Esempio di gestione di una PILA mediante Array oppure *container Stack*

prof.ssa P.Grandillo

ARRAY *pila_array.ccp*

```
#include <iostream>
#include <fstream>
using namespace std;
#define MAX 10
    int Testa=MAX;
    char  ESPR  [MAX];          //-- dichiarazione  array ESPR
    ifstream finp  ("parentin.txt");
void scriviPila (char x) {
     cout<<"ESPR=";     //-- stampa tutti gli elementi nella Pila
     for(int k = MAX-1; k >= Testa; k--)
         cout << ESPR [k];           //-- elemento nella Pila
     if (x == 'p')
         cout<<"\n\n errore: Pila piena";
}
int Push  (char let){
     if (Testa == 0){          //-- errore: array ormai pieno
         scriviPila ('p');
         return -1;
     }
     Testa--;
     ESPR [Testa] = let;         //-- inserimento nella Pila
     return 0;
}
int Pop  (char& LTesta){
     if (Testa == MAX) {           //-- errore: Pila vuota
         cout<<"\n\n errore: Pila vuota";
         return -1;
     }
     scriviPila ('n');                    //-- OK, Pila non vuota
     LTesta = ESPR[Testa];//-- salvato in LTesta ultimo elem.
     Testa++;        //-- aggiornam. indice Testa a penult.elem.
     return 0;
}
```

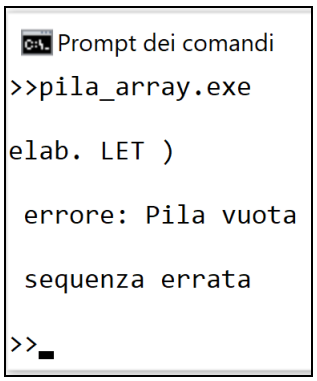
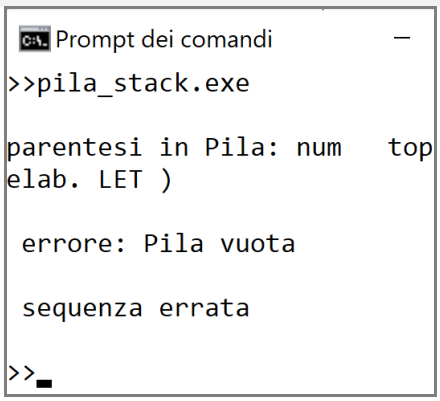
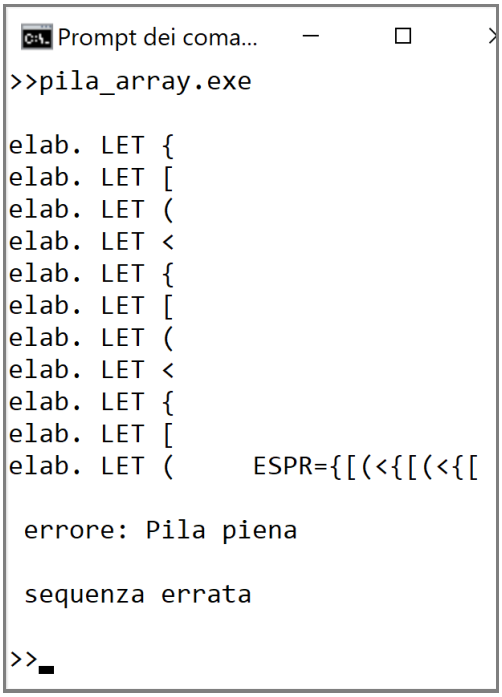
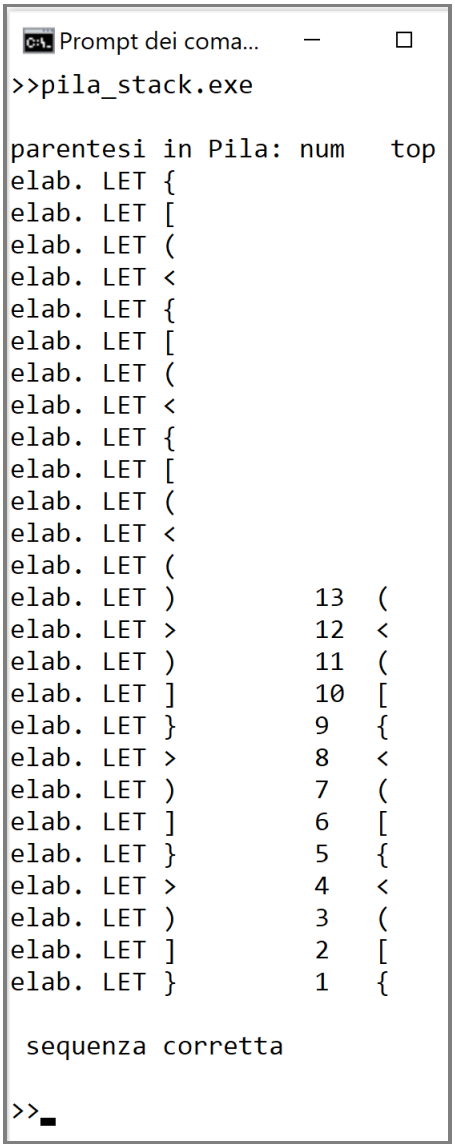
container STACK *pila_stack.ccp*

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

  stack <char> ESPR;          //-- dichiarazione  stack ESPR

  ifstream finp  ("parentin.txt");

void scriviPila () {                                  //-- stampa:
    cout << "     " << ESPR.size(); //-- numero elementi pila
    cout << '\t'    << ESPR.top ();     //-- ultimo elemento
    //-- si può accedere solo all'ultimo elemento inserito nella Pila
}



int Push  (char let){

    ESPR.push (let);                 //-- inserimento nella Pila
    return 0;
}


int Pop  (char& LTesta){

    if ( ESPR.empty() ) {         //-- errore: Pila vuota
        cout<<"\n\n errore: Pila vuota";
        return -1;
    }
    scriviPila ();                          //-- OK, Pila non vuota
    LTesta = ESPR.top();  //-- salvato in LTesta ultimo elem.
    ESPR.pop();              //-- rimosso ultimo elem. dalla Pila
    return 0;
}
```

Esempio di gestione di una PILA mediante Array oppure *container Stack*


```
int main(){
  char LET, letPila;   //-- LET parentesi letta da file
  int N, esito=0;      //-- letPila prelevata da array ESPR
    finp >> N;
    for (int i=0; i<N && esito==0; i++){
        finp >> LET;
        cout<<"\nelab. LET " << LET <<'\t';
        if (LET == '{' || LET == '['
        || LET == '(' || LET == '<') //-- parentesi aperta
            esito = Push(LET);       //-- inserimento in Pila
        else {                                //-- parentesi chiusa
            esito = Pop (letPila);    //-- prelievo da Pila
            if (esito == 0)           //-- e controllo
              if (LET == '}'   && letPila == '{'
              ||  LET == ']'   && letPila == '['
              ||  LET == ')'   && letPila == '('
              ||  LET == '>'   && letPila == '<')
                esito = 0;
            else
                esito = -1;
      }
    }
    if ( esito < 0)
        cout << "\n\n sequenza errata";
    else
      if (Testa < MAX ) {
        cout << "\n\n sequenza errata, Pila piena\n\n\t\t";
        scriviPila ('e');
      }
      else
        cout << "\n\n sequenza corretta";
    cout << endl;
    finp.close();      } //-- FINE main
```

```
int main(){
  char LET, letPila;
  int N, esito=0;
    cout << "\nparentesi in Pila: num \t top";
    finp >> N;
    for (int i=0; i<N && esito==0; i++){
        finp >> LET;
        cout<<"\nelab. LET " << LET <<'\t';
        if (LET == '{' || LET == '['
        ||  LET == '(' || LET == '<')
            esito = Push(LET);
        else {
            esito = Pop (letPila);
            if (esito == 0)
              if (LET == '}' && letPila == '{'
              ||  LET == ']' && letPila == '['
              ||  LET == ')' && letPila == '('
              ||  LET == '>' && letPila == '<')
                esito = 0;
              else
                esito = -1;
        }
    }
    if ( esito < 0)
        cout << "\n\n sequenza errata";
    else
      if ( ! ESPR.empty() ) {   //-- errore: Pila NON vuota
        cout <<"\n\n sequenza errata, Pila piena\n\n\t\t";
        scriviPila ();
      }
      else
        cout << "\n\n sequenza corretta";
    cout << endl;
    finp.close();    } //-- FINE main
```

OUTPUT con *pila_array.exe*	**OUTPUT con *pila_stack.exe***
caso 1 sequenza **errata**	4 **)[](**
	
caso 2 - sequenza **corretta** ma pila piena (non gestibile con array di soli 10 elementi)	26 **{[(<{[(<{[(<()>)]}>)]}>)]}**
	

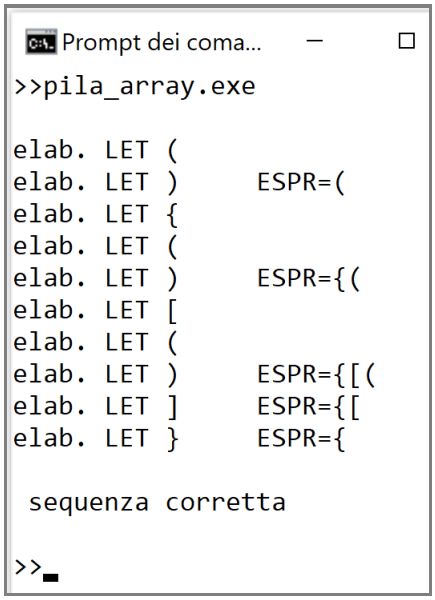
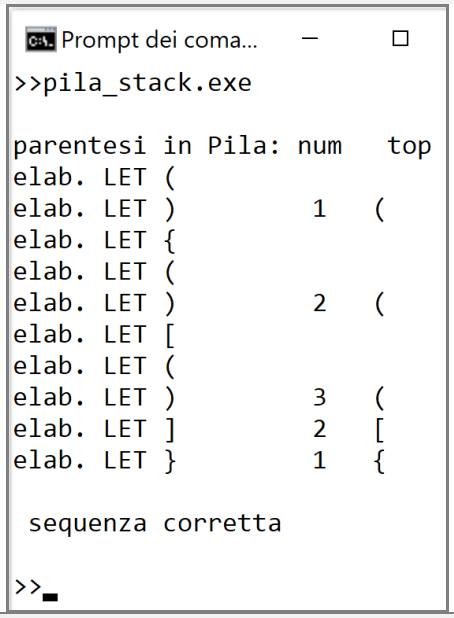
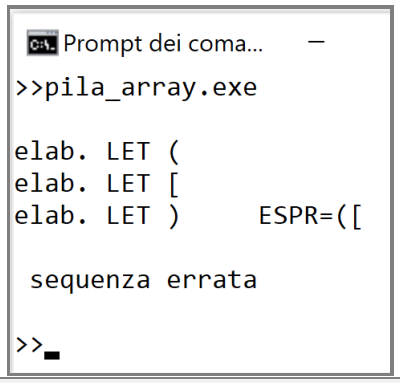
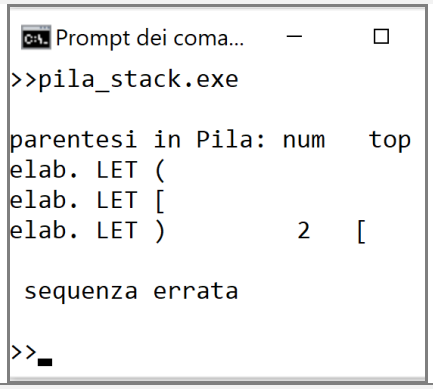
NOTA:

*con il container **stack** non è possibile accedere alle informazioni contenute nella Pila, ma solo all'ultimo elemento inserito;*

*l'accesso si effettua con il **metodo top()** esempio:* **ESPR.top()**

per approfondimenti consultare :
http://www.cplusplus.com/reference/stack/stack/

 Esempio di gestione di una PILA mediante Array oppure *container Stack*

OUTPUT con *pila_array.exe*	**OUTPUT con *pila_stack.exe***
caso 3 sequenza **corretta**	10 **(){(){[()]}**
	
caso 4 sequenza **errata**	4 **([)]**
	
caso 5 sequenza **errata**	12 **([{()()}](){**