Pràctica 1. Estructures de dades bàsiques. Piles i cues

1. Introducció

***Objectius**:* Familiaritzar-se amb les estructures de piles i cues

Temes de teoria relacionats amb la pràctica: Tema 3 Estructures de dades lineals bàsiques

2. Enunciat

Exercici 1. Definir i implementar el TAD (tipus abstracte de dades) **ArrayStack** (pila en array) de caràcters amb els següents membres i mètodes:

```
class ArrayStack{

private:

      int MAX_STACK=100;

      std::vector<char> data = std::vector<char>(MAX_STACK);

      int t;                  // top de la pila

public:

      ArrayStack();           //inicialitza una pila, buida

      bool empty ();          //indica si la pila és buida

      bool full ();           //indica si la pila és plena

      void push (const char item);

      //introdueix un element a la pila,
      // i retorna una excepció si no ha estat possible

      void pop ();

      //treu un element de la pila,
      //i retorna una excepció si no ha estat possible

      char top();             //retorna l'element del top de la pila
      // retorna una excepció si no ha estat possible

      void print();           // imprimeix tot el contingut de la pila

};
```

Fer un **main.cpp** que posi alguns caràcters a la pila, en tregui uns altres, en torni a posar i finalment buidi la pila.

Modificar l'anterior main.cpp per, usant la pila avaluar una expressió numèrica i verificar si els parèntesi, claus i claudàtors estan ben aparellats. La introducció de les expressions serà per teclat.

Per exemple, el programa hauria de mostrar el següent:

Introdueix una expressió numèrica
4 * [5 -3] * (4 – 3 – 2 + [4 +1])
L'expressió està ben aparellada

Introdueix una expressió numèrica
4 * [2 * (4 *6 + [4 +1])
L'expressió està mal aparellada

Per fer aquest exercici segurament necessitaràs fer servir algunes de les funcions de string i de vector.
Recorda a fer: `#include <string>`
I que pots recórrer l'string amb un for, segons el següent esquema:

```
string cadena
for (char c: cadena) {
     ...
}
```

Recorda a fer : `#include <vector>`
I que pots reservar espai amb: `vector<type> nom(SIZE);` on type és el tipus contingut dins el vector i SIZE la mida a reservar.

A més a més, contesta a les següents preguntes: Com has comprovat amb una pila que l'expressió està ben aparellada? Quins errors controles i com ho fas?

Exercici 2. Definir i implementar el tipus el TAD **ArrayQueue** (cua en array) d'enters amb una capacitat de MAX_QUEUE elements (MAX_QUEUE=100). El TAD ha de permetre executar el següent codi:

```
ArrayQueue MevaCua;
cout<< «Mida actual de la cua: » << MevaCua.size() << endl;
cout<< «Encuem 3 elements a la cua... » << endl;
MevaCua.enqueue(1); MevaCua.enqueue(2); MevaCua.enqueue(3);
MevaCua.print();
cout <<  «Cua plena (0:no, 1:si): » << MevaCua.full() <<endl;
cout<< «Treiem 1er element de la cua: » << MevaCua.dequeue() << endl;
MevaCua.print();
cout<< «Treiem 2on element de la cua: » << MevaCua.dequeue() << endl;
```

```
cout<< «Encuem 2 elements a la cua... » << endl;
MevaCua.enqueue(4); MevaCua.enqueue(5);
cout<< «Treiem 3er element de la cua: » << MevaCua.dequeue() << endl;
cout<< «Mida actual de la cua: » << MevaCua.size() << endl;
cout <<  «Cua buida (0:no, 1:si): » << MevaCua.empty() <<endl;
```

Crear el **main.cpp** que contingui el codi anterior. Addicionalment, en el main heu de recollir les excepcions i gestionar-les.

A més a més, contesteu a les següents preguntes: Quina és la sortida del main? Quins errors controleu dels que es poden produir quan es demana afegir o eliminar elements de la cua? Què feu per controlar els errors i com ho feu?

Exercici 3. (**OPCIONAL**) Definir i implementar el TAD (tipus abstracte de dades) **ArrayStack** genèric, que es correspon amb un pila representada en array (en aquest cas la seva representació és en array dinàmic) i definida amb templates tal i com es mostra a continuació:

```
#ifndef ARRAYSTACK_H
#define ARRAYSTACK_H

#include <cstdlib>
#include <iostream>
#include <string>
#include <stdexcept>

using namespace std;

template <class E>
class ArrayStack {
    enum { DEF_CAPACITY = 100};
public:
    ArrayStack(int cap = DEF_CAPACITY); // constructor
    ArrayStack(const ArrayStack& orig); // copy constructor
    virtual ~ArrayStack(); // destructor
    int size () const;     // return the size
    bool empty() const;    // return TRUE if the stack is empty
    bool full() const;     // return TRUE if the stack is full
    const E& top() const ; // return the top element of stack
    void push(const E& e); // introduce an element in the stack
    void pop() ;           // remove top element from the stack
    // ... housekeeping functions
    void print() const;

private:
    E* S;
    int capacity;
    int t;

};
```

3. Lliurament

A partir de la descripció del problema, es demana:

- Escriure una memòria explicativa que inclogui els següents punts:
    - Portada amb: número de pràctica i títol complet, nom, cognom, NIUB de cada membre del grup, professor/a de pràctiques, número de parella assignat i grup de pràctiques al que assistiu (A, B, C, F)
    - Comentaris de cada exercici: observacions, decisions preses i resposta a les preguntes plantejades, si n'hi ha.

  La memòria s'entregarà en un document memoriaPX.pdf on X és el número de la pràctica.
- Implementar els exercicis en C++ . Lliurar el codi C++ corresponent als vostres exercicis en una carpeta anomenada codi, amb una subcarpeta per a cada exercici.

Com a màxim el dia del lliurament es penjarà en el campus virtual un fitxer comprimit **en format ZIP** amb el nom del grup (A, B, C o F), el número de parella (ParX), el nom dels dos membres del grup i el numero de la pràctica com a nom de fitxer. Per exemple, **GrupA_Par2_BartSimpsonLisaSimpson_P1.zip**, on Par2 indica que és la "parella 2" i P1 indica que és la "pràctica 1". El fitxer ZIP inclourà: la memòria i la carpeta amb tot el codi.

Els criteris per acceptar la pràctica són:

- La pràctica ha de funcionar en la seva totalitat.
- La pràctica ha de ser orientada a objectes.

IMPORTANT: La còpia de la implementació de la pràctica implica un zero a la nota de pràctiques de l'assignatura per les parelles implicades (tant la que ha copiat com la que ha deixat copiar).

4. Planificació

Lab4 i Lab5. Març 2016

Per aquesta pràctica els professors proposen la següent planificació:

- **Setmana 1**
    - Implementació de l'exercici 1.
    - Comentar el codi de l'exercici 1.
- **Setmana 2**
    - Implementació de l'exercici 2.
    - Comentar el codi de l'exercici 2.
    - Implementació de l'exercici 3.
    - Comentar el codi de l'exercici 3.
    - Realització de la memòria.

Cada setmana s'obrirà una tasca al campus virtual per dipositar la feina feta fins aquell moment.

Lliurament:

- Grups de dijous (B,C,D i F): dia 30 de març de 2016
- Grup de dimecres (E): dia 5 d'abril de 2016