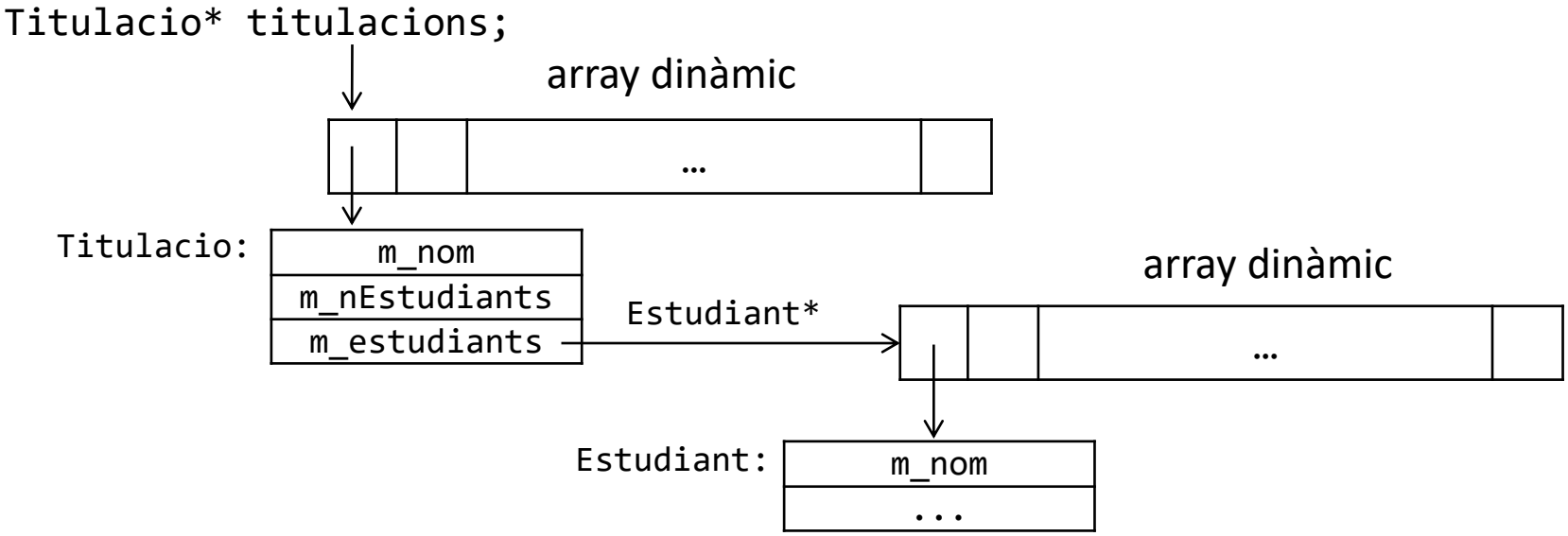
Tema 2 – Exercici Opcional 1



- En els nostres programes haurem de gestionar diferents arrays dinàmics, cadascun d'un tipus diferent, però tots amb necessitats similars a nivell de funcionalitat: inserir i eliminar elements, accedir als elements, gestió de la memòria dinàmica, ...
- La classe `Vector` que us proposem implementar en aquest exercici dóna un patró comú per dissenyar una classe que permeti gestionar arrays dinàmics de qualsevol tipus amb mínims canvis.
- Aquesta classe està inspirada en la classe `vector` de la llibreria estàndard de C++.

Tema 2 – Exercici Opcional 1

```
class Vector
{
public:
   Vector();
   Vector(const Vector& v);
   ~Vector();
   Vector& operator=(const Vector& v);
   bool insereix(const Punt& pt, int posicio);
   bool elimina(int posicio);
   int getNElements();
   int getMaxElements();
   Punt& operator[] (int posicio);
private:
   Punt *m_array;
   int m_nElements;
   int m_maxElements;
   Punt m_valorDefecte;
};
```

Apuntador per crear i gestionar l'array dinàmic.

Número d'elements que s'han inserit al vector.

Mida de l'array dinàmic. Número màxim d'elements que es poden inserir al vector.

Tema 2 – Exercici Opcional 1

```
class Vector
{
public:
   Vector();
   Vector(const Vector& v);
   ~Vector();
   Vector& operator=(const Vector& v);
   bool insereix(const Punt& pt, int posicio);
   bool elimina(int posicio);
   int getNElements();
   int getMaxElements();
   Punt& operator[] (int posicio);
private:
   Punt *m_array;
   int m_nElements;
   int m_maxElements;
   Punt m_valorDefecte;
};
```

Constructor per defecte. Número elements i mida array dinàmic incialitzats a 0. Apuntador a nul.

Destructor: ha d'alliberar l'array dinàmic.

Tema 2 – Exercici Opcional 1

```
class Vector
{
public:
   Vector();
   Vector(const Vector& v);
   ~Vector();
   Vector& operator=(const Vector& v);
   bool insereix(const Punt& pt, int posicio);
   bool elimina(int posicio);
   int getNElements();
   int getMaxElements();
   Punt& operator[] (int posicio);
private:
   Punt *m_array;
   int m_nElements;
   int m_maxElements;
   Punt m_valorDefecte;
};
```

Constructor de còpia i operador d'assignació: han de duplicar l'array dinàmic i copiar tots els elements.

Retorna el número d'elements reals del vector.

Retorna la mida de l'array dinàmic.

Retorna l'element que ocupa la posició de l'array indicada per paràmetre. Si la posició no és vàlida retorna el valor per defecte guardat a `m_valorDefecte`.

Tema 2 – Exercici Opcional 1

```
class Vector
{
public:
   Vector();
   Vector(const Vector& v);
   ~Vector();
   Vector& operator=(const Vector& v);
   bool insereix(const Punt& pt, int posicio);
   bool elimina(int posicio);
   int getNElements();
   int getMaxElements();
   Punt& operator[] (int posicio);
private:
   Punt *m_array;
   int m_nElements;
   int m_maxElements;
   Punt m_valorDefecte;
};
```

Insereix un nou element a la posició indicada.

Elimina l'element que està a la posició indicada.

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

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

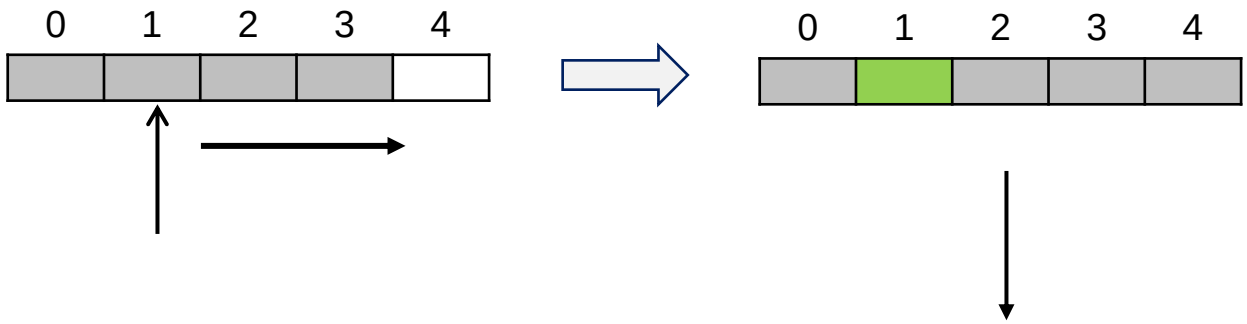
La resta de mètodes ja estan implementats

Tema 2 – Exercici Opcional 1

```
bool insereix(const Punt& pt, int posicio);
```

- Afegir un nou element a la posició indicada, desplaçant tots els elements posteriors una posició a la dreta.
- Retorna `false` si la posició no és vàlida (entre 0 i el número d'elements)

```
m_nelements = 4
m_maxElements = 5
posicio = 1
```



Què passa si ara volem afegir un nou element al vector?

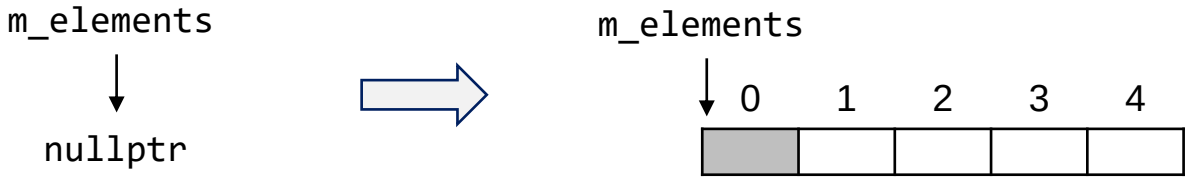
- No tenim espai a l'array per afegir nous elements.
- Hem de **redimensionar** l'array i reservar més espai de memòria per poder afegir els nous elements.

Tema 2 – Exercici Opcional 1

```
bool insereix(const Punt& pt, int posicio);
```

- **Redimensionar** l'array dinàmic, si no té mida suficient per poder inserir el nou element, seguint aquests criteris:
  - Si la mida de l'array dinàmic és 0 (primera inserció) es crearà l'array dinàmic amb una mida prefixada de 5 elements.

```
m_nelements = 0
m_maxElements = 0
posicio = 0
```

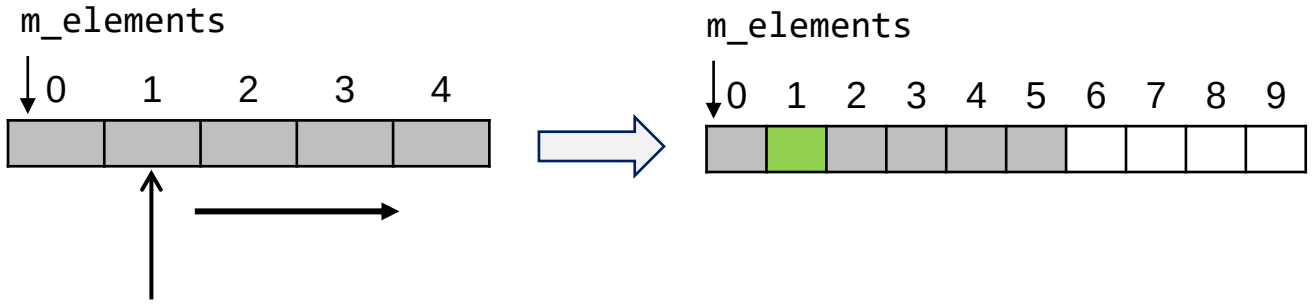


Tema 2 – Exercici Opcional 1

```
bool insereix(const Punt& pt, int posicio);
```

- **Redimensionar** l'array dinàmic, si no té mida suficient per poder inserir el nou element, seguint aquests criteris:
  - Si la mida de l'array dinàmic és diferent de 0 (insercions després de la primera) es redimensionarà la mida de l'array dinàmic al doble de la seva dimensió actual.
  - Redimensionar l'array implicarà reservar memòria per a un nou array amb la nova mida, fer la còpia de tots els elements de l'array antic al nou i alliberar l'array original. Les posicions noves de l'array queden buides per poder ser ocupades pels nous elements que es vagin inserint.

```
m_nelements = 5
m_maxElements = 5
posicio = 1
```

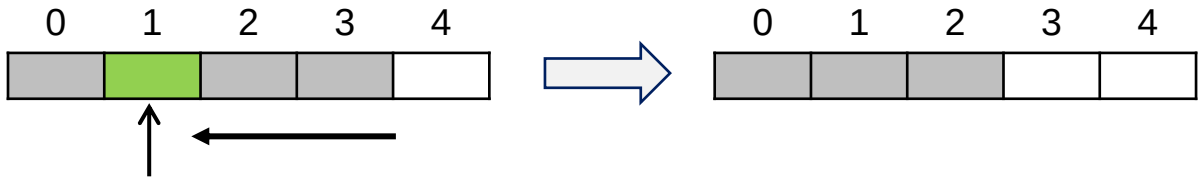


Tema 2 – Exercici Opcional 1

```
bool elimina(int posicio);
```

- Elimina l'element de la posició indicada, desplaçant tots els elements posteriors una posició a l'esquerra.
- Retorna `false` si la posició no és vàlida (entre 0 i el número d'elements)

```
m_nelements = 4
m_maxElements = 5
posicio = 1
```



Tema 2 – Exercici Opcional 1

```
bool elimina(int posicio);
```

- Redimensionar l'array dinàmic, si queda massa buit després d'eliminar, seguint aquests criteris:
  - Si el número d'elements del vector és inferior a una quarta part de la mida de l'array dinàmic: redimensionar l'array dinàmic a la meitat de la seva mida actual.
  - En cap cas, després de redimensionar la mida de l'array dinàmic pot ser inferior a la mida prefixada inicial de 5 elements.
  - Redimensionar l'array implicarà reservar memòria per a un nou array amb la nova mida, fer la còpia de tots els elements de l'array antic al nou i alliberar l'array original.