Infix to Postfix Conversion - Detailed Breakdown (Hinglish)

Introduction

Yeh document C++ code ka detailed breakdown provide karta hai jo ek infix expression ko postfix expression me convert karta hai. Code likhne ka thought process, structure, working, dry run, aur test cases ko achhe se explain kiya gaya hai. Ek short notes section bhi diya gaya hai quick revision ke liye.

Code Snippet

```
#include<iostream>
#include<string>
#include<stack>
using namespace std;

int priority(char ch){
  if(ch=='+'||ch=='-') return 1;
  else if(ch=='*'||ch=='/')return 2;
}

string eval(string v1,string v2,char ch){
  string s = "";
  s += v1;
  s += v2;
  s.push_back(ch);
  return s;
}

int main(){
  string s = "1+(2+6)*4/8-3";
  stack<string> val;
  stack<char> op;
  for(int i=0;i<s.length();i++){
    if(s[i]>=48 && s[i]<=57){
```

```
  val.push(to_string(s[i]-48));
 }
 else{
  if(op.size()==0){
    op.push(s[i]);
  }
  else if(s[i]=='('||op.top()=='(') op.push(s[i]);
  else if(op.size()>0 && s[i]==')'){
    while(op.top()!='('){
      string v2 = val.top();
      val.pop();
      string v1 = val.top();
      val.pop();
      char ch = op.top();
      op.pop();
      string ans = eval(v1,v2,ch);
      val.push(ans);
    }
    op.pop();
  }
  else if(priority(op.top())<priority(s[i])) op.push(s[i]);
  else{
    while(op.size()>0 && priority(op.top())>=priority(s[i])){
    string v2 = val.top();
    val.pop();
    char ch = op.top();
    op.pop();
    string v1 = val.top();
    val.pop();
    string ans = eval(v1,v2,ch);
    val.push(ans);
```

```
        }
        op.push(s[i]);
      }
    }
  }
  while(op.size()>0){
        string v2 = val.top();
        val.pop();
        char ch = op.top();
        op.pop();
        string v1 = val.top();
        val.pop();
        string ans = eval(v1,v2,ch);
        val.push(ans);
  }
  cout<<val.top();
  return 0;
}
```

Thought Process Behind the Code

1. **Operands Handle Karna:**
   - Agar character ek digit hai, toh usko string me convert karke val stack me push kar dete hain.
2. **Operators Handle Karna:**
   - Agar operator stack empty hai toh push kar dete hain.
   - Agar opening bracket (aata hai, toh ise operator stack me push kar dete hain.
   - Agar closing bracket) aata hai, toh tab tak process karte hain jab tak ek opening bracket (nahi mil jata.
   - Agar current operator ki precedence operator stack ke top wale se zyada hai, toh ise push kar dete hain.
   - Nahi toh, pehle top wale operators evaluate karte hain fir naye operator ko push kar dete hain.

3. **Final Processing:**
    - Loop ke baad jitne bhi bache huye operators hain unko process karte hain jab tak operator stack empty nahi ho jata.

Dry Run (Step-by-Step Execution)

Input Expression: **1+(2+6)*4/8-3**

Postfix Expression Output:

126+4*8/+3-

Stepwise Execution

Step	**Character**	**Action**
1	1	Push to val (1)
2	+	Push to op (+)
3	(	Push to op (()
4	2	Push to val (2)
5	+	Push to op (+)
6	6	Push to val (6)
7	)	Evaluate (2 6 + → 26+), pop (
8	*	Push to op (*)
9	4	Push to val (4)
10	/	Push to op (/)
11	8	Push to val (8)
12	Evaluate	(4 8 / → 48/), push result
13	Evaluate	(26+ 48/ * → 126+48/*)
14	-	Push to op (-)
15	3	Push to val (3)
16	Evaluate	(126+48/* 3 - → 126+48/+3-)

Final Output

126+4*8/+3-

Short Notes (Quick Revision)

- **Operands (digits)** → Directly push onto val stack.
- **Operators (+, -, , /)* → Precedence ke according push ya evaluate karna.
- **Parentheses (( ))** → Process karna jab tak opening bracket (nahi mil jata.
- **Final Evaluation** → Sabhi remaining operators ko process karna.
- **Final val.top()** → Postfix expression return karega.

Summary

Yeh C++ program ek infix expression ko postfix expression me convert karta hai do stacks ka use karke (ek operands ke liye aur ek operators ke liye). Algorithm precedence rules follow karta hai aur brackets ko sahi se handle karta hai. Dry run se program ki correctness verify hoti hai.
