**Experiment No. 3C**

**Title :** Implementation of prefix to postfix expression using recursion.

**Problem Statement :** Convert the given prefix expression to postfix expression in C++ using recursion.

**Algorithm :**

**S1 :** Start

**S2 :** Declare a stack of type string using stack library in C++. Declare two strings for prefix and postfix expressions.

**S3 :** Take the input from user for prefix expression.

**S4 :** Declare isoperator() which returns true if character is operator and false if not.

**S5 :** Using recursion take every character of the expression starting from last character.

**S6 :** Check if the character is an operator if not push the character in stack.

**S7 :** If the character is an operator pop twice from stack and concatenate the characters then push in stack again.

**S8 :** The value in stack at the end of the expression is answer of postfix expression

**S9 :** Stop

**Code :**

#include<iostream>

#include<stack>

using namespace std;

string prefix; //Declare strings for prefix and postfix

string postfix;

int i;

int l;

stack<string> s; //Declare a stack of type string

bool isoperator(char ch) //checks if character is operator or not

{

switch(ch)

{

case '+' :

case '-' :

case '\*' :

case '/' :

return true;

}

return false;

}

void eval(int i)

{

if(i<0)

{

postfix = s.top();s.pop();

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

return;

}

if(isoperator(prefix[i])) //if it is an operator pop the strings and concatinate

{

string p1 = s.top();s.pop();

string p2 = s.top();s.pop();

string temp = p1+p2+prefix[i];

s.push(temp);

}

else{

s.push(string(1,prefix[i])); //push in stack if not an operator

}

i = i - 1;

eval(i);

}

int main()

{

cout<<"enter the prefix expression :";

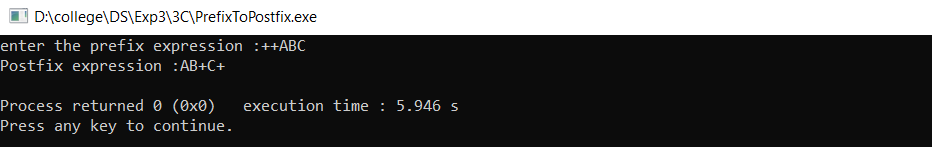
cin>>prefix;

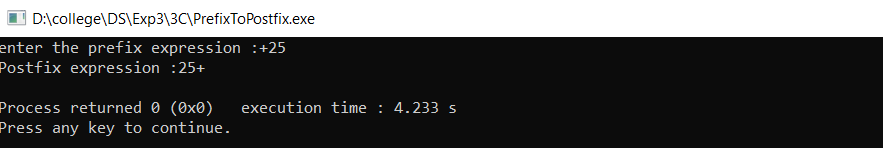
l = prefix.size();

eval(l-1); //Recursion

}

**Output :**





**Analysis :**

The program checks if character is an operator or not but if the expression has numbers or the variables both are treated similarly rather than returning the answer of expression with numbers.

This code is feasible in C++ since we declare the stack using header and concatenation of strings is simple in C++ as compared to C.

Since we use recursion the code is a bit inefficient as compared to loop.