C++0x - Aralık Tabanlı for Döngüleri (Range Base for Loops)

Kaan Aslan
17 Kasım 2009

C++0x'te dile eklenmesine karar verilmiş özelliklerden biri de aralık tabanlı *for* döngülerdir. Aralık tabanlı *for* döngülerine *C#* ve *VB.NET* gibi dillerde *foreach* döngüleri de denilmektedir. Aralık tabanlı *for* döngüleri için önerilmiş olan genel sentaks biçimi aşağıdaki gibidir:

```
for (<aralık değişkeni bildirimi> : <dizilim ifadesi> )
    <deyim>
```

Gördüğünüz gibi döngü yine *for* anahtar sözcüğü ile kuruluyor. Sentaks biçiminin *Java*'dakine benzerliği dikkatinizi çekmiştir. En güncel taslaklara göre (*N2960=09-0150*) bu deyimin eşdeğeri aşağıdaki gibidir:[1]

```
{
    auto &&__range = (<dizilim ifadesi>);
    for (auto __begin = <ilk-iter>, __end = <son-iter>;
                    __begin != __end; ++__begin ) {
        <aralık değişken bildirimi> = *__begin;
        <deyim>
     }
}
```

Burada <*dizilim ifadesi*> döngüdeki *':'* atomunun sağındaki ifadeyi, <*ilk-iter*> aralığın başına ilişkin iteratör değerini, <*son-iter*> ise aralığın sonuna ilişkin iteratör değerini belirtiyor. Dizilim belirten ifade dizi olabilir ya da *begin* ve *end* üye fonskiyonlarına sahip bir nesne tutan sınıf (*container class*) olabilir. Eğer dizilim ifadesi bir dizi ise eşdeğer açılımdaki *__begin* bu dizinin başlangıç adresini *__end* ise son elemanından sonraki adresi belirtir.[2] Örneğin:

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
//...
for (int x : a)
    std::cout << x << std::endl;
```

Bu örnekte dizinin elemanları ekrana yazdırılmaktadır. Aşağıdaki örneği inceleyiniz:

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
//...
for (int &r : a)
    r *= r;
```

Bu örnekte ise dizinin her elemanı onun karesi ile yer değiştiriliyor. Aralık tabanlı *for* döngülerini nesne tutan sınıflar için de kullanabiliriz. Örneğin:

```
std::vector<int> a;
//...
for (int i = 0; i < 10; ++i)
    a.push_back(i);

for (int x : a)
    std::cout << x << std::endl;
```

Bu kod aşağıdaki ile işlevsel olarak eşdeğerdir:[3]

```
std::vector<int> a;
//...
for (int i = 0; i < 10; ++i)
    a.push_back(i);

for (auto iter = a.begin(); iter != a.end(); ++iter) {
    int x = *iter;
    std::cout << x << std::endl;
}
```

Aralık tabanlı döngüler dizilerle ve nesne tutan sınıflarla daha rahat çalışmamızı sağlıyor. Bu özelliğin *C++*'a eklenmesinin yerinde bir karar olduğunu düşünüyoruz.

---

[1] Standardizasyon komitesinin concept kavramını çıkarmaya karar vermesi aralık tabanlı döngülerin taslaklardaki anlatımını da büyük ölçüde değiştirmiştir. Buradaki anlatım makalenin yazıldığı tarihte çıkmış olan en son taslaklara dayandırılmaktadır.

[2] Taslak anlatımlarında aralığın başına ve sonuna ilişkin iteratör değerlerinin global *begin* ve *end* fonksiyonlarıyla elde edileceği belirtilmiştir. Burada anlatım karmaşıklığına yol açmamak için bunu belirtmiyoruz. Ayrıntılar için kaynaklar bölümündeki taslaklara başvurabilirsiniz.

[3] Tabi işlevsel olarak eşdeğerlik söz konusudur. Yoksa yukarıda da belirtildiği gibi gerçek eşdeğer kod şöyle açılabilir:

```
{
    auto &&__range = a;
    for (auto __begin = begin(__range), __end = end(__range);
                        __begin != __end; ++__begin ) {
        int x = *__begin;
        std::cout << x << std::endl;
     }
}
```

Burada *begin* ve *end, <iterator>* kütüphanesine eklenmiş olan global *template* fonksiyonlardır.

Kaynaklar

Douglas, G., Dawes, B. (2009). *Generalized Range-Based For Loop Wording (Without Concepts). (N2930=09-0120)* *http://www.openstd.org/JTC1/SC22/WG21/docs/papers/2009* adresinden alınmıştır.

Working Draft, Standard for Programming Language C++ (N2960=09-0150). (2009). http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2009 adresinden alınmıştır.