Overloading

Alberto Ferrari

Overloading (sovraccarico)

- Una famiglia di funzioni aventi lo stesso nome, ma un diverso set di argomenti (signature) è detta in rapporto di Overloading, o sovraccaricata.
- Si può parlare di overloading di funzioni, di costruttori e di operatori.
- Sovraccaricare il costruttore di una classe è una pratica comune per gli sviluppatori di librerie, in quanto permette di fornire allo sviluppatore finale diverse modalità per inizializzare gli oggetti.
- In C++ è ammesso l'overloading degli operatori

Overloading di Operatori

- Gli operatori +,-,==,… sono funzioni usate con una sintassi particolare
- C++ consente di sovraccaricare gli operatori così che accettino argomenti di tipo classe
  - Funzione tra le più apprezzate del linguaggio
  - Rende il programma molto più chiaro rispetto a chiamate a funzione equivalenti
  - Non abusarne quando invece l'uso non è chiaro
- Almeno uno degli operandi deve essere di tipo classe

Overloading di Operatori (2)

- Si effettua scrivendo una funzione con sintassi comune, ma con nome dato dalla parola chiave *operator* seguita dal simbolo dell'operatore di cui fare l'overloading
- Unici operatori già utilizzabili su membri di tipo classe:
  - = (assignment): **pericoloso per classi contenenti puntatori**
  - &
- L'overload deve essere esplicito: fare l'overload di = non condiziona +=, -= o !=

Overloading di Operatori come Funzioni non Membro

- E' possibile effettuare l'overloading semplicemente dichiarando le funzioni, senza necessità che siano membro della classe
- Svantaggio: non potendo accedere ai campi privati, hanno necessità di utilizzare accessor
  - La classe è obbligata a fornire tali accessor
  - Overhead nella chiamata degli accessor
- Per ovviare agli svantaggi, l'overloading con funzioni non membro generalmente dichiara le funzioni come friend

Overloading di Operatori come Funzioni Membro

- L'oggetto chiamante serve come primo parametro quindi
  - Gli operatori binari hanno un solo parametro
  - Gli operatori unari non hanno parametri
- Vantaggi
  - È più nello spirito OOP
  - È più efficiente
- C'è uno svantaggio
  - L'oggetto più a sinistra **deve** essere membro della classe
  - Non sempre conviene o non sempre è possibile (es.: operatori >> e <<)

esOverloading.cpp

Funzioni friend

- Una funzione friend di una classe ha accesso ai membri privati della classe pur non essendone membro
- **Deve essere dichiarata friend nella definizione della classe** (per chiarezza meglio se all'inizio)
- Viene definita e chiamata come una funzione ordinaria
- L'uso di funzioni friend migliora le prestazioni (non necessitano di accessor)

Alberto Ferrari

Funzioni friend

- Le funzioni friend più comuni sono gli operatori sovraccaricati
- Una funzione può essere friend di più classi
- Offrono il vantaggio della conversione automatica di tutti gli operandi ma per alcuni autori non sono nello spirito OOP

esFriend.cpp

Alberto Ferrari

Overloading degli Operatori << e >>

- Gli operatori << e >> possono essere sovraccaricati per essere usati per l'I/O degli oggetti di una classe
- Notazione di immediata comprensione
- Non possono essere sovraccaricati come membri: l'operatore più a sinistra non è infatti del tipo della classe per cui si vogliono sovraccaricare << e >>, che invece richiedono rispettivamente **ostream&** e **istream&**

Overloading degli Operatori << e >> (2)

- Di fatto le chiamate effettuate sono nella forma:
  cin >> object; → operator>>(cin, object);
- Le funzioni sovraccaricate degli operatori << e >> ritornano rispettivamente un oggetto ostream e istream, in modo da poter concatenare più chiamate all'operatore stesso
  cout << object1 << object2;

esOverloadingInsertionAndExtractionOperators.cpp

Overloading dell'Operatore =

- L'operatore = (assegnamento) deve essere sovraccaricato come membro
- Se non viene sovraccaricato, viene creato automaticamente
- L'operatore = di default copia i valori delle variabili membro di un oggetto nelle corrispondenti variabili membro di un altro oggetto
- Problemi con i puntatori: nel caso ci siano puntatori, l'operatore = **deve** essere sovraccaricato!

Alberto Ferrari

Il Puntatore this

- Un oggetto ha accesso al proprio indirizzo mediante il puntatore *this*
- *this* non è parte dell'oggetto, piuttosto è passato come argomento implicito a tutte le funzioni membro dell'oggetto

```
void setNum(int num)
{
   this->num = num;
}
```

Alberto Ferrari

L’Operatore = e l’Auto-Assegnamento

- Nell’utilizzo dell’operatore = sovraccaricato è bene testare se si sta tentando un’operazione di auto-assegnamento per evitare *memory leak*

```
// Overloaded = operator; avoids self assignment
const String &String::operator=( const String
   &right )
{
   cout << "operator= called\n";

   if ( &right != this )
   { // avoid self assignment
      delete [] sPtr; // prevents memory leak
      length = right.length; // new String length
      setString( right.sPtr );
   }
   else
      cout << "Attempted assignment of a String
   to itself\n";

   return *this;   // enables cascaded
   assignments
}
```

Alberto Ferrari

Memory Leak

- Un memory leak ("falla nella memoria") è un particolare tipo di consumo non voluto di memoria dovuto al mancato rilascio della memoria non più utilizzata da parte dei processi. Il termine non è etimologicamente corretto, visto che la memoria non viene persa fisicamente, piuttosto diventa inutilizzabile per un difetto del software.

Alberto Ferrari

Overloading degli Operatori ++ e --

- Occorre un modo per distinguere la forma prefissa dalla forma postfissa
- La forma prefissa è sovraccaricata come qualsiasi altro operatore unario
- Nella forma postfissa si aggiunge un parametro fittizio di tipo **int** (solo nella dichiarazione e definizione). Si tratta di un “dummy value” senza nessun altro scopo

Alberto Ferrari

Overloading dell’Operatore []

- L’operatore [] deve essere sovraccaricato come membro
- Se si vuole usare il valore ritornato come l-value deve ritornare una reference
- Il parametro indice può essere di qualsiasi tipo ed è l’argomento della chiamata alla funzione membro

Alberto Ferrari

Overloading: Riassunto

- Almeno un operando deve essere di tipo classe
- Non si può:
  - Creare un nuovo operatore
  - Cambiare il numero di operandi di un operatore
  - Cambiare la precedenza di un operatore
  - Dare un argomento di default a un operatore sovraccaricato
- I seguenti operatori possono essere sovraccaricati solo come membri: =, [], ->, ()

Alberto Ferrari

Overloading: Riassunto (2)

- I seguenti operatori possono essere sovraccaricati solo come non membri o friend: <<, >>
- I seguenti operatori non possono essere sovraccaricati: ., ::, sizeof, ?:
- Per alcuni operatori l'overloading è sconsigliato:
  - && e || quando sovraccaricati effettuano la valutazione completa dell'espressione
  - , quando sovraccaricato non garantisce l'ordine di valutazione da sinistra a destra

Alberto Ferrari