



Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
COMPS Department

Experiment	1
Aim	Infix to postfix conversion using Stack
Name	David Daniels
UID	202330038
Class	Div -A
Batch	C
Date of Submission	14-8-24

Theory	<p>A stack is a fundamental data structure in computer science that follows the Last In, First Out (LIFO) principle. the last element added to the stack will be the first one to be removed. It can be implemented by an array.</p> <p>Basic Operations</p> <ol style="list-style-type: none">1. Push: Add an element to the top of the stack.2. Pop: Remove and return the element from the top of the stack.3. Peek (or Top): Retrieve the element at the top of the stack without removing it.4. IsEmpty: Check if the stack is empty.5. IsFull: Check if the stack is full
Algorithm	<p>Read input from Left-to-Right and</p> <p>if an operand is read copy it to the output</p> <p>if operator is '(' then push it into the stack</p> <p>If operator is ')' then pop the stack until '(' is not found. When that occurs, both parentheses are discarded</p> <p>if an operator is read and has a higher precedence than the operator at the top of the stack, the operator being read is pushed onto the stack</p> <p>while the precedence of the operator being read is lower than or equal to the precedence of the operator at the top of the stack, the operator at the top of the stack is popped and copied to the output</p> <p>when reached the end of the expression, the remaining operators in the stack are popped and copied to the output.</p>

Problem Solving

Page No. _____
Date _____

David Daniels

ex $1 + 2 * (4 / 2 + 1) - 4$

1) 1 2 2) 1 2 3) 1 2 4 2

+

*
+

/

/

c

*

+

4) 1 2 4 2 / 1 5) 1 2 4 2 / 1 + 6) ~~1 2 4 2 / 1 +~~
1 2 4 2 / 1 + *

+
c
*
+

+
+

-

7) ~~1 2 4 2 / 1 +~~ 1 2 4 2 / 1 + * + 4 -

Program(Code)

```
#include <iostream>

using namespace std;

class in_to_post
{
public:
    int top;
    string output="";
    int size = 20;
    char arr [20];

    bool is_full()
```

```

{
    if(top==size-1)
    {
        return true;
    }
    return false;
}

bool is_empty()
{
    if (top==-1)
    {
        return true;
    }
    return false;
}

void push(char data)
{
    if (is_full())
    {
        cout<<"is full"<<endl;
        return;
    }
    cout<<"pushing "<<data<<endl;
    top++;
    arr[top]=data;
    return;
}

char pop()
{
    if (is_empty())
    {
        cout<<"EMPTY"<<endl;
        return 'f';
    }
    else
    {
        char data = arr[top];
        top--;
        cout<<"popping "<<data<<endl;
        return data;
    }
}

bool is_operator(char s)
{
    if (s=='*' || s=='/' || s=='+' || s=='-' || s=='^' || s=='$' || s=='') || s=='(')

```

```

    {
        return true;
    }
    return false;
}

int precedence(char s)
{
    if (s=='*'||s=='/')
    {
        return 2;
    }
    else if(s=='+'||s=='-')
    {
        return 1;
    }
    else if(s=='^'||s=='$')
    {
        return 3;
    }
    else
    {
        return -1;
    }
}

string convert (string imput)
{
    top=-1;
    int j=0;
    for (int i = 0; i < imput.length();i++)
    {
        if (!is_operator(imput[i]))
        {
            output[j]=imput[i];
            i++;
            j++;
        }
        else
        {
            if (imput[i]=='(')
            {
                push(imput[i]);
                i++;
            }
            else if(imput[i]==')')
            {
                //keep on popping till (
                while (true)
                {

```

```

        char temp=arr[top];
        if (temp=='(')
        {
            pop();
            break;
        }

        output[j]=pop();
        j++;
    }
    i++;

}
else if(precedence(input[i])>precedence(arr[top]))
{
    push(input[i]);
    i++;
}
else if (precedence(input[i])<=precedence(arr[top]))
{
    output[j]=pop();
    j++;
}

}

}

//empty out
while (!is_empty())
{
    output[j]=pop();
    j++;
}

return output;
}

};

int main(int argc, char const *argv[])
{
    in_to_post e1;
    string i;
    cout<<"Enter Expression:"<<endl;
    getline(cin,i);
    cout<<e1.convert(i)<<endl;
    return 0;
}

```

Output	<pre> Enter Expression: 1+2*(4/2+1)-4 pushing + pushing * pushing (pushing / popping / pushing + popping + popping (popping * popping + pushing - popping - 1242/1+*+4- </pre>
Conclusion	Thus I have learned how to implement stack and convert an infix expression to postfix in C++