Programmazione Avanzata

uso range-based for loop

Pierluigi Roberti

pierluigi.roberti@unitn.it

Uso di range-based for loop

Lo standard C++11 ha introdotto un nuovo costrutto per il controllo di flusso che prende il nome di range-based for loop (ciclo for su sequenze).

Da un punto di vista semantico, questo nuovo costrutto è del tutto equivalente al tradizionale ciclo for.

Tuttavia, soprattutto nel caso in cui il ciclo sia basato sull'uso di iteratori, la nuova sintassi risulta più leggibile e quasi auto esplicativa.

```
//dichiarazione ed inizializzazione vector
vector<int> v = {0, 1, 2, 3, 4, 5};
vector<int>::iterator it;
```

Uso di range-based for loop

Utilizzo del ciclo for tradizionale:

Un ciclo for tradizionale richiede l'inizializzazione della variabile iterativa it, la definizione esplicita della condizione di uscita e l'incremento della variabile iteratore.
Inoltre, la variabile iterativa, sia essa un puntatore o un iteratore, come in questo caso, deve essere dereferenziata per consentire l'accesso all'elemento desiderato nel corpo del ciclo.

```
cout << "for su <vector>:" << endl;
for (it = v.begin(); it != v.end(); ++it) {
        cout << *it << "\n";
}
```

Uso di range-based for loop

Utilizzo del range-based for:

La sintassi del ciclo basato su intervalli è invece nettamente più snella, si compone di tre elementi principali:

[1] **Range-declaration**: la dichiarazione di una variabile avente lo stesso tipo degli elementi dell'intervallo, in questo caso int. La scelta del nome è del tutto arbitraria, soggetta alle stesse regole che si applicano per la definizione di identificatori in C++.

[2] **Range-expression:** una espressione, anche un valore temporaneo, il cui valore sia una sequenza di elementi. Lo standard identifica come tali tutti gli oggetti per cui sono definiti dei metodi membro begin() e end(), ad esempio le classi container, oppure cui siano applicabili le funzioni std::begin() ed std::end(). Ciò rende possibile usare questo costrutto anche su array semplici.

[3] Il **corpo del ciclo**, in cui possono apparire le istruzioni break e continue.

Uso di range-based for loop

```
vector<int> v = {0, 1, 2, 3, 4, 5};

cout << "range-based for su <vector>:" << endl;
for (int n : v) {
        cout << n << "\n";
}

cout << "range-based for con auto su <vector>:" << endl;
for (auto n : v) {
        cout << n << "\n";
}

int dato[10];
cout << "range-based for con auto array int:" << endl;
for (auto valore : dato) {
        cout << valore << "\n";
}
```

Uso di range-based for loop

L'uso dei metodi std::begin() ed std::end() per l'implementazione del costrutto range-based for loop può implicare che gli **elementi della sequenza siano copiati**, qualora per la classe dell'elemento sia propriamente definita la semantica della copia.
Per cicli che accedono gli elementi in sola lettura, ciò può tradursi in un decremento delle performance, soprattutto nel caso di classi complesse!

```
vector<int> v = {0, 1, 2, 3, 4, 5};
cout << "for su <vector> const iterator:" << endl;
vector<int>::const_iterator cit;

for (cit = v.cbegin(); cit != v.cend(); ++cit)          {
       cout << *cit << "\n";
}

//valido solo dal c++17
for (auto& n : as_const(v)) {
       cout << n << "\n";
}
```