Operatör Öncelikleri

19-Mayıs-2009

Aşağıdaki ifadeyi inceleyiniz:

```
b = ++a + foo() * bar();
```

pek çok *C* ve *C++* programcısı işlemlerin aşağıdaki sırada yapılacağının garanti olduğunu sanıyor: ($İ_n$: n'inci işlem):

```
İ1: foo()
İ2: bar()
İ3: ++a
İ4: İ1 * İ2
İ5: İ3 + İ4
İ6: b = İ5
```

Aslında bu durum gerçeği yansıtmıyor. Burada pekala önce ++ işlemi yapılabilir. *foo* fonksiyonunun da *bar* fonksiyonundan önce çağrılması zorunlu değildir. Bunu söylediğimizde bu kez "hani fonksiyon çağırma operatörünün önceliği ++'dan yüksekti?" diye soruyorlar.

Operatörlerin öncelik tablosu aslında bir uydurmadır. Zaten standartlarda böyle bir tablo da yoktur. Operatör öncelikleri bağlam bağımsız gramerden (*context free grammar*) çıkan sonucun yaklaşık ifadesidir. Operatör öncelikleri konusundaki gerçekler öncelik tablosu yoluyla açıklanamaz. O halde şöyle söylemeliyim "operatörlerin öncelik tablosu gerçeğin kendisini anlatmaz ama onun yaklaşık bir benzerini kolayca anlatmamıza yardımcı olur". Biz de *Dernek*'teki *C* eğitimlerinde bu konunun üzerinde fazlaca durmayız. "*Sistem Programlama ve İleri C Uygulamaları"* kursunda sentaks açıklama yöntemlerinin ve *BNF* notasyonunun anlatıldığı bölümde bu gerçeklerden bahsederiz.

Şimdi konumuza dönelim yeniden. Yukarıdaki ifadede neden *foo()* çağırması önce yapılmak zorunda değil? Standartlardaki ilgili bölümün *CFG (context free grammar)*'sini aşağıda veriyorum. (Ancak burada kısaltma amacıyla bazı kısımları bazı yerlerden kestim. Kestiğim yerleri ... sembolünden anlayabilirsiniz). Önce biraz inceleyin:

primary-expression:
  identifier
  constant
  string-literal
  (*expression* **)**

postfix-expression:
  primary-expression
  postfix-expression **[** *expression* **]**
  postfix-expression **(** *argument-expression-listopt* **)**
  ...

argument-expression-list:
    assignment-expression
    argument-expression-list **,** *assignment-expression*

unary-expression:
    postfix-expression
    ++ *unary-expression*
    -- *unary-expression*
    ...
 ...

multiplicative-expression:
    cast-expression
    multiplicative-expression ***** *cast-expression*
    multiplicative-expression **/** *cast-expression*
    multiplicative-expression **%** *cast-expression*

additive-expression:
    multiplicative-expression
    additive-expression **+** *multiplicative-expression*
    additive-expression **-** *multiplicative-expression*
 ...

assignment-expression:
    conditional-expression
    unary-expression assignment-operator assignment-expression
    assignment-operator: one of
    = *= /= %= += -= <<= >>= &= ^= |=

expression:
    assignment-expression
    expression **,** *assignment-expression*

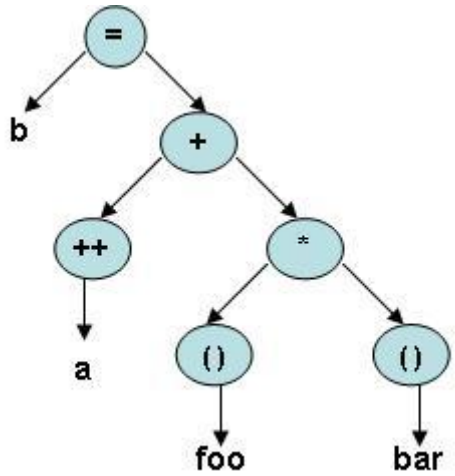
C ve *C++* standartlarında iki operandlı operatörlerin sol tarafındaki ifadelerin mi yoksa sağ tarafındaki ifadelerin mi önce yapılacağı belirsiz *(unspecified)* bırakılmıştır. Yani derleyicileri yazanlar bunlardan herhangi birini seçebilirler. Yaptıkları seçimi de dökümante etmek zorunda değillerdir. Bu durumda standartlardaki gramerden yukarıdaki ifade için şu sonuçları çıkartabiliriz:

1. Atama operatörünün sağ tarafı hesaplanıp *b*'ye atanacak.

2. *foo() * bar()* işleminin sonucu *++a* ile toplanacak, fakat bunların yapılış sırası değişebilir. Yani önce *++a* yapılabilir ya da önce *foo() * bar()* yapılabilir.

3. *foo()* çağırmasının sonucu *bar()* çağırmasının sonucuyla çarpılacak. Fakat işlemlerin yapılma sırası değişebilir.

Daha iyi anlamak için ifade ağacı oluşturmanızı salık veririm:



İki operandlı operatörlerin herhangi bir sırada yapılacağını göz önünde bulundurarak ağacı inceleyin. + operatörünün solu ya da sağı önce yapılabilir. * operatörünün de solu ya da sağı önce yapılabilir. Fakat kesinlikle * operatörünün sonucu ++ operatörünün sonucuyla toplanacaktır, *foo* çağırmasının sonucu *bar* çağırmasının sonucuyla çarpılacaktır.

C ve *C++*'ta *&&* , ||, *? :*, = ve *','* operatörlerinin özel durumları olduğunu biliyorsunuz. Örneğin *&&* ve || operatörlerinin her zaman sol tarafı önce yapılmaktadır.

İki operandlı operatörlerdeki operand işleme belirsizliğinin her programlama dilinde var olduğunu sanmayın. Örneğin *C#* ve *Java*'da kesinlikle sol taraftaki operandın önce yapılması öngörülmektedir. Dolayısıyla yukarıdaki ifade bu dillerde gerçekten aşağıdaki sırada yapılmak zorundadır:

```
b = ++a + foo() * bar();

İ1: ++a
İ2: foo()
İ3: bar()
İ4: İ2 * İ3
İ5: İ1 + İ4
İ6: b = İ5
```

Umarım karışık gelmemiştir size :-).