PROGRAMMIAMO

C++ - Polimorfismo

Polimorfismo

Con **polimorfismo** in generale si intende la capacità di qualcosa di assumere forme diverse. Nella programmazione ad oggetti per polimorfismo si intende la possibilità di ridefinire (con lo stesso nome) i metodi ereditati da una classe base.

Il polimorfismo è dunque strettamente legato al concetto di ereditarietà e può assumere due forme:

- **overloading**: quando il metodo della classe derivata differisce da quello della classe base per il tipo e/o il numero di parametri;
- **overriding**: quando il metodo della classe derivata sovrascrive il metodo della classe base mantenendo inalterato il numero e il tipo di parametri.

Overloading

Abbiamo già incontrato più volte il concetto di **overloading**. Supponiamo per esempio di avere una classe *ellisse* così definita:

```
class ellisse {
        protected: double asseMag, asseMin;
        public:
           void set(double a, double b) {asseMag =a; asseMin = b;}
           double area();
           double perimetro();
        };
```

asseMag e asseMin rappresentano rispettivamente l'asse maggiore e l'asse minore dell'ellisse.

Supponiamo ora di voler derivare dalla precedente una classe cerchio:

```
class cerchio : public ellisse {
                        public:
                        void set(double a) {asseMag =a; asseMin =a;}
                        } ;
```

Si noti che il metodo set è stato riscritto usando un solo parametro (il cerchio è infatti un caso particolare di ellisse con i due assi uguali).

Si tratta di un caso di overloading: il compilatore riconosce quale metodo invocare in base al numero dei parametri passati. Si noti che il metodo set della classe base ellisse continua a essere disponibile anche per la classe derivata cerchio.

Overriding

Nell'overriding invece la classe derivata riscrive identicamente un metodo della classe base, con gli stessi parametri. Questo accadrebbe per esempio, nel caso precedente, se ridefinissimo il metodo set nel seguente modo:

```
class cerchio : public ellisse {
                public:
                void set(double a, double b) {asseMag =a; asseMin =a;}
                } ;
```

Si noti che il parametro b viene semplicemente ignorato in questo caso. A parte l'opportunità o meno di questa scelta di programmazione, è interessante notare che con l'overriding il metodo della classe base non risulta più disponibile nella classe derivata in quanto viene mascherato dall'identico metodo della classe derivata.

precedente - successiva