

Laboratori 2. Pràctica 2

Estructura de Dades

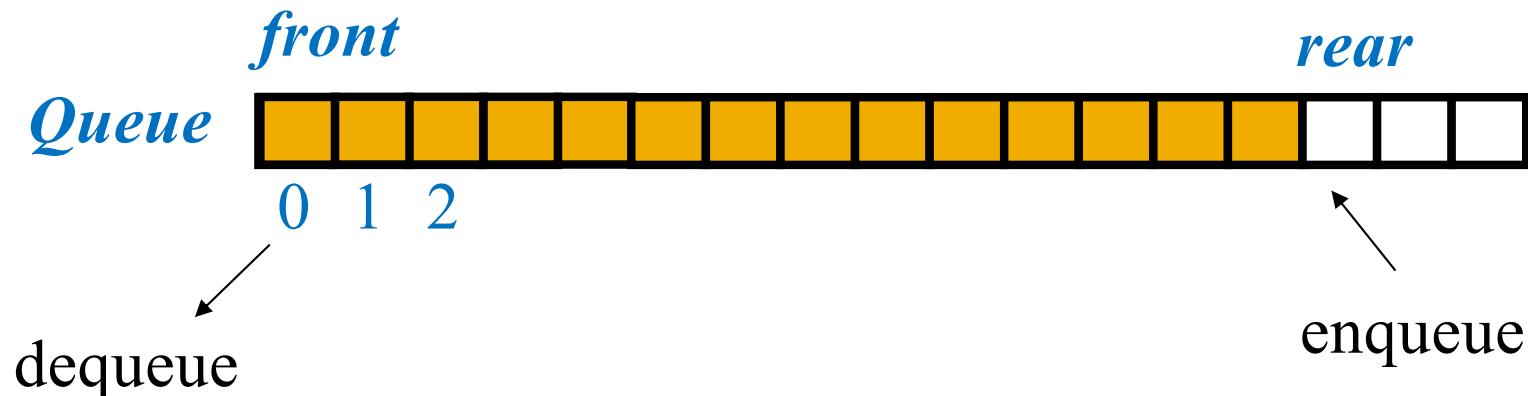
Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

Exercici 1

- **Objectiu:**
 - Definir i implementar el **TAD ArrayQueue**
- **Definir i implementar sense usar templates**
 - La representació del TAD es farà en arrays, usant un STL vector
 - Les operacions són les definides a l'enunciat
- **Fer un `main.cpp` que usi el TAD ArrayQueue i que contingui els dos casos de prova**

Queue

- Un Queue és una estructura (First-In First-Out)
- S'insereix pel final, i s'esborra per l'inici de la cua



ArrayQueue

- La implementació es farà usant STL vector
- Sobre el codi ...
 - Optimitzeu les operacions del TAD amb el menor cost computacional que sigui possible
 - Comenteu el codi (les operacions del TAD i el main)
 - Poseu noms entenedors a les variables i als mètodes
 - Identieu el codi adequadament

Laboratori 3. Pràctica 2

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

Exercici 2

- **Objectiu:**
 - Definir i implementar el TAD **LinkedQueue**
- **Definir i implementar usant templates**
 - La representació del TAD es farà en una estructura amb encadenaments simples
 - Les operacions són les definides a l'enunciat
- Fer un **main.cpp** que usi el TAD LinkedQueue i que contingui els dos casos de prova de l'exercici 1.
- S'han de controlar les excepcions.

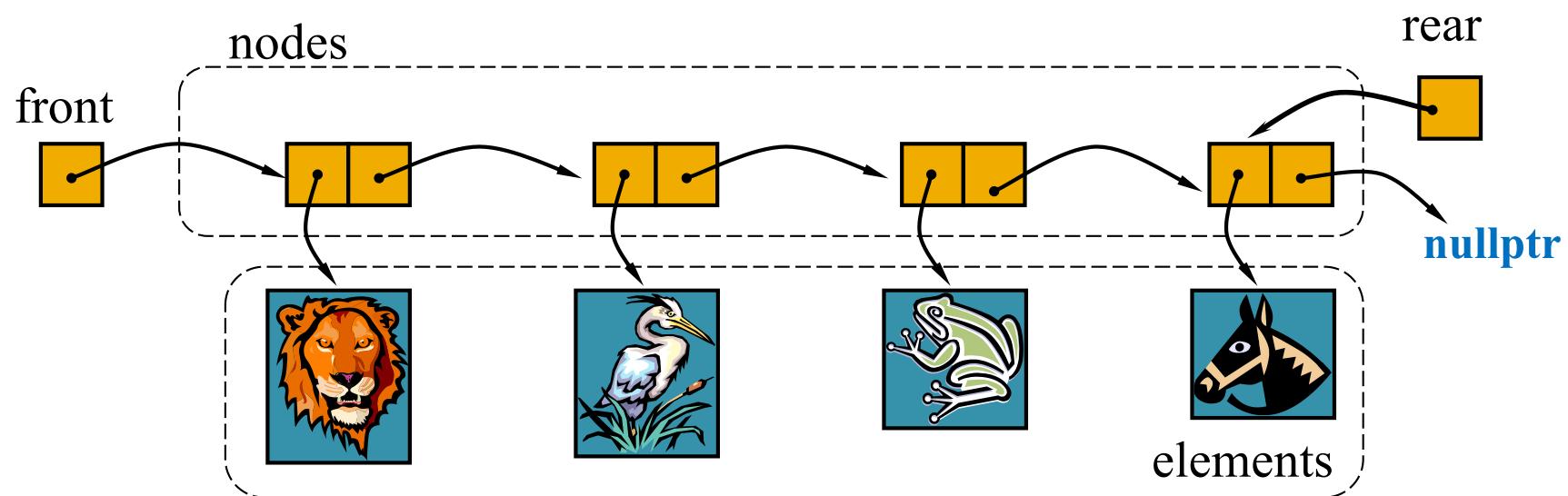
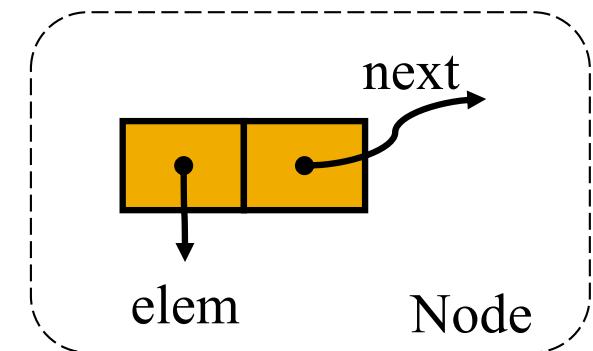
Exercici 3

- **Objectiu:**
 - Usar el **TAD LinkedQueue**
- **Es demana resoldre un problema d'una cua de sortida de vols utilitzant aquest TAD**
 - Fer el menú del programa
 - Implementar la càrrega dels fitxers de les entrades a la cua de vols
 - Implementar eliminar elements de la cua
 - Implementar inserir elements de la cua
- **S'han de controlar les excepcions**

Descripció de la Queue amb encadenaments simples

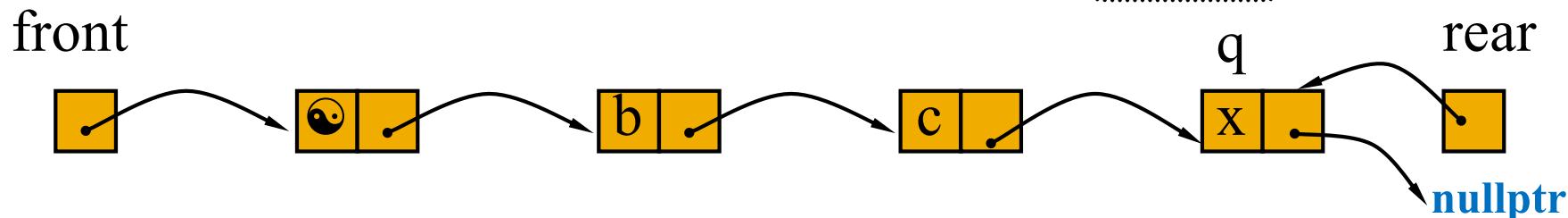
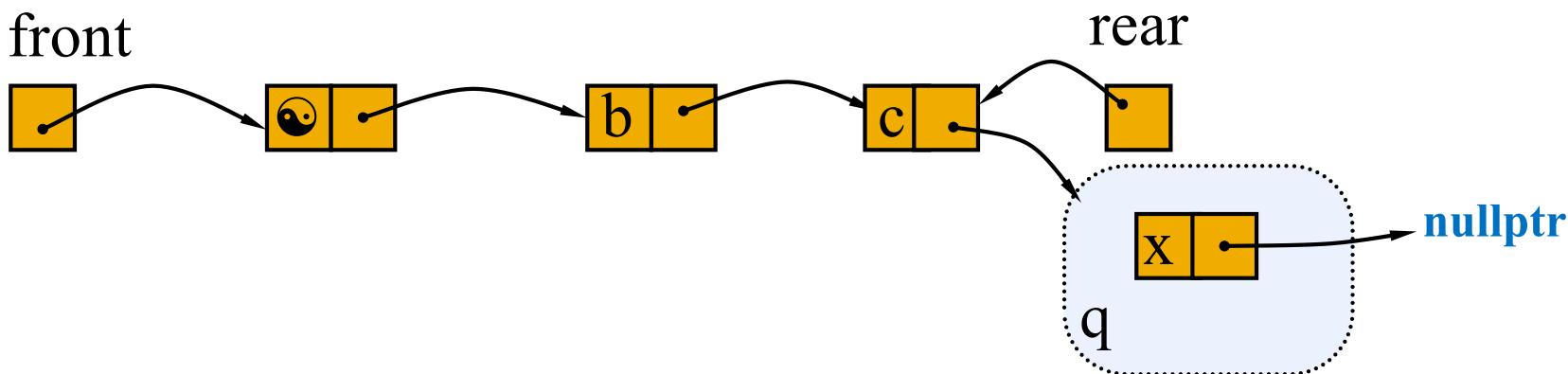
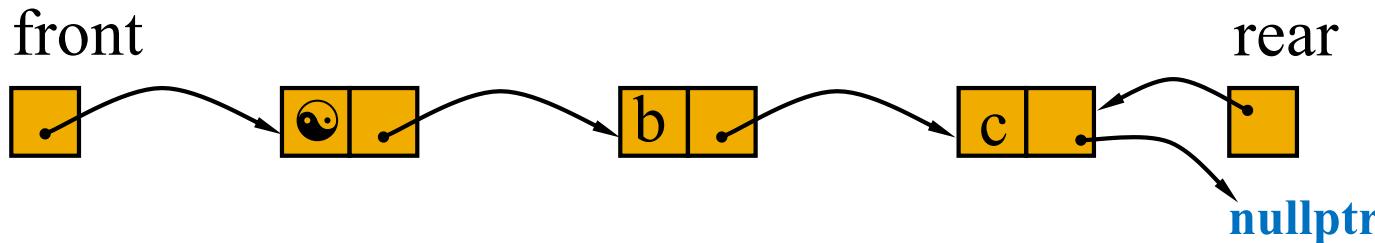
Queue amb encadenaments simples

- Queue amb encadenaments simples és la implementació natural del TAD Queue
- Els nodes implementen guarden:
 - element
 - link al següent node



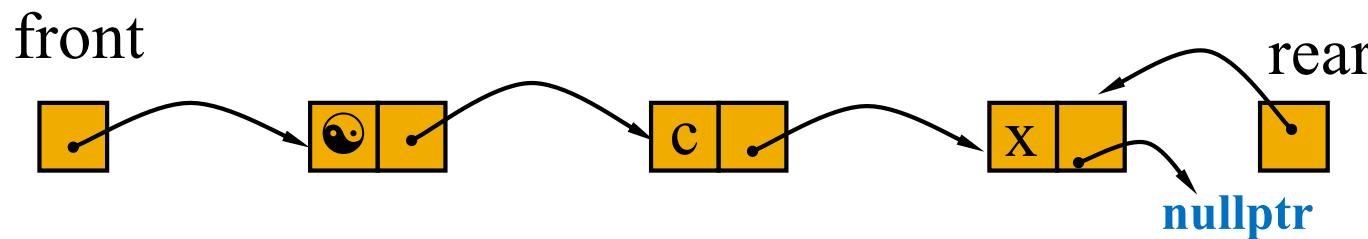
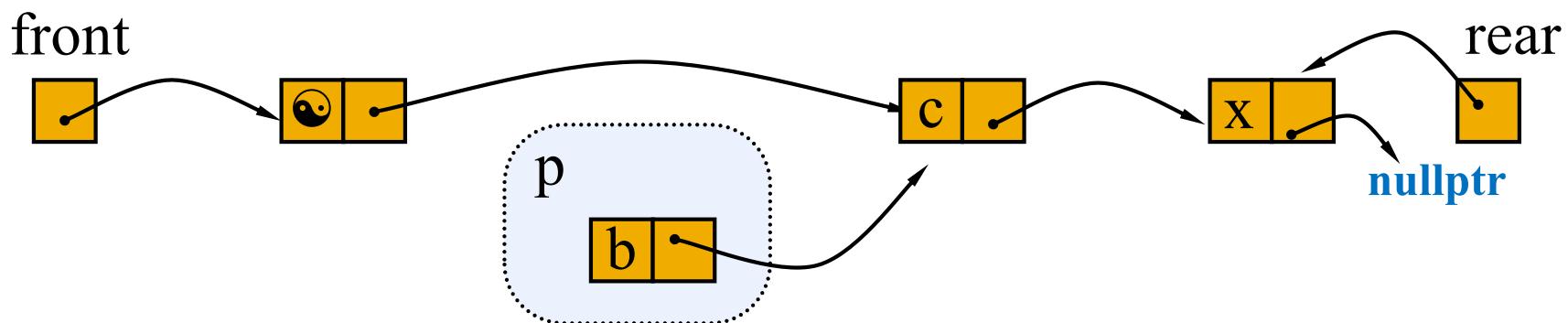
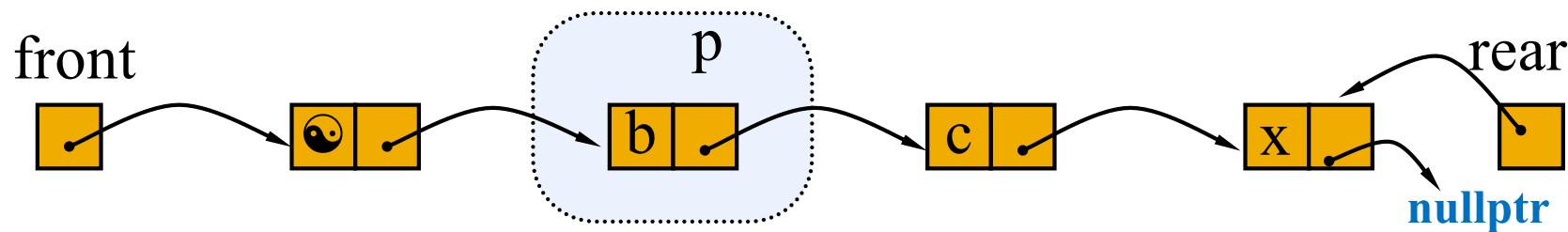
Inserció al final de la Queue

- Visualització de l'operació enqueue(x), la qual inserta x al final de la cua. (el sentinel·la és el node pintat com ☺)



Eliminar a l'inici de la Queue

- Visualització de l'operació `dequeue()`



Descripció de Templates en C++

Templates

- Els templates són útils per definir un únic codi en un conjunt de funcions relacionades (**function template**) o un conjunt de classes relacionades (**class template**)

```
template <class Type>
Type larger(Type x, Type y)
{
    if (x >= y)
        return x;
    else
        return y;
}
```

cout << larger(5, 6) << endl
cout << larger ('A', 'B') << endl;

Exemple funció template

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

template <class T>
T sum (T a, T b){
    T result;
    result = a + b;
    return result;
}

int main () {
    int i=5, j=6, k;
    double f=2.0, g=0.5, h;
    k=sum<int>(i,j);
    h=sum<double>(f,g);
    cout << k << '\n';
    cout << h << '\n';
    return 0;
}
```

Classe Template

```
template <class elemType>
class listType
{
public:
    bool isEmpty();
    bool isFull();
    void search(const elemType& searchItem, bool& found);
    void insert(const elemType& newElement);
    void remove(const elemType& removeElement);
    void destroyList();
    void printList();

    listType();

private:
    elemType list[100];    //array to hold the list elements
    int length;           //variable to store the number
                          //of elements in the list
};
```

Definició del tipus
genèric

Exemple Class Template

```
template<class ItemType>
class GList
{
public:
    bool IsEmpty() const;
    bool IsFull() const;
    int Length() const;
    void Insert( /* in */ ItemType item );
    void Delete( /* in */ ItemType item );
    bool IsPresent( /* in */ ItemType item ) const;
    void SelSort();
    void Print() const;
    GList();                                // Constructor
private:
    int      length;
    ItemType data[MAX_LENGTH];
};
```

*Template
parameter*

Instanciant un Class Template

Es poden crear llistes de diferents tipus

```
// Client code
GList<int> list1;
GList<float> list2;
GList<string> list3;

list1.Insert(356);
list2.Insert(84.375);
list3.Insert("Muffler bolt");
```

template argument

El compilador genera tres llistes diferents

```
GList_int list1;
GList_float list2;
GList_string list3;
```

- Amb Templates linkar el codi de manera diferent. Hi ha varíes maneres:
 - La més simple és posar el .h i la seva implementació juntes (no feu el .cpp, cal incloure el codi del .cpp al .h)

Exemple de Template

```
#ifndef COMPLEXE_H
#define COMPLEXE_Htemplate
<class T>class Complexe{
    private:
        T v_real;
        T v_imaginari;
    public:
        Complexe<T>(T r=0, T i=0);
        T real() const;
        T imaginari() const;
    };
    template <class T>
        Complexe<T>::Complexe(T r , T i) : v_real(r), v_imaginari(i){    }
    template <class T>
        T Complexe<T>::real() const{    return v_real;}
    template <class T>
        T Complexe<T>::imaginari() const{    return v_imaginari; }
#endif /* COMPLEXE_H */
```

Templates

- Al campus virtual trobareu:
 - un document anomenat *Exemple Templates Complex* que implementa l'exemple de Complex fet amb class template