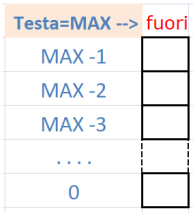
Gestione di una PILA mediante ARRAY

problema OIS dic. 2014 - controllo corretta sequenza di parentesi

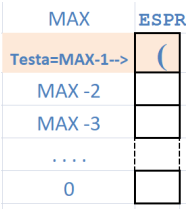
prof.ssa P.Grandillo

```
#include <iostream>
#include <fstream>
using namespace std;

#define MAX 5001
  char ESPR[MAX];

  int Testa=MAX;

   ifstream fr ("parentin.txt");
   ofstream fw ("parentou.txt");

int Push (char let){     //-- inserisce nella Pila ESPR
    if (Testa == 0){
       return -1;          //-- errore: Pila piena
    }
    Testa--;//--decremento dell'indice, Testa scende
    ESPR[Testa] = let;
    return 0;
}

int Pop (char& LTesta){//-- preleva dalla Pila ESPR
    if (Testa == MAX) {//-- l'elem.in Testa,l'ultimo
       return -1;          //-- errore: Pila vuota
    }        //-- ad ogni richiamo di Pop espone a video
    cout<<"\nESPR=";   //-- il contenuto attuale di PILA
    for(int k = MAX-1; k >= Testa; k--)
       cout<<ESPR[k];
    LTesta = ESPR[Testa];
    Testa++; //--incremento,
    return 0; //--Testa risale
}
int main(){
  char LET, letPila;
  int N, esito=0;

   fr >> N;
   for (int i=0; i<N && esito==0; i++){
       fr >> LET;
       if (LET == '{'  ||  LET == '['
       ||  LET == '('  ||  LET == '<')
                          //--- se parentesi aperta
          esito = Push(LET);
       else               //--- se parentesi chiusa
       {  esito = Pop (letPila);
          if (esito == 0)
            if (LET == '}' && letPila == '{'
            ||  LET == ']' && letPila == '['
            ||  LET == ')' && letPila == '('
            ||  LET == '>' && letPila == '<')
                esito= 0;
          else
                esito=-1;
       }
   }
   if ( esito < 0
   ||   Testa < MAX )
       fw << "malformata";
   else
       fw << "corretta";
   fr.close();
   fw.close();
   return 0;
}
```

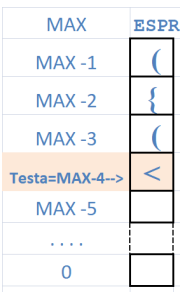
valore iniziale di **Testa**



Testa alla ricezione della prima parentesi **(**



Testa alla ricezione della prima parentesi <



Il programma deve controllare la correttezza di una sequenza di parentesi di vari tipi:

angolate <> , tonde () , quadre [] , graffe {}

Letto dal file di input **parentin.txt** il numero **N** di parentesi da controllare e poi la sequenza degli **N** caratteri successivi (le parentesi), se la sequenza presenta errori il programma deve scrivere sul file di output **parentou.txt** la frase "malformata", altrimenti la stringa "corretta".

Non va rispettata una gerarchia tra i vari tipi di parentesi, ma solo che ***una parentesi chiusa sia preceduta da una parentesi aperta*** dello stesso tipo.

esempi di **sequenze corrette**:

12 ()([]{(<>)})	14 {{{[[(())]]}}}

52 <{{{{{[[[[{{{{{{[[[[<<<<(())>>>>]]]]}}}}}}]]]]}}}}}>

esempi di **sequenze malformate**:

12 ([{()()}]({	4)][(	1 (
5 <({})	4 ()][	1 }

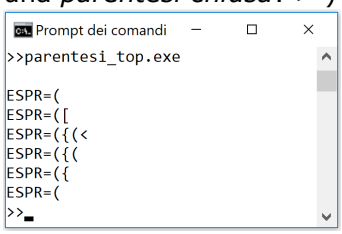
Il programma gestisce una **PILA mediante** il **vettore** di caratteri **ESPR**; legge dal file di input il numero **N** (12 nel primo esempio) di parentesi da controllare, e poi legge gli **N** valori successivi.

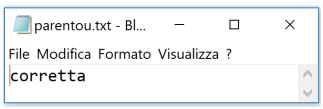
Una ***parentesi aperta*** viene inserita nell'array **ESPR** (mediante la funzione **Push**) nella posizione indicata da **Testa-1** (**Testa** indica l'ultimo elemento inserito in PILA e inizialmente, impostato a **MAX**, indica l'estremo superiore **esterno al vettore**: Pila vuota).

Per una ***parentesi chiusa*** viene prelevata l'ultima parentesi inserita nella **PILA** (mediante la funzione **Pop**) che viene poi controllata dal main per la corrispondenza di tipo (il parametro **LTesta** è passato per riferimento (**&**) quindi il main riceve nel parametro attuale **letPila** il valore prelevato dalla PILA dalla funzione **Pop**).

Il valore **-1** restituito dalla funzione **Push** indica che la **PILA** è già piena e che non si possono inserire altre *parentesi aperte*: < ([{ .

Il valore **-1** restituito dalla funzione **Pop** indica che la **PILA** è vuota anche se nel file di input si presenta una *parentesi chiusa*: >)] } .





^ file di output

← output a video per la sequenza con N=12