YAPILAN İŞ: C++ komutları ve yazımına aşina olmak ve nesneye yönelik programlama konusuna yatkınlığı artırmak için doküman okuma ve örnek projeler ile çalışma.

Not: C++ ile diğer konular üzerinde çalışmaya başlamadan önce genel yazım kuralları, algoritma mantığı, temel komutlar, veri tipleri ve operatörler, diziler, if-else statementlar ve döngüler hakkında bilgi sahibi olunmalı ve pratik yapılmalıdır.

Not: Bahsedilen konular ile ilgili örnek kodlara https://github.com/BerkeCanGoktas/FonksiyonVeOBB adresindeki konu ile ilgili klasörlere erişerek ulaşılabilir.

Konu: Fonksiyonlar

Fonksiyonlar bir görevi (bir kod parçası) yerine getirmek için tanımlanan özel işlemlerdir. Özellikle program içerisinde birden fazla kez tekrar eden kod bloklarını daha pratik ve okunabilir bir şekilde işlemek için kullanılırlar. Fonksiyonların genel yapısı,

-Döndürülecek veri tipi- -fonksiyon adı-(-fonksiyona verilecek parametreler-){

    -fonksiyon çağrıldığında çalışacak komut satırları-

    }

Şeklindedir.

Döndürülecek veri tipi: fonksiyonlar “return” komutu ile yaptıkları işlemler sonrasında bir değer döndürebilirler. Fonksiyonun veri tipi bu döndürülen değerle uyumlu olmalıdır. Fonksiyon bir değer döndürmüyorsa tipi “void”dir.

Not: return komutundan sonraki fonksiyon satırları çalıştırılmaz.

Fonksiyon adı: fonksiyonu daha sonra çağırmak için kullanılacak isimdir.

Fonksiyona verilecek parametreler: fonksiyonlar çağrılırken çeşitli veri türlerinden değişkenler veya sabitler fonksiyona verilebilir ve daha sonra fonksiyonun bu verileri işlemesi sağlanabilir. Bu fonksiyonlara esneklik ve kullanım genişliği sağlar. Fonksiyona verilecek parametreler fonksiyon adından sonra parantez () içinde ve birden fazla ise aralara virgül konarak yazılır.

Not: fonksiyon çağrılırken, fonksiyonun gerektirdiği sayıda parametre vermemek hataya sebep olur.

Fonksiyon çağrıldığında çalışacak kod satırları: fonksiyonlar, fonksiyon_ismi(parametre1, parametre 2) şeklinde isim ve gerekli parametreler yazılarak çağrılır. Bu çağırma sonucunda çalıştırılan kodlar fonksiyon tanımlanırken süslü parantezler arasına yazılmış olan kodlardır.

Not: fonksiyonlar çağrılmadan önce tanımlanmak zorundadırlar. Yani bir fonksiyon çağrıldığı bir satırın aşağısında tanımlanamaz. Ancak fonksiyonların süslü parantez içerisindeki çalıştıracağı kodları çağırmadan önce belirtme zorunluluğu yoktur. Yani bir fonksiyon,

Döndürülecek veri tipi- -fonksiyon adı-(-fonksiyona verilecek parametreler-)

Şeklinde tanımlanıp daha sonraki satırlarda (programın sonu bile olabilir)

-Döndürülecek veri tipi- -fonksiyon adı-(-fonksiyona verilecek parametreler-){

-fonksiyon çağrıldığında çalışacak komut satırları-

}

Şeklinde tanımlanması tamamlanabilir.

Not: Fonksiyon içerisinde tanımlı değişkenler local değişken olarak isimlendirilir ve değerleri fonksiyona gönderilen parametreler ile birlikte fonksiyon dışına çıkamaz. Bunun için parametrelerin referans ile gönderilmesi gerekir.

Recursive Fonksiyonlar: Recursive fonksiyonlar içlerinde yine kendilerini çağıran fonksiyonlardır. Bu sayede bir fonksiyonun birden fazla kez çalıştırılması gerektirilen problemler daha kolay çözülebilmektedir.

Not: Bir fonksiyon kendi içinde başka bir fonksiyonu da çağırabilir. Hatta bir fonksiyon içerisinde çağrılan fonksiyonun yine çağırıcı fonksiyonu çağırması ile de recursive yapılar elde edilebilir.

Function Overloading: Fonksiyonların farklı veri tipleri döndürebilmesi aynı isimli ve farklı veri tipi döndüren fonksiyonların aynı program içerisinde kullanılmasına ve daha dinamik bir yapı oluşmasına olanak sağlamaktadır. Bir fonksiyonu farklı veri tipleri ile tanımlamaya function overloading denir. Derleme esnasında parametreler sayesinde hangi türde fonksiyonun çağrılacağı derleyici tarafından anlaşılır. Bu sayede aynı görevi yapan fonksiyonları aynı isimle tanımlama ve dolayısıyla okuma, yazma kolaylığı sağlanır.

Friend Functions: Friend functionlar sınıflar içerisine “friend” ön eki ile tanımlanırlar, sınıf dışarısında ise gövdeleriyle birlikte tanımlamaları yapılır. Bu fonksiyonların sınıf içerisindeki private değişkenler de dahil olmak üzere tüm üyelere erişimi vardır.

Lambda Fonksiyonları: Lambda fonksiyonları sınıf objeleri ile oluşturulan [operator() ile] dinamik fonksiyon yapılarını sınıfa gerek kalmadan karşılamak için ortaya atılmıştır. Lambda fonksiyonları,

[](parametreler){

Komut satırları

}

Şeklinde tanımlanırlar.

Köşeli parantezler arasına komut satırları içinde yakalanması gereken değişkenler, referanslar, objeler belirtilir veya fonksiyona bağlı olarak hiçbir şey belirtilmeyebilir.

Template Functions: Bazen kullanacağımız fonksiyonların türleri belirsiz olabilir. Böyle durumlarda template <class Veri_Tipi_Değişkeni> satırını fonksiyonun üstüne ekleyerek fonksiyonun tipini verilen değişkenlere göre dinamik hale getirebiliriz.

Not: <> içerisinde birden fazla veri tipi oluşturabiliriz. Fonksiyonun sağlıklı çalışması için bu değişken veri tiplerinin, parametreleri tanımlamada doğru kullanılması gerekir. Yoksa float bir değişkene int tipi verilip veri kaybı yaşanması gibi durumlar olabilir.

Virtual Functions: Virtual fonksiyonlar soyut sınıflar içerisinde tanımlanır ve süslü parantezler (ve içindeki kod satırları) yerine = 0 ile tanımlanırlar. Ayrıca başlarına virtual etiketi eklenir. Sanal fonksiyonlar sayesinde child sınıfların aynı isimli fonksiyonlarına mother sınıftan bir pointer objesi ile erişilebilir.

Not: Pointer ile fonksiyon çağırma işlemlerinde “.” yerine “->” kullanılır.

Rapor Tarihi: 17.10.2021

Yazan: Berke Can GÖKTAŞ