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: Nesneye Yönelik Programlama

Nesneye yönelik programlama, programların daha okunabilir ve düzenli olmasını sağlayan, yazma kolaylığı ve çalışma hızını iyileştiren bir programlama yaklaşımıdır. Nesneye yönelik programlamanın temelini sınıf ve objeler oluşturur.

Sınıf: Objelerin yaratıldığı template olarak özetlenebilir. Classlar objelerin özellikleri ve fonksiyonlarını belirtirler.

Obje: Sınıftan yaratılan instancelar objelerdir. Objeler oluşturuldukları classtaki özellikleri alır ve fonksiyonları kullanırlar.

Örnek verilirse bu konu daha iyi anlaşılabilir. Araba bir sınıftır. Araba sınıfının renk, marka gibi özellikleri vardır. Ayrıca arabalar hızlanmak, fren yapmak gibi fonksiyonlara sahiptir. Şimdi bu araba sınıfından kırmızı renkli bir Ford marka araba objesi oluşturulduğunu düşünelim. Bu araba sınıfının tüm özelliklerine sahiptir ve aynı zamanda fren yapmak, hızlanmak gibi fonksiyonları da kullanabilir.

Bir class yaratmak için class key wordü kullanılır. Var olan bir classtan obje oluşturmak içinse class_name obj_name şeklinde bir başlatma yapılır. Objeler üstünden değişkenler veya fonksiyonlara erişmek için . operatörü kullanılır.

Bir classta bulunan özellikler farklı access koşullarına sahip olabilir. Bu özellikler public (herkes tarafından erişilebilir), protected (class ve bu classın alt sınıfları tarafından erişilebilir) ve private (doğrudan erişilemez) şeklindedir. Access tanımlaması Access_name: şeklindedir ve bir tanımlama yapılmadığı müddetçe default olarak publictir. Doğrudan erişilmesi gerekmeyen özelliklerin private olması daha düzgün bir yazım biçimi olarak kabul edilir çünkü daha iyi kontrol ve olası hataların önlenmesini sağlar. Peki private özelliklere nasıl erişilebilir?

Private özelliklere public fonksiyonlar sayesinde ulaşılabilir (Encapsulation). Bu fonksiyonlara get ve set denir. Get fonksiyonu private özelliği çağırmaya, set ise değiştirmeye (güncellemeye) yarar. Get ve set dışında bir de constructor ve destructor şeklinde iki tane

daha sıkı kullanılan fonksiyon vardır. Bu fonksiyonların adı class adıyla aynı olur (destructorda başına ~ gelir) ve constructor obje ilk oluşturulduğunda çağrılacak komutları, destructor ise yok edildiğinde çağrılacak komutları belirtir. Constructor sayesinde obje oluşturma sırasında değer verme işlemi yapılabilir. Destructor ise işi bittikten sonra objeyi yok eder (hafıza açısından önemlidir). Compiler default olarak bir destructor oluşturur, yani genellikle ayrıca bir destructor tanımlamasına gerek yoktur.

Inheritence:

Inheritence bir sınıfın başka bir sınıfın özelliklerini alarak tanımlanmasıdır. class class_name : access_type inherited_class_name{attributes} şeklinde tanımlanır. Inherited class üst sınıfının izin verilen tüm üyelerine erişebilir. Aynı isimli fonksiyonlar varsa aksi belirtilmedikçe kendi fonksiyonunu kullanır (Polymorphism). Ayrıca bir sınıf birden fazla sınıftan türetilmiş olabilir.

Abstract Class:

En az bir virtual function içeren sınıf soyut sınıftır. Soyut sınıflardan obje tanımlanamaz. Kalıtılan sınıfları daha iyi yönetmek için kullanılır. Soyut sınıftan kalıtılan bir sınıfın, virtual functionlarla aynı isimde fonksiyonlara sahip olması gerekir. Yoksa kalıtılmış bu sınıf da soyut sınıf haline gelir.

Template Class: template classlar class tanımından yukarıya template<class Type1, class Type2,…> şeklinde istenen sayıda tip tanımlaması yapılarak oluşturulur. Bu classların attributeları değişen tiplerde değerlere sahip olabilir. Ayrıca fonksiyonları da değişen tiplerde değerler döndürebilir.

Not: Templatelar ile dinamik tipler kullanılabildiği gibi, sabit bir tip de tanımlanabilir. Örneğin template<class T, int N> ibaresi T'nin değişken bir tip olduğunu N'nin ise bir int olduğunu gösterir. Daha sonra class içerisinde bu N tekrar tanımlanmadan kullanılabilir (örneğin bir arrayin uzunluğunu kararlaştırmak için kullanılabilir).

Not: Eğer ki bir sınıfın gönderilen bazı veri tiplerinde özel işlemler yapmasını istersek template specialization kullanabiliriz. Örneğin bir sınıf int türündeki objesinin değerini bir artırırken, char türündeki bir objesini bir sonraki karaktere geçirebilir. Specialization için

template<>

class Class_Name <Specialization_Type>{komutlar} şeklinde bir tanımlama yapılır.

Konu: Thread Value Return

Threadlerden value return etmek için çeşitli yollar kullanılabilir. Bunlardan ilki future kütüphanesi ile mümkündür. Bu kütüphane ile gelen promise ve future objeleri bunu mümkün kılar. promise<value_type> objName ile bir obje oluşturulur ve threadteki fonksiyona pointer ile verilir. Bu objeye değer ataması set_value fonksiyonu ile yapılır. Değerin istendiği threadte ise bir future objesi future<value_type> objName şeklinde

tanımlanır. Bu objeye promise objesinden değer almak için get_future fonksiyonu kullanılır. En sonunda ise get fonksiyonu ile future objesinden değer çekilir.

Diğer bir yol ise async kullanmaktır. Auto varName = async(funcName, params) şeklinde değer çekilebilir. Bu kullanımda ayrı bir thread oluşturulmaz, async bu satırda kendisi thread oluşturup değeri çeker.

Üçüncü bir yol ise threadte çalışacak fonksiyonlara parametreleri referans olarak yollayıp değişimin global olmasını sağlamaktır. Ancak hem threadler arası eşzamanlılık olmadığında hem de değişkenlerin değerlerini korumak gerektiğinde vb durumlarda problemlere yol açacağı için tercih edilmemesi daha uygun olur.

Async: future kütüphanesi ile gelen, threadlerin birbirinden bağımsız ve paralel çalışmasını sağlayan bir işlevdir. future<value_type> objName = async(launch::async, funcName, params) şeklinde bir tanımlama ile çağrılır.

Konu: Boost Kütüphanesinin Eklenmesi

Bu harici kütüphane cmake tarafından direk tanındığı için harici işlem gerektirmez. İndirilen dosyalar içerisindeki bootsrap.bat ardından b2.exe çalıştırılır. Daha sonra visual studioda çözüme sağ tıklanır. Properties > C/C++ > General > Additional Include Directories kısmına boostun indirildiği kısım yazılır. Daha sonra yine properties içinde Linker>General>Additional Library Directories kısmına boost içindeki stage klasöründeki lib klasörü gösterilir. Son olarak projeye istenen kütüphaneler boost/kütüphaneadı.hpp şeklinde include edilir.

Berke Can GÖKTAŞ

24.10.2021