21143 Tanmay Karmarkar

Paranthesis Checker

Problem statement:
In any language program mostly syntax error occurs due to unbalancing delimiter such as (), {}, []. Write C++ program using stack to check whether given expression is well paranthesized or not.

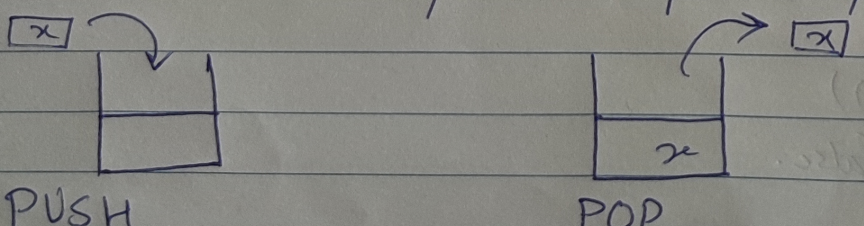
Objective: To
1) Implement stack data structure using array or linked list.
2) Implement paranthesis checker using stack.

Outcome:
Write a program in C++ to create paranthesis checker using stack

S/W & H/W requirements:
Windows 10, mingw compiler, VSCODE, 8GB RAM, 512GB SSD.

Theory: Stack is a data structure that follows LIFO rule. The element the enters last goes out first. Stack can be implemented using either array or linked list. With array, number of elements are limited due to static memory allocation. Linked list provides dynamic memory allocation.

Stack has 2 main operations push & ~~pull~~ pop



Algorithm for main

1. Take input of string
2. Pass it through the bool function.
3. If it returns true print expression is valid else unbalanced.

Algorithm for bool is balanced.

1. Declare stack s of char data type.
2. Declare char c.
3. Traverse through the string expression.
4. If it has '(', '[', '{' push in stack.
5. If stack is empty return false.
6. Switch statement for each ')', ']', '}'.
   for ')' : If st.top = '{' || '[' return false else true.
   ']' : If s.top = '(' || '{' return false else true.
7. '}' : If s.top = '[' || '(' return false else true.
8. ~~If stack~~ Return (s.empty).

Pseudocode:

```
bool isbalanced ()
stack <char> s;
 char c;
for (int i=0; i<exp.length() ; i++)
{ if (exp[i] = '(' || '{' || '[')
  {  s.push (exp[i])
      continue
  }
if (s.empty ())
    return false.
```

```
switch (exp[i])
case ')' : ch = s.top()
           s.pop()
  if (ch == '{' || '[')
    return false;
    break
```

Similar for case ']' & case '}'.

```
return (s.empty()).

int main()
string exp;
cout << "Enter expression";
cin >> exp
if(isbalanced(exp))
cout << "balanced";
else cout << "Not Balanced";
```