STACK – (infix2postfix)

**CODE:**

#include<iostream>

#include<string>

using namespace std;

class stack

{

public:

int top;

int size;

char\* arr;

stack()

{

top = -1;

cout << " SIZE : ";

cin >> size;

arr = new char[size];

}

bool isempty()

{

if (top == -1)

return true;

else return false;

}

bool isfull()

{

if (top == (size - 1))

return true;

else return false;

}

void push(char a)

{

if (isfull())

{

cout << " \n The stack is full !!";

}

else

{

top++;

arr[top] = a;

}

}

void pop()

{

arr[top] = -1;

top--;

}

char gettop()

{

return arr[top];

}

};

class infixtopostfix

{

public:

string expression;

string result;

stack s;

infixtopostfix()

{

expression = "";

result = "";

}

void input()

{

cout << " \n Enter the Expression : \n";

cin.ignore();

getline(cin, expression);

}

int precedence(char c)

{

if (c == '(' )

return 3;

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

return 2;

else if (c == '+' || c == '-')

return 1;

return 0;

}

void solution()

{

for (int i = 0; i < expression.length(); i++)

{

char c = expression[i];

if (c >= 'a' && c <= 'z' || c>='A' && c <= 'Z' || c>=0 && c <= 9)

{

result += c;

}

else if (c == '(') //if the opening bracket founf push it on stack

{

s.push(c);

}

else if (c == ')') //if closing bracket found

{

while (s.gettop() != '(') //pop everything until the opening bracket is found

{

result += s.gettop();

s.pop();

}

s.pop();

}

else if (c == '+' || c == '/' || c == '\*' || c == '-') //if the operator is found

{

while ( precedence(c) <= precedence(s.gettop())) //while top is ( and the precedence is less the the operator on stack add top eles to result

{

result += s.gettop();

s.pop();

}

s.push(c); //if the precedence is greater than the top element then push it on stack

}

}

if (!s.isempty())

{

while (!s.isempty()) //add all the remaining operators to result

{

result += s.gettop();

s.pop();

}

}

cout << " \n\n\t\t Expression Converted to Postfix !!!\n";

cout << "\n InFix :\t" << expression << endl;

cout << " PostFIx : \t" << result << endl;

}

};

int main()

{

infixtopostfix infx;

int repeat = 1,ch;

while (repeat == 1)

{

cout << " \n\n\t\t --------------------- STACK ---------------------\n\n";

cout << " 1. Enter The Infix Expression \n";

cout << " 2. Show the Expression\n";

cout << " 0. Exit \n";

cin >> ch;

switch (ch)

{

case 1:

system("cls");

infx.input();

break;

case 2:

system("cls");

infx.solution();

break;

case 0:

repeat++;

break;

}

}

system("pause");

return 0;

}