C++ ile Programlama Standardı

Genel Prensipler

- C dilinde varolan bir özellik C++ dilinde başka bir özellik tarafından iyileştirilmişse, C++'daki özellikler kullanılır. Buna örnek:
  - ♦ **structure** yerine **class**
  - ♦ **malloc(), calloc(), free()** fonksiyonları yerine **new() ve delete()** operatörleri
  - ♦ **printf(), scanf()** yerine **>>, <<** operatörleri
  - ♦ dosyalar arasında bilgi saklamaya yarayan **static** anahtar kelimesi yerine aşağıda **modüller** kısmında anlatılan mekanizma
- Veri Yapıları (Data Structures) nesneye yönelik kavramlar dikkate alınarak tasarlanır.
- Kullanıcı tarafından tanımlanmş bir class 'a ait nesnelerin input/output yapılması gerekiyorsa, << ve >> operatörleri kullanıcı tarafından tanımlanan sınıflar için **overload** edilir.
- Overload yapılan tüm operatörler anlam açısından ilk anlamlarından önemli farklılık göstermez.
- Programda tüm literal değerler constantlara atanarak kullanılır. Program içinde literal yer almaz. (literallere örnek: "abc", 3.14, 44)
- **goto** komutu kullanılmaz

Modüller

Büyük bir programın modüllere ayrılması, her modülün bir interface, bir de implementasyon kısmının olması, programın sınıflar şeklinde organize edilmesi sayesinde olur. Sınıf tanımları, modülün adını taşıyan header (.h uzantılı) dosyada bulunur. **Instance** ve **sınıf** metodları ile **sınıf değişkenleri** (class variables) modülün adını taşıyan **.cpp** uzantılı dosyada tanımlanır. Header dosyası implementasyon dosyasına dahil edilir. Tanımlanan modülün sunduğu sınıfları kullanmak isteyen tüm diğer implementasyon dosyaları da modülün header dosyasını dahil ederler.

Programcının kullanmak istediği tüm küresel (global) değişkenler, sınıf değişkeni olarak tanımlanır (**statik** anahtar kelimesi kullanılarak). Global değişkenler gibi, modül içinde kullanılmak istenen veya dışa ihraç edilmek istenen fonksiyonlar da sınıf metodları olarak tanımlanır. Bunun için modülün adı ile ayni olan bir sınıf kullanılır (örnek olarak, bundan böyle, tanımlamak istediğimiz

modülün adının **test** olduğunu varsayalım). Öyle ise, test isimli bir sınıfımız olacaktır. Bu sınıfın **public** kısmında başka modüllerden görülmesine izin verilen sınıf değişkenleri, sınıf metodları, yeni sınıf tanımları ve **enum** tanımları bulunur. Bu sınıfın **private** kısmında ise sadece modül içinden görülebilen sadece bu sınıf içinde görülen sınıf değişkenleri, sınıf metodları, yeni sınıf tanımları, **enum** tanımları bulunur. Bu sınıfın tanımı, sınıf ile ayni ismi taşıyan, **test.h** isimli bir dosyada bulunur. **test.cpp** isimli dosyada ise test sınıfı içinde beyan edilmiş olan sınıf metodları, sınıf değişkenleri ile iç sınıf (nested class) lara ait metodların ve sınıf değişkenlerinin tanımı bulunur. Bu tanımlar modülün kullanıcıları olan diğer modüllerden saklıdır. Böylece **bilgi saklama** pensibine de bağlı kalınmış olur.

Bir programı oluşturan bütün modüllerin yapısı bu şekildedir. Bu modüller içinde en azından ana modül görevini yapacak olan **main modülü** bulunmak zorundadır. **main()** fonksiyonunu **main.cpp** dosyasında tanımlanır.

Değişkenlerle ilgili Kurallar

- Yerel değişkenler olabildiğince kullanılacakları yere yakın olarak tanımlanırlar.
- Değişkenlere ilk tanımlandikları yerde, eğer anlamlı ise, değerler atanır.
- Değeri değişmeyecek olan nesneler **const** anahtar sözcüğü ile nitelendirilir. Bu kural, değişmeyecek olan parametreler için de geçelidir.

Macrolar ile Ilgili Kurallar

Makrolar olabildiğince az kullanılır. Eger ille de makro yazılması gerekiyorsa, makro'nun parametreleri makro vücudu içinde her kullanıldıkları yerde parantez içine alınırlar. Makro vücudunun kendisi de paranatez içine alınır.

Parametreler ile ilgili Kurallar

Büyük nesnelerin parametre olarak geçilmesi gerekiyorsa, ve bu parametreler fonksiyon içinde değişmeyeceklerse, **constant reference** olarak fonksiyona geçilir.

Program karmaşıklığı ile ilgili kurallar

- Bır expression'da 10'dan fazla operatör bulunmaz.
- Iç içe geçmis döngü (nested loop) sayısı 6'yı aşmaz.

- Metodların vücudundaki kodun satır sayısı 60'ı geçmez.
- Iç içe geçmiş sınıflarda (nested classes) iç içe geçme derinliği ikiyi aşmaz.
- Iç içe geçmiş **if** komutlarında iç içe geçme derinliği dördü aşmaz.

Görünüm ve Okunabilirlikle ilgili Kurallar

- Metod tanımları, sınıf tanımları aşağıdaki yorum ile ayrılır.

/**/

- **for** döngüsü şeklen aşağıdaki gibidir.

```
for(...) {
   statement1;
   statement2;
   ......
}
```

- **if** komutu şeklen aşağıdaki gibidir.

```
if expr then
     stmts
else
     stmts
```

- **Aç parantez, kapa parantez** gerektiren tüm yapılar, **for** döngüsündeki şekilde yazılır.
- Bir satırda en çok bir tanım yapılır, en çok bir statement yazılır.

Indentatıon (içe alma) 5lik bloklar halinde yapılır. Örneğin, **for** döngüsünün vücudunda bulunan statementler 5 boş harf yerı kadar **for** anahtar kelimesinden içeride yazılır.

- Her satır 80 colon'u aşmayacak şekilde gerekirse birden çok satıra bölünür.
- Bir metodun vücudu 1 satırdan fazla ise, metod sınıf tanımının dışında tanımlanır.

Açıklamalar (Comments)

Her Dosyada Bulunması Gereken Bilgiler

Her header (.h) ve implementatıon (.cpp) dosyasında aşağıdaki bilgiler yorum olarak yer alır:

- Dosyanın adı ve directory'si
- Modülün hangi projenin parçası olduğu
- Dosyanın işlevi
- Dosyanın en son ne zaman ve kim tarafından modifiye edildiği
- Modülün görevini yerine getırebilmesi için daha nelerin yapılması gerektiği f
- Hangi dosyaları açıldığı, kapadığı, yaratıldığı (sadece implementatıon dosyasında)

Bir header dosyası **#include** komutu ile bir dosyaya dahil edilmişse, bu dosyanın hangi fonksiyonlar, tanımlar, makrolar, değişkenler için dahil edildiği açıklanır.

Metodlarla ilgili Bilgiler

Her metodun işlevi, parametrelerinin ne amaca hizmet ettiği, hangi metodlar tarafından çağrıldığı yorum olarak fonksiyondan önce yazılır.

Sınıflarla ilgili Bilgiler

Her sınıf için tanımlanan sınıf değişkenleri, metodları ile **instance** değişkenleri ve metodlarının işlevleri açık bir şekilde anlatılır.