**Modern Education Society’s  
College of Engineering, Pune**

|  |  |
| --- | --- |
| **NAME OF STUDENT:** Prathamesh Kalyan Sable | **CLASS:** SE Comp 1 |
| **SEMESTER/YEAR:** Sem-3 / 2022-23 | **ROLL NO:** 015 |
| **DATE OF PERFORMANCE:**  / /2022 | **DATE OF SUBMISSION:** / /2022 |
| **EXAMINED BY:** Prof. N.S. Gore | **EXPERIMENT NO: DSL D-27** |

**TITLE :PERFORM VARIOUS OPERATION ON STACK AND IMPLEMENT ITS APPLICATION.**

**PROBLEM STATEMENT :** Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions

i. Operands and operator, both must be single character.

ii. Input Postfix expression must be in a desired format.

iii. Only '+', '-', '\*' and '/ ' operators are expected.

**OBJECTIVES :  
 1.** To understand structure of stack.  
 **2.** To understand How to Create, Display and perform various operation on stack.

**OUTCOME :** 1. To operate on the various structured data.  
 2. To analyze the problem to apply suitable algorithm and data structure.  
 3. To discriminate the usage of various structure in approaching problem solution.

**PRE-REQUISITES :  
 1.** Knowledge of C++ Programming  
 2. Knowledge of stack.

**APPARATUS :**

Computer Machine, c++ compiler installed, Code Editor, etc.

**QUESTIONS :**1. Evaluate the following postfix expression and show all steps: ab\*c+d-e+  
2. Write an algorithm to prefix to postfix.

**SOURCE CODE:**

#include <iostream>

#define MAX\_ARR\_SIZE 30

using namespace std;

// global value list

int VAR\_VALS[58] = {0};

int VAR\_COUNT = 0;

bool IS\_VAL\_SET = false;

// Stack to store characters

class char\_stack {

    char list[MAX\_ARR\_SIZE];

    public:

    int top;

    char\_stack() { top = -1; }

    bool isempty() { return (top <= -1) ? true : false; }

    bool isfull() { return (top == MAX\_ARR\_SIZE - 1) ? true : false; }

    void push(char elmt) {

        if (not isfull()) {

            top++;

            list[top] = elmt;

        }

    }

    char peek() { return (top != -1) ? list[top] : ' '; }

    char pop() {

        if (not isempty()) {

            top--;

            return list[top + 1];

        }

    }

};

class stack {

    int list[MAX\_ARR\_SIZE];

    public:

    int top;

    stack() { top = -1; }

    bool isempty() { return (top <= -1) ? true : false; }

    bool isfull() { return (top == MAX\_ARR\_SIZE - 1) ? true : false; }

    void push(int elmt) {

        if (not isfull()) {

            top++;

            list[top] = elmt;

        }

    }

    int pop() {

        if (not isempty()) {

            top--;

            return list[top + 1];

        }

    }

    void dispaly() {

        for (int i = 0; i < top + 1; i++) {

            cout << list[i] << " ";

        }

        cout << endl;

    }

};

// function which return the proirity of the operation

int priority(char opp) {

    if (opp == '+' || opp == '-')

        return 1;

    else if (opp == '\*' || opp == '/')

        return 2;

    else

        return 0;

}

string infixToPostfix(string exp) {

    string postfix;

    char ch;

    char\_stack opps;

    for (int i = 0; i < exp.size(); i++) {

        ch = exp[i];

        // if char is parenthesis so highest priority

        if (ch == '(') {

            opps.push('(');

        } else if (ch == '+' || ch == '^' || ch == '-' || ch == '\*' || ch == '/') {

            // all operators in stack with higher ar equal priority are poped

            while (!opps.isempty() && priority(ch) <= priority(opps.peek()))

                postfix += opps.pop();

            opps.push(ch);

        } else if (ch == ')') {

            while (opps.peek() != '(')

                postfix += opps.pop();

            opps.pop();

        } else {

            postfix += ch;

        }

    }

    while (!opps.isempty())

        postfix += opps.pop();

    return postfix;

}

// function to set values of variables

void setVariables(int count = 0) {

    int varnum;

    if (count == 0) {

        cout << "Enter Number of Variables in Expression:";

        cin >> varnum;

    } else {

        varnum = count;

    }

    VAR\_COUNT = varnum + 64;

    for (int i = 0; i < varnum; i++) {

        cout << "Enter Value for '" << (char)(65 + i) << "':";

        cin >> VAR\_VALS[i];

        VAR\_VALS[i + 32] = VAR\_VALS[i];  // setting value of lowercase variables

                                         // same as uppercase

    }

    IS\_VAL\_SET = true;

}

// postfix calculator

int evalPostfix(string exp) {

    int max\_var = 0;

    for (int i = 0; i < exp.size(); i++) {

        if (exp[i] >= 65 && exp[i] <= 90) {

            if (exp[i] > max\_var) {

                max\_var = exp[i];

            }

        } else if (exp[i] > 97 && exp[i] <= 122) {

            if ((exp[i] - 32) > max\_var) {

                max\_var = (exp[i] - 32);

            }

        }

    }

    if (!IS\_VAL\_SET || max\_var > VAR\_COUNT)

        setVariables(max\_var - 64);

    int len = exp.size();

    int val1, val2, ans, temp;

    stack values;

    for (int i = 0; i < len; i++) {

        switch (exp[i]) {

            case '+':

                val2 = values.pop();

                val1 = values.pop();

                temp = val1 + val2;

                values.push(temp);

                break;

            case '\*':

                val2 = values.pop();

                val1 = values.pop();

                temp = val1 \* val2;

                values.push(temp);

                break;

            case '-':

                val2 = values.pop();

                val1 = values.pop();

                temp = val1 - val2;

                values.push(temp);

                break;

            case '/':

                val2 = values.pop();

                val1 = values.pop();

                temp = val1 / val2;

                values.push(temp);

                break;

            default:

                values.push(VAR\_VALS[int(exp[i] - 65)]);

                break;

        }

    }

    return values.pop();

}

int main() {

    cout << "\n\*\*\*\nNot Case Sensitive\nMaximum 30 characters\n\*\*\*\n" << endl;

    int ch, ans;

    bool while\_ctrl = true;

    string temp, LastInfixExp;

    while (while\_ctrl) {

        cout << "MENU\n1. Infix To Postfix\n2. Set Variable Values\n3. Postfix "

                "Calculator\n4. Exit\nEnter Your Choice:";

        cin >> ch;

        switch (ch) {

            case 1:

                cout << "Enter a Infix Expression:";

                cin >> temp;

                LastInfixExp = infixToPostfix(temp);

                cout << "Postfix Expression is :" << LastInfixExp << endl;

                break;

            case 2:

                setVariables();

                cout << "Variables set sucessfully" << endl;

                break;

            case 3:

                if (LastInfixExp.size() != 0) {

                    cout << "Enter Postfix Expression(Enter 0 to use last "

                            "converted Expression):";

                    cin >> temp;

                    if (temp == "0") {

                        ans = evalPostfix(LastInfixExp);

                    } else {

                        ans = evalPostfix(temp);

                    }

                    cout << "Value of expression is " << ans << endl;

                } else {

                    cout << "Enter Postfix Expression:";

                    cin >> temp;

                    ans = evalPostfix(temp);

                    cout << "Value of expression is " << ans << endl;

                }

                break;

            case 4:

                cout << "Thank You!" << endl;

                while\_ctrl = false;

                break;

            default:

                cout << "Enter a Valid Choice:" << endl;

                break;

        }

    }

    return 0;

}

**OUTPUT:**



