix, char* postfix) {
    int i, j = 0;
    char item, x;

    for (i = 0; infix[i] != '\0'; i++) {
```

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 97, Col 1, Pos 1771 | Insert | Read/Write | default

Search

ENG IN 11:49:05 29-09-2025