MP 20-21 Tema 1
Objectes i pas de paràmetres
Sessió 8: Programació orientada a objectes

Classes: tipus de mètodes

```
class Punt
{
public:
   Punt();
   Punt(float x, float y);
   ~Punt();

   void setX(float x);
   void setY(float y);
   void llegeix();

   float getX() const;
   float getY() const;
   void mostra() const;
   float distancia(Punt& p) const;
private:
   ...
```

Constructor/Destructor: són els mètodes que es criden quan es crea/destrueix un objecte de la classe

Modificadors: serveixen per canviar l'estat de l'objecte

Consultors: serveixen per consultar/recuperar l'estat de l'objecte. S'haurien de declarar com a `const`

- Tots els mètodes que no modifiquen l'estat de l'objecte s'han de declarar `const`
- Evita errors i facilita que es puguin declarar objectes constants de la classe
- Dins del codi dels mètodes `const` no podem modificar la representació interna (atributs) de la classe

Classes: pas de paràmetres per valor

```
class Punt
{
public:
   Punt();
   Punt(float x, float y);
   ~Punt();

   void setX(float x);
   void setY(float y);
   void llegeix();

   float getX() const;
   float getY() const;
   void mostra() const;
   float distancia(const Punt& p) const;
```

1. **Eficiència:** definim sempre els objectes de qualsevol classe com a paràmetres per referència. Així evitem fer una evitem fer una còpia de tot l'objecte.
2. **Pas per valor**: definim el paràmetre com a `const` per evitar que pugui ser modificat dins de la funció
3. Dins de la funció només podem utilitzar **p1** amb mètodes declarats com a `const`

Només mètodes `const`

```
void calculaDistancia(Punt p1)
{
   Punt p2;

   p1.mostra();
   p2.llegeix();
   float d = p1.distancia(p2);
   cout << "Distancia: " << d << endl;
};
```

```
void calculaDistancia(const Punt& p1)
{
   Punt p2;

   p1.mostra();
   p2.llegeix();
   float d = p1.distancia(p2);
   cout << "Distancia: " << d << endl;
};
```

Classes: utilització de mètodes i objectes const UAB

- Tots els mètodes d'una classe que no modifiquin la representació interna (atributs) de la classe s'han de declarar com a mètodes constants utilitzant `const`.
- En el pas de paràmetres per valor els objectes s'han de definir com a referència constant, `const` &.
- Pels objectes que s'hagin declarat com a constants només podem utilitzar els mètodes declarats també com a constants.

Exercici

Corregeix tots els errors que hi ha en el codi següent:

```
class Complex
{
public:
   void setReal(float real);
   void setImg(float img);
   void llegeix() const;
   void mostra() const;
   float getReal() const;
   float getImg() const;
   Complex suma(Complex& c);
   Complex resta(Complex& c);
   Complex multiplica(Complex& c);
};
```

```
int main()
{
   Complex a, b;
   char operacio;

   a.llegeix();
   b.llegeix();
   cin >> operacio;
   calcula(a, b, operacio);
   return 0;
}
```

```
void calcula(const Complex& c1, const Complex& c2, char operacio)
{
   if (operacio == '+')
     c1 = c1.suma(c2);
   else
     if (operacio == '-')
       c1 = c1.resta(c2);
     else
       c1 = c1.multiplica(c2);
   c1.mostra();
}
```

Exercici: solució

Corregeix tots els errors que hi ha en el codi següent:

```
class Complex
{
public:
   void setReal(float real);
   void setImg(float img);
   float getReal();
   float getImg();
   void llegeix();
   void mostra() const;
   Complex suma(const Complex& c) const;
   Complex resta(const Complex& c) const;
   Complex multiplica(const Complex& c) const;
};
```

```
int main()
{
   Complex a, b;
   char operacio;

   a.llegeix();
   b.llegeix();
   cin >> operacio;
   calcula(a, b, operació);

   return 0;
}
```

```
void calcula(const Complex& c1, const Complex& c2, char operacio)
{
   Complex c3;
   if (operació == '+')
     c3 = c1.suma(c2);
   else
     if (operació == '-')
       c3 = c1.resta(c2);
     else
       c3 = c1.multiplica(c2);
   c3.mostra();
}
```

Exercici: solució

```
Complex Complex::suma(const Complex& c) const
{
   Complex resultat;

   resultat.m_real = m_real + c.m_real;
   resultat.m_img = m_img + c.m_img;
   return resultat;
}

Complex Complex::resta(const Complex& c) const
{
   Complex resultat;

   resultat.m_real = m_real - c.m_real;
   resultat.m_img = m_img - c.m_img;
   return resultat;
}

Complex Complex::multiplica(const Complex& c) const
{
   Complex resultat;

   resultat.m_real = (m_real*c.m_real) - (m_img*c.m_img);
   resultat.m_img = (m_real*c.m_img) + (m_img*c.m_real);
   return resultat;
}
```