Classi e overloading degli operatori

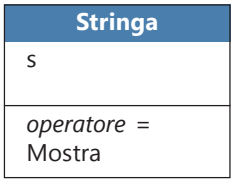
L'**operator overloading** (sovraccarico dell'operatore) indica la possibilità di definire nuove funzionalità ai vari operatori del linguaggio (++, – –, =, ecc.), in modo che il compilatore sia in grado di comprendere il loro impiego nei nuovi tipi di dati creati dal programmatore.
L'*operator overloading* non è altro che un *function overloading* applicato a funzioni particolari chiamate **operator** seguite dall'operatore a cui si riferiscono. Di seguito vengono presentati due esempi di *overloading* degli operatori associati ad una classe.
Il tipo *string* consente di dichiarare variabili non numeriche. Per poter utilizzare questo tipo occorre inserire nel programma la direttiva *#include <string>*, che rende disponibile la libreria del linguaggio C++.
Nel linguaggio C e nelle versioni meno recenti di C++ questo tipo di dato non era disponibile e le stringhe venivano memorizzate in *array* di caratteri. Ancora oggi, per compatibilità con programmi e funzioni del passato, si ha a disposizione la libreria **<cstring>** con tutte le funzioni per la manipolazione di *array* di caratteri, quali **strcpy** per la copia di *array* di caratteri o **strcmp** per il loro confronto, presenti nella libreria *<string.h>* del linguaggio C.

Supponiamo quindi di non disporre del tipo *string* e di voler implementare la classe *Stringa* per la gestione degli *array* di caratteri. Per le diverse manipolazioni degli *array* si impiegano le funzioni di libreria, mentre con l'*operator overloading* si possono definire gli operatori per le usuali operazioni di assegnazione, confronto, ecc.

ESEMPIO 1

Definire l'operatore = per l'assegnazione di array di caratteri.

Il progetto introduce una nuova classe *Stringa* e sovraccarica l'*operatore* = di assegnazione in modo che si possano assegnare array di caratteri anche ad oggetti di questo tipo. Il diagramma della classe *Stringa* è il seguente:



L'attributo *s* rappresenta un array di caratteri.
Il primo metodo rappresenta *l'operator overloading* per l'operazione di assegnazione di array di caratteri, il secondo consente di visualizzare un oggetto dichiarato come istanza della classe *Stringa*.

La codifica della classe *Stringa* e dei suoi metodi è riportata di seguito.

```
class Stringa {
  char s[30];
public:
  // metodi
   Stringa operator= (const char *sorgente);
  void Mostra();
};
```

```
Stringa Stringa::operator= (const char *sorgente)
{
   strcpy(s, sorgente);

   return *this;
}

void Stringa::Mostra()
{
   cout << s << endl;
}
```

Per poter utilizzare la funzione predefinita *strcpy* per la copia tra stringhe, occorre inserire nel programma la direttiva:

```
#include <cstring>
```

Per verificare le funzionalità della classe si può scrivere il seguente programma:

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

// funzione principale
int main()
{
   Stringa s1, s2; // istanze della classe

   s1 = "Milano";
   s2 = "Roma";
   s1.Mostra();
   s2.Mostra();

   return 0;
}
```

La funzione *operator* = ha un valore di ritorno per consentire anche assegnazioni ripetute come:

```
s1 = s2 = "Napoli";
```

Il valore di ritorno è un oggetto di classe *Stringa*. La funzione *operator* = è una funzione membro della classe *Stringa*; per indicare che il valore di ritorno è la classe stessa si usa l'istruzione

```
return *this;
```

La parola riservata **this** del linguaggio C++ indica l'indirizzo della classe di appartenenza (quindi *this* è un puntatore implicito all'oggetto della classe). La parola *this* può essere tradotta in questo contesto come *oggetto corrente* per la funzione membro che si sta definendo.

Si deve osservare che l'*overloading* dell'operatore = proposto in questo esempio consente di assegnare valori costanti (racchiusi tra doppi apici) ad un oggetto di tipo *Stringa*. L'assegnazione tra due oggetti dello stesso tipo *Stringa*, come nella seguente istruzione:

```
s2 = s1;
```

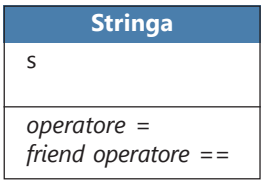
è invece garantita dall'**operatore di assegnazione di default** per la classe *Stringa*.

Un secondo esempio risolve il problema di rappresentare il confronto tra array di caratteri usando l'operatore ==, normalmente non applicabile agli array.

ESEMPIO 2

Definire l'operatore == per il confronto tra due array di caratteri.

Alla classe *Stringa* presentata nell'esempio precedente, viene aggiunto un ulteriore *overloading* applicato all'operatore == per il confronto tra oggetti dichiarati come istanze della classe *Stringa:*



In questo caso, l'*overloading* sull'operatore deve essere necessariamente di tipo **friend**.

La codifica modificata della classe *Stringa* e dell'*overloading* dell'operatore == è riportata di seguito: l'operatore di confronto ha come parametri le due stringhe da confrontare e restituisce un valore booleano (*true/false*) come esito del confronto.

```
   class Stringa {
     char s[30];
   public:
     // metodi
     Stringa operator= (const char* sorgente);
     friend bool operator== (Stringa& a, Stringa& b);
     void Mostra();
   };

Stringa Stringa::operator= (const char* sorgente)
{
   strcpy(s, sorgente);

   return *this;
}

void Stringa::Mostra()
{
   cout << s << endl;
}
```

```
bool operator== (Stringa &a, Stringa &b)
{
   bool c;

   if (strcmp(a.s, b.s) == 0) c = true;
   else c = false;

   return c;
}
```

Nell'*operator overloading* viene utilizzata la funzione *strcmp*, disponibile nella libreria <*cstring*>. Si provi a verificare la funzionalità dell'operatore == eseguendo il programma con le seguenti istruzioni nel *main*:

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

// funzione principale
int main(void) {
   Stringa s1, s2; // istanze della classe

   s1 = "Milano";
   s2 = "Parigi";
   if (s1 == s2) cout << "Stringhe uguali" << endl;
   else cout << "Stringhe diverse" << endl;

   return 0;
}
```