Tema 2 – Exercici Opcional 2

- En els nostres programes haurem de gestionar diferents estructures dinàmiques enllaçades, cadascuna amb un tipus diferent, però totes amb necessitats similars a nivell de funcionalitat: inserir i eliminar elements, accedir als elements...
- En aquest exercici **implementarem la classe `Llista`** com un patró comú que permeti gestionar llistes enllaçades de qualsevol tipus amb mínims canvis.
- A la **llibreria estàndar**d de C++ hi ha una classe **`forward_list`** amb una funcionalitat similar:
  http://www.cplusplus.com/reference/forward_list/forward_list/

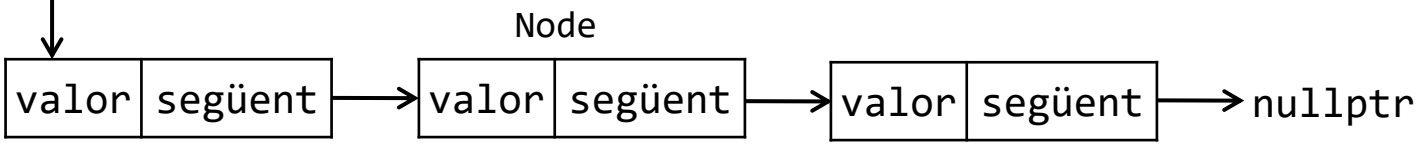
Tema 2 – Exercici Opcional 2

```
class Llista
{
public:
   Llista();
   Llista(const Llista& v);
   ~Llista();
   Llista& operator=(const Llista& v);
   Node* insereix(const Punt& pt, Node
   Node* elimina(Node* posicio);
   int getNElements();
   Node* getInici();
   bool esBuida();
private:
   Node* m_primer;
};
```

```
class Node
{
public:
   <tipus>& getValor();
   Node* getNext();
   void setValor(const <tipus>& valor);
   void setNext(Node* next);
 private:
   <tipus> m_valor;
   Node* m_next;
};
```

Canviant el tipus del valor podem implementar llistes que permetin guardar objectes de qualsevol tipus. En aquest exercici suposarem que la llista guarda objectes de la **classe Punt**.

Apuntador al primer element de l'estructura enllaçada.



Tema 2 – Exercici Opcional 2

```
class Llista
{
public:
   Llista();
   Llista(const Llista& v);
   ~Llista();
   Llista& operator=(const Llista& v);
   Node* insereix(const Punt& pt, Node* posicio);
   Node* elimina(Node* posicio);
   int getNElements();
   Node* getInici();
   bool esBuida();
private:
   Node* m_primer;
};
```

- `Llista();` → Constructor per defecte: inicialitza apuntador a nul.
- `Llista(const Llista& v);` → Constructor de còpia: duplica tots els nodes.
- `~Llista();` → Destructor: allibera tots els nodes de la llista.
- `Llista& operator=(const Llista& v);` → Operador assignació: duplica tots els nodes.
- `int getNElements();` → Retorna el número d'elements de la llista.
- `Node* getInici();` → Retorna un apuntador al primer element.
- `bool esBuida();` → Retorna si l'estructura està buida (no té cap element).

En aquest exercici **només heu d'implementar**:

- Mètode **`insereix`**
- Mètode **`elimina`**
- **Operador d'assignació**

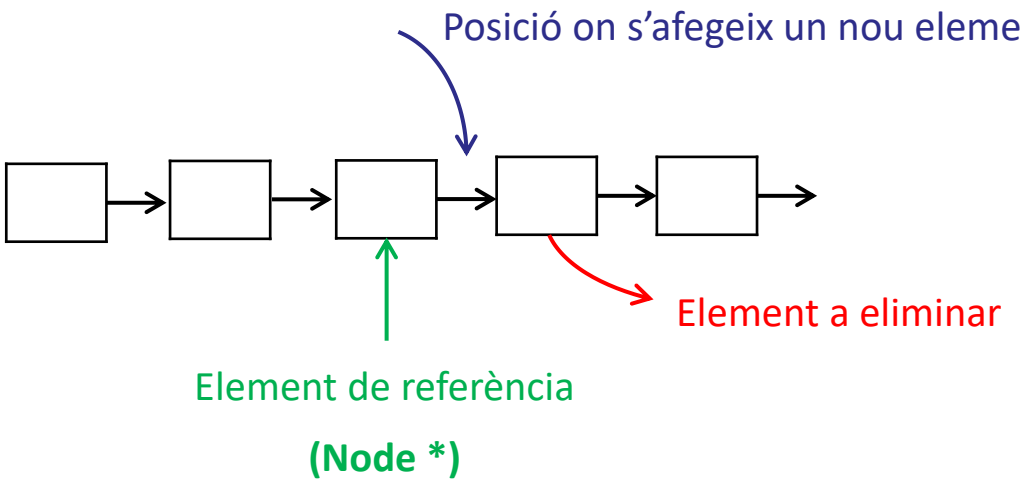
La resta de mètodes ja estan implementats

Tema 2 – Exercici Opcional 2

En una llista hem de poder:

- Afegir un element a una posició qualsevol de la llista.
  - L'element nou s'afegirà en la posició següent a la que ocupa un element de referència donat.
- Eliminar un element a una posició qualsevol de la llista.
  - L'element que s'eliminarà és el que ocupa la posició següent a un element de referència donat.

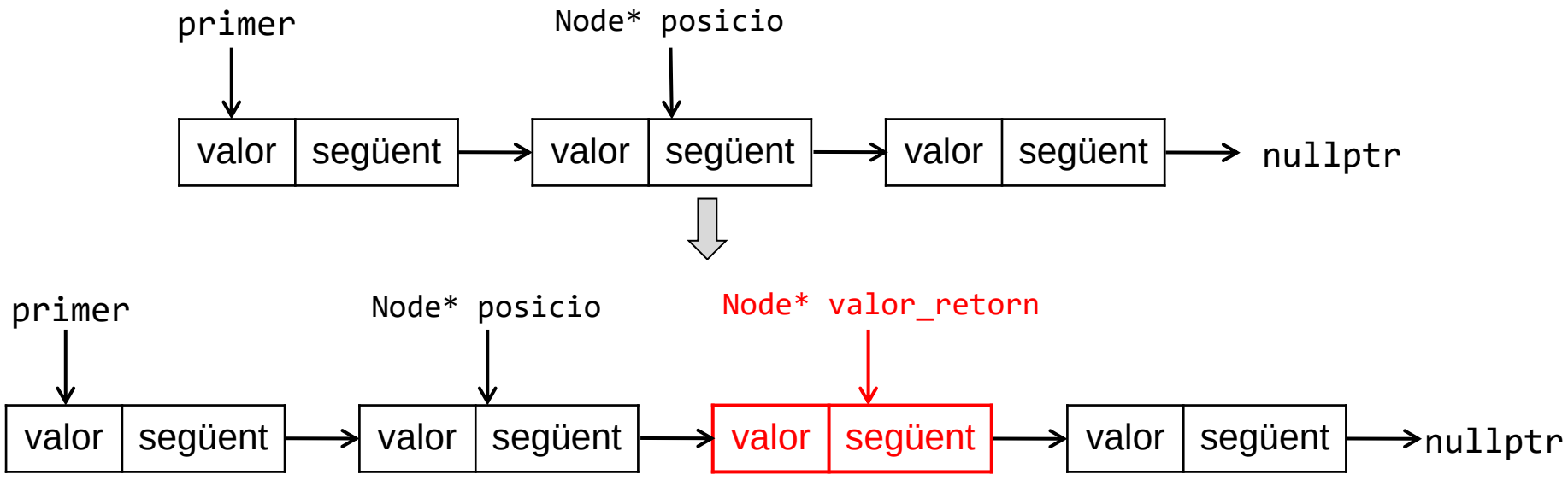
Necessitem algun mètode per identificar una posició qualsevol de la llista



Tema 2 – Exercici Opcional 2

```
Node* insereix(const <tipus>& valor, Node* posicio);
```

- Afegeix un element del tipus de la llista a la posició posterior a la que ens indica un apuntador de referència.
- Si l'apuntador és nul l'afegeix al principi de la llista.
- Retorna un apuntador al nou element que s'ha inserit.

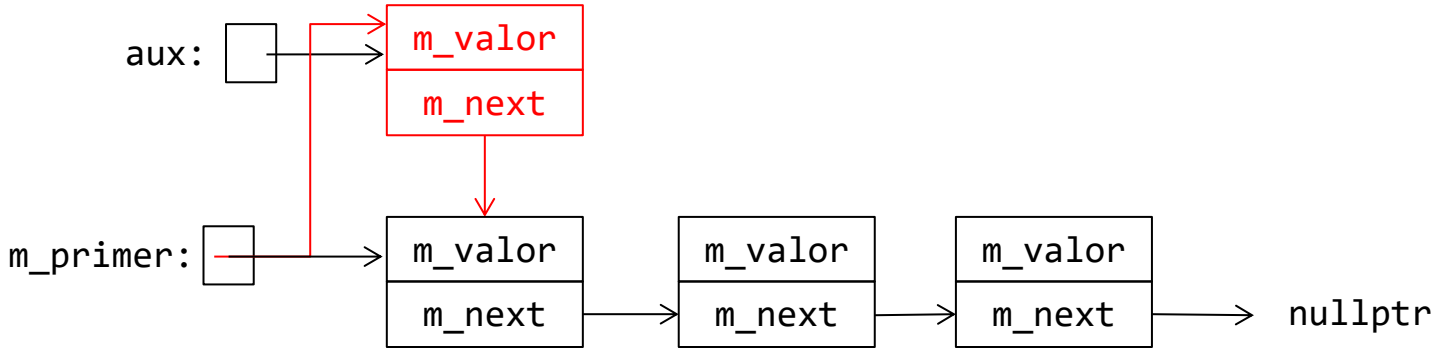


Tema 2 – Exercici Opcional 2

```
Node* insereix (const <tipus>& valor, Node* posicio);
```

Inserir al principi de la llista (`posicio == nullptr`)

1. Crear i inicialitzar el nou node.
2. Enllaçar nou node amb primer.
3. Modificar l'apuntador primer al nou node.

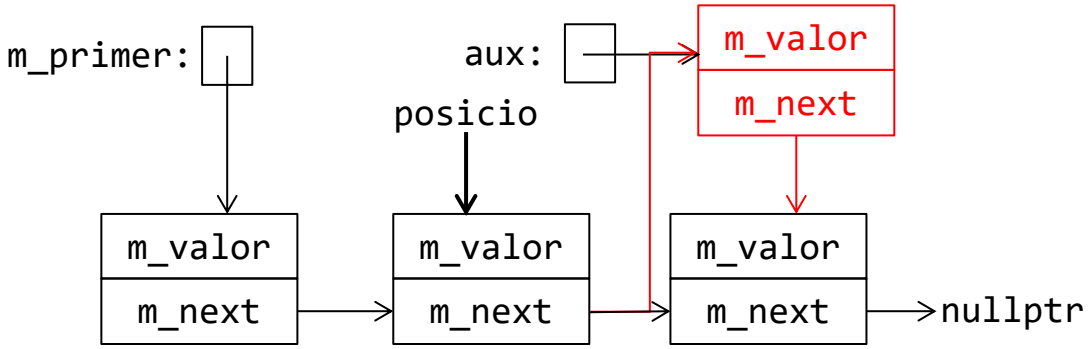


Tema 2 – Exercici Opcional 2

```
Node* insereix (const <tipus>& valor, Node* posicio);
```

Inserir al mig de la llista (`posicio != nullptr`)

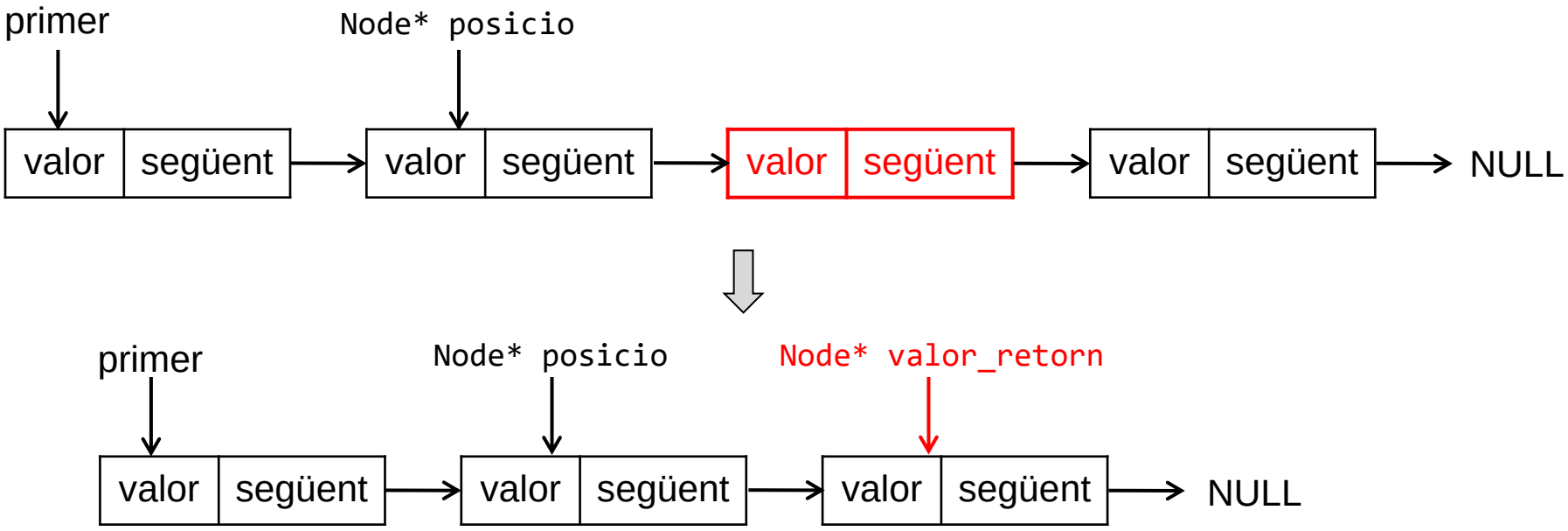
1. Crear i inicialitzar el nou node.
2. Enllaçar nou node amb el següent del node indicat per `posició.`
3. Modificar l'apuntador `m_next` del node indicat per `posició.`



Tema 2 – Exercici Opcional 2

```
Node* elimina(Node* posicio);
```

- Elimina l'element de la posició posterior a la que ens indica un apuntador.
- Si l'apuntador és nul elimina el primer element de la llista.
- Retorna un apuntador a l'element posterior al que s'ha eliminat.

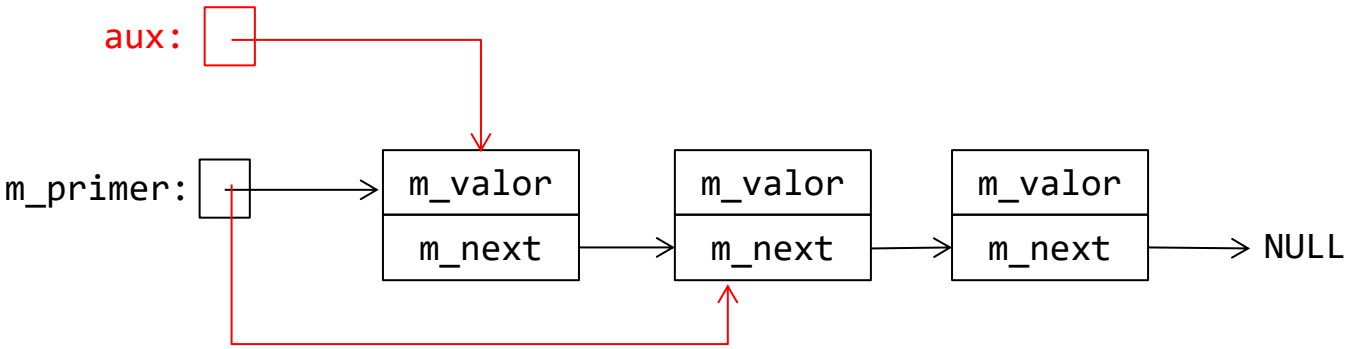


Tema 2 – Exercici Opcional 2

```
Node* elimina(Node* posicio);
```

Elimina el primer element de la llista (`posicio == nullptr`)

1. Guardar apuntador al primer element de la llista.
2. Modificar l'apuntador primer al node següent.
3. Eliminar el primer node.



Tema 2 – Exercici Opcional 2

```
Node* elimina(Node* posicio);
```

Eliminar un element del mig de la llista (`posicio != nullptr`)

1. Recuperar l'element següent del node apuntat per `posició.`
2. Enllaçar el node apuntat per `posicio` amb el següent del node eliminat.
3. Eliminar node següent `posició.`