**24F-0040**

**Laiba**

**Lab 05**

**Task:**

#include <iostream>

#include <string>

#include <stack>

using namespace std;

void working(string expression) {

stack<int> stack;

int op1, op2, result;

int start = 0, end = 0;

int len = expression.length();

while (end <= len) {

if (end == len || expression[end] == ' ') {

if (start == end) {

start = end + 1;

end++;

continue;

}

string ch = expression.substr(start, end - start);

if (ch.size() == 1 && (ch == "+" || ch == "-" || ch == "\*" || ch == "/")) {

if (stack.size() < 2) {

cout << "Operands are less than 2" << endl;

return;

}

op2 = stack.top();

stack.pop();

op1 = stack.top();

stack.pop();

switch (ch[0]) {

case '+': result = op1 + op2; break;

case '-': result = op1 - op2; break;

case '\*': result = op1 \* op2; break;

case '/':

if (op2 == 0) {

cout << "Zero can't be divided!" << endl;

return;

}

result = op1 / op2;

break;

}

stack.push(result);

}

else {

try {

int num = stoi(ch);

stack.push(num);

}

catch (...) {

cout << "Invalid token: " << ch << endl;

return;

}

}

start = end + 1;

}

end++;

}

if (stack.size() == 1) {

cout << "Result: " << stack.top() << endl;

}

else {

cout << "Invalid expression!" << endl;

}

}

int main() {

string expression;

cout << "Enter the Problem/expression: ";

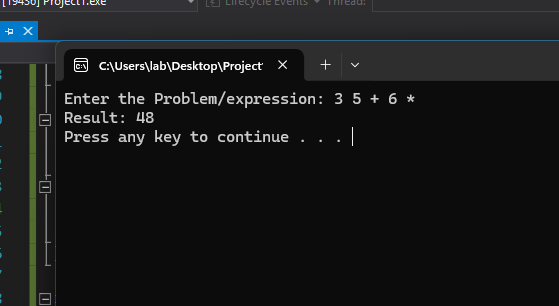
getline(cin, expression);

working(expression);

system("Pause");

return 0;

}



**Task**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int precedence(char op) {

if (op == '+' || op == '-') return 1;

if (op == '\*' || op == '/') return 2;

return 0;

}

bool isOperator(char ch) {

return ch == '+' || ch == '-' || ch == '\*' || ch == '/';

}

void infixToPostfix(const string& expression) {

stack<char> st;

string postfix = "";

for (char ch : expression) {

if (ch == ' ') continue;

if (isalnum(ch)) {

postfix += ch;

}

else if (ch == '(') {

st.push(ch);

}

else if (ch == ')') {

while (!st.empty() && st.top() != '(') {

postfix += st.top();

st.pop();

}

if (!st.empty()) st.pop();

}

else if (isOperator(ch)) {

while (!st.empty() && precedence(st.top()) >= precedence(ch)) {

postfix += st.top();

st.pop();

}

st.push(ch);

}

}

while (!st.empty()) {

postfix += st.top();

st.pop();

}

cout << "Postfix expression: " << postfix << endl;

}

int main() {

string expression;

cout << "Enter infix expression (e.g. (a+b)\*c): ";

getline(cin, expression);

infixToPostfix(expression);

system("Pause");

return 0;

}

