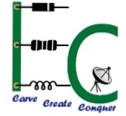
****

**THE NATIONAL INSTITUTE OF ENGINEERING**

**MYSURU-570008**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**C++ COURSE PROJECT [EC0412]**

VI Semester, A Section

Report On

**Implementation of Stack to check Parenthesis matching**

**Under the guidance of:**

**Ms Sharmila B S**

Assistant Professor, Dept. of ECE.

**Submitted By:**

Manav G : 4NI15EC056

Akhilesh Guddad : 4NI15EC008

Basavaraj R Sunagad : 4NI15EC025

Bhargava S B : 4NI15EC027

**2017-18**

**Table of Content**

Introduction ………………………………………………………………………………………………..1

Working ………………………………………………………………………………………………………2

Experimental Results……..………………….…………………………………………………..…….3

Reference …………………………………………………….……………………………………………..4

Appendix …………………………………………………………………………………………………….4

**Introduction**

Parenthesis matching is a method to check whether every left parenthesis has a matching right parenthesis in a given expression. This is also important in programming where a missing or extra parenthesis can lead to erroneous output.

**Working**

The principle involved is to find a match for every opening parenthesis a closing parenthesis.

The steps involved are:

1. Scan the expression from left to right.
2. Whenever a left parenthesis is encountered push it onto stack.
3. Whenever a right parenthesis is encountered pop the stack and check for parenthesis match.
4. If match is true proceed until stack is empty and, else put a message to check the expression
5. If the stack is empty and still a right parenthesis is encountered then also put a message to check the expression.

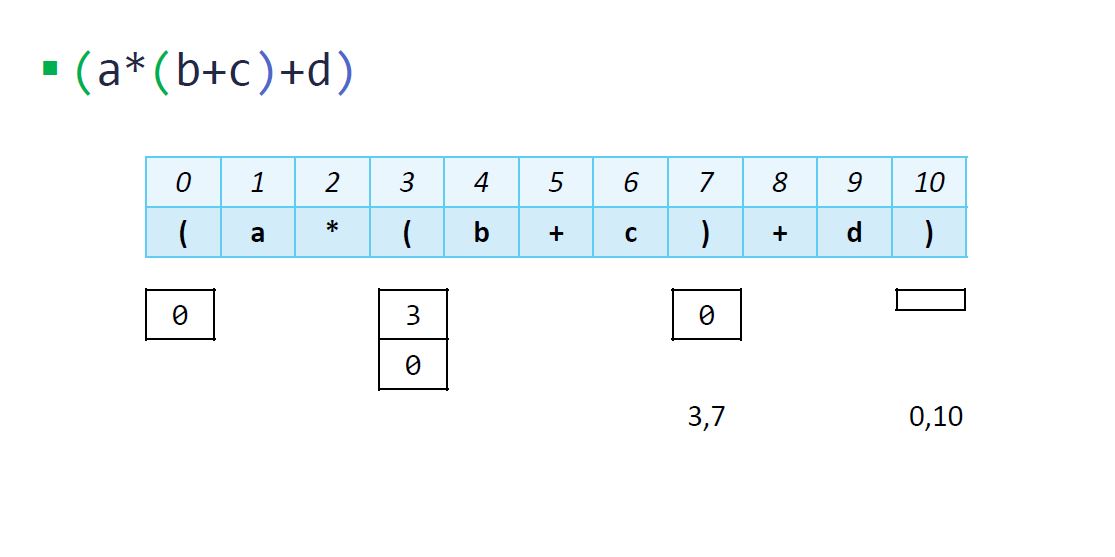
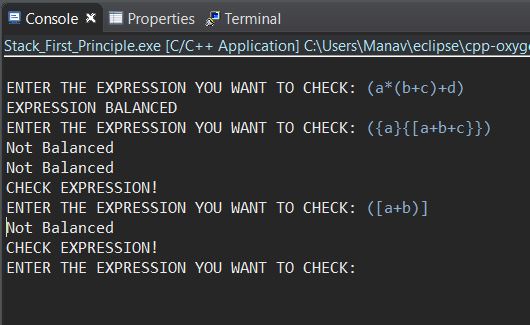


Fig.1 Example for Parenthesis matching

**Experimental Results**

****

**Reference**

* cds.iisc.in/wp-content/uploads/DS286.AUG2016.L8.Stacks.pdf

**Appendix**

Code in C++

/\*

\* parenthesis\_checker.cpp

\* this code uses a stack to check a mathematical expression if the left and right parenthesis are matched.

\* Such algorithms are generally used in GUI based environments.

\* Created on: 15-Apr-2018

\*

\*/

#include <iostream>

#include<string.h>

using namespace std;

class Stack { //class to create a stack

private:

int \*p;

int length;

public:

int top;

Stack(int = 0);

~Stack();

void push(int); //to push an element

int pop(); // to pop an element

void display(); //to display the contenets of the stack

int empty(); // to check is stack is empty or not

};

Stack::Stack(int size) {

top = -1;

length = size;

if (size == 0)

p = 0;

else

p = new int[length];

}

Stack::~Stack() {

if (p != 0)

delete[] p;

}

void Stack::push(int elem) {

if (p == 0) //If the stack size is zero, allow user to mention it at runtime

{

cout << "Stack of zero size" << endl;

cout << "Enter a size for stack : ";

cin >> length;

p = new int[length];

}

if (top == (length - 1)) //If the top reaches to the maximum stack size

{

cout << "\nCannot push " << elem << ", Stack full" << endl;

return;

} else {

top++;

p[top] = elem;

}

}

int Stack::pop() {

if (p == 0 || top == -1) {

//cout << "Stack empty!";

return -1;

}

int ret = p[top];

top--;

return ret;

}

void Stack::display() {

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

cout << p[i] << " ";

cout << endl;

}

int Stack::empty() {

if (top == -1) {

return (1);

} else {

return (0);

}

}

bool areParenthesisBalanced(char expr[])

{ int size= strlen(expr);

Stack s(size);

char a, b, c;

int t,q,r;

// Traversing the Expression

for (int i=0; i<strlen(expr); i++)

{

if (expr[i]=='('||expr[i]=='['||expr[i]=='{')

{

// Push the element in the stack

s.push(expr[i]);

}

else

{

switch (expr[i])

{

case ')':

// Store the top element in a

a = s.top;

t= s.pop();

if (s.p[a]=='{'||s.p[a]=='['||t==-1){

cout<<"Not Balanced\n";

s.push(a);

}

break;

case '}':

// Store the top element in b

b = s.top;

r=s.pop();

if (s.p[b]=='('||s.p[b]=='['||r==-1){

cout<<"Not Balanced\n";

s.push(b);

}

break;

case ']':

// Store the top element in c

c=s.top;

q=s.pop();

if (s.p[c]=='('||s.p[c]=='{'||q==-1){

cout<<"Not Balanced\n";

s.push(c);

}

break;

}

}

}

// Check Empty Stack

if (s.empty())

return true;

else

return false;

}

int main()

{

char expr[50];

cout<<"ENTER THE EXPRESSION YOU WANT TO CHECK: ";

cin>>expr;

if(areParenthesisBalanced(expr))

cout<<"EXPRESSION BALANCED";

else

cout<<"CHECK EXPRESSION!";

}