UNIVERSITAT DE BARCELONA

Sessió 4. Teoricopràctica

Estructura de Dades
Curs 2020-2021

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

Contingut

1. Exercici PilaEstatica

Exercici PilaEstatica

PilaEstatica

- En aquest exercici es vol implementar una pila estàtica
- L'especificació les pròpies d'una pila
- La representació serà un array estàtic, amb un màxim definit a la variable DEFAULT_MAX_STACK
- La implementació en C++ sense usar templates

PilaEstatica.h

- Comenceu fent una clase PilaEstatica i poseu:
  - Les operacions de la pila:
    - Push, Pop, isEmpty, size, isFull, top, print
    - També afegiu el constructor per defecte, el constructor còpia i el destructor
  - Els atributs:
    - L'array per guardar els elements de la seqüència
    - Una variable per saber on està el cim de la pila

PilaEstatica.h (Solució)

```
#include <iostream>
#include <stdexcept>

using namespace std;

class PilaEstatica {
    private:
        enum { DEFAULT_MAX_STACK = 30};

    int _pila[DEFAULT_MAX_STACK]; // stack
    int _cim;     // _top

public:
    PilaEstatica(); // constructor
    PilaEstatica(const PilaEstatica& orig); // copy constructor
    virtual ~PilaEstatica(); // destructor // NO NECESSARI
    int size () const;      // return the size
    bool isEmpty() const;     // return TRUE if the stack is empty
    bool isFull() const;
    const int& top() const ; // return the top element of stack
    void push(const int& e); // introduce an element in the stack
    void pop() ;             // remove top element from the stack
    void print() const;      // print the contents of the stack
};
```

PilaEstatica.cpp

- Implementeu cadascuna de les operacions del TAD PilaEstatica en aquest ordre:
  - constructor per defecte
  - constructor còpia
  - size
  - isEmpty
  - isFull
  - top
  - push
  - pop
  - print

PilaEstatica.cpp (solució)

```
PilaEstatica::PilaEstatica( )
{
    this->_cim = -1;
}

PilaEstatica::PilaEstatica(const PilaEstatica& orig)
{
    for (int i = 0;  i < orig.size(); i++)
    {
        this->_pila[i] = orig._pila[i];
    }
    this->_cim = orig._cim;
 }
```

PilaEstatica.cpp (solució)

```
int PilaEstatica::size () const {return (_cim+1); }

bool PilaEstatica::isEmpty() const { return (_cim<0); }
// return TRUE if the stack is empty

bool PilaEstatica::isFull() const {
   return ( (_cim+1) == this->DEFAULT_MAX_STACK);
}

const int& PilaEstatica::top() const {
  if (isEmpty())  throw out_of_range("top(): empty stack");
    return (_pila[_cim]);
}// return the top element of stack
```

PilaEstatica.cpp (solució)

```
void PilaEstatica::push(const int& e) {
  if (size() == this->DEFAULT_MAX_STACK)
      throw  out_of_range("push():  full stack");
  _pila[++_cim] =e;
}// introduce an element in the stack

void PilaEstatica::pop() {
 if (this->isEmpty())
       throw out_of_range("pop(): empty stack");
 --_cim;
}// remove top element from the stack
```

PilaEstatica.cpp (solució)

```
void PilaEstatica::print() const  {
    if  (this->isEmpty())
      cout << "Stack =[]" << "  size " << size() << endl;
    else   {
        cout << "Stack=[" ;
        for (int i = 0; i<=_cim;i++){
            if (i != _cim ) cout << _pila[i] << " ," ;
            else cout << _pila[i] ;
        }
        cout << "] size " << size() << " " << endl;
    }
}
```

main.cpp

- Implementeu un main que us permeti utilitzar la pila i veure si us funciona bé

PilaEstatica.cpp (solució)

```
int main() {
    try{
      int valors[4]  = {1, 2, 3, 4};  PilaEstatica pila;
      for (int i = 0 ; i < 4; i++) { pila.push(valors[i]); }
        pila.print();   int i  = 1;
        while (!pila.isEmpty()){
          cout << " top " << i << " es: " <<  pila.top() << endl;
          pila.pop();  i++;
        }
        pila.print();
    }
    catch (exception const& ex) {
         cerr << "Exception: " << ex.what() <<endl;
         return -1;
        }
    return 0;
}
```