Cues: especificació i implementació.

15 de març de 2014

1 Especificació de la classe queue (cua)

```
template <class T> class queue {
// Tipus de mòdul: dades

// Descripció del tipus: Estructura lineal que conté elements de tipus T i que
// permet consultar i eliminar només el primer element afegit

private:

public:

// Constructores

queue();
/* Pre: cert */
/* Post: El resultat és una cua sense cap element */

queue(const queue &original);
/* Pre: cert */
/* Post: El resultat és una cua còpia d'original */
// Destructora: Esborra automàticament els objectes locals en sortir
// d'un àmbit de visibilitat

~queue();

// Modificadores
```

```
void push(const T& x);
/* Pre: cert */
/* Post: El paràmetre implícit és com el paràmetre implícit original amb
   x afegit com a darrer element */

void pop();
/* Pre: El paràmetre implícit no està buit */
/* Post: El paràmetre implícit és com el paràmetre implícit original però
   sense el primer element afegit al paràmetre implícit original */

// Consultores

T front() const;
/* Pre: El paràmetre implícit no està buit */
/* Post: El resultat és el valor més antic afegit al paràmetre implícit */

bool empty() const;
/* Pre: cert */
/* Post: El resultat indica si el paràmetre implícit és buit o no */
};
```

2 Implementació d'una cua genèrica

2.1 .hpp

```
#ifndef _QUEUE_PRO2_

#define _QUEUE_PRO2_

#include <cassert>
#include <vector>

using namespace std;

template <class T> class queue {
// Tipus de mòdul: dades

// Descripció del tipus: Estructura lineal que conté elements de tipus T i que
// permet consultar i eliminar només el primer element afegit
```

```
private:
  static const int MAX_SIZE = 1000; // la implementació permet emmagatzemar
                                    // com a molt MAX_SIZE - 1 elements
  int head; // apunta la posició de elems que conté el primer element de la cua
  int tail; // apunta a la primera posició lliure de elems per col·locar
            // elements a la cua
  vector<T> elems;
public:

// Constructores

queue()
/* Pre: cert */
/* Post: El resultat és una cua sense cap element */
{
   head = tail = 0;
   elems = vector<T>(MAX_SIZE);
}

queue(const queue &original)
/* Pre: cert */
/* Post: El resultat és una cua còpia d'original */
// Destructora: Esborra automàticament els objectes locals en sortir
// d'un àmbit de visibilitat
{
  head = original.head;
  tail = original.tail;
  elems = original.elems;
}

~queue() {
}

// Modificadores

void push(const T& x)
/* Pre: El paràmetre implícit no està ple (not full()) */
/* Post: El paràmetre implícit és com el paràmetre implícit original amb
   x afegit com a darrer element */
```

```
{
   assert(not full());
   elems[tail] = x;
   tail = (tail + 1) % elems.size();
       // equivalent a:
       //   if (tail == int(elems.size()) - 1) tail = 0;
       //   else ++tail;
}

void pop()
/* Pre: El paràmetre implícit no està buit */
/* Post: El paràmetre implícit és com el paràmetre implícit original però
   sense el primer element afegit al paràmetre implícit original */
{
   assert(not empty());
   head = (head + 1) % elems.size();
       // equivalent a:
       //   if (head == int(elems.size()) - 1) head = 0;
       //   else ++head;
}

// Consultores

T front() const
/* Pre: El paràmetre implícit no està buit */
/* Post: El resultat és el valor més antic afegit al paràmetre implícit */
{
  assert(not empty());
  return elems[head];
}

bool empty() const
/* Pre: cert */
/* Post: El resultat indica si el paràmetre implícit és buit o no */
{
  return head == tail;
}

bool full() const
/* Pre: cert */
```

```
/* Post: El resultat indica si el paràmetre implícit és ple o no */
{
  return head == (tail + 1) % elems.size();
}

};

#endif
```