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: Bazı C++ Konseptleri

Vector

Vectorler, vector kütüphanesi ile gelen bir nevi dinamik arraylerdir. İçerisine eklenen eleman sayısına göre boyutları değişebilir. Vectorler, vector<type> name şeklinde tanımlanır. Burada type int gibi bir veri tipi olabileceği gibi, class veya user-defined bir veri tipi de olabilir. Vectorler çeşitli kullanışlı fonksiyonlara sahiptir (swap(), begin(), size() gibi) ve bir iterator ile ekrana yazdırılabilir veya elemanlarına tek tek erişilebilir.

Exception Handling

Exception, program çalışırken oluşan bir problemdir. Sıfıra bölmeye çalışmaktan hafıza sorunlarına kadar çok geniş bir problem yelpazesi olduğu gibi tanımlanan özel durumlar da exception handlingle ele alınabilir. Throw komutu ile Exception olması halinde Exception fırlatılır. Exception beklenen kod bloğu try{} ibaresinde süslü parantezler içinde yazılır. Daha sonra catch(){} bloğuyla exception yakalanır ve süslü parantezler içerisindeki kodlar çalıştırılır. Ayrıca catch çeşitli parametreler alıp bunları değerlendirebilir.

Files

C++ ile dosya yönetimi için fstream kütüphanesi eklenmelidir. Bu kütüphanede yazma (ostream), okuma (istream), append etme gibi işlemler mevcuttur. Dosyaların istenilen işlem türünde bir obje yaratıp, bu obje üstünden methodlarla kontrol edilmesi gerekir.

Pointer

Pointerlar veri, değişken, fonksiyon vb.nin hafızadaki adresini tutarlar. Bir pointerı tanımlamak için adresini tutacağı tür * değişken adı yapısı kullanılır. Pointerların işaret ettiği adresten veriyi almak için yine * operatörü (overload edilmiş bu * operatörlü kullanıma dereferencing denir) pointerın adından önce konur.

Not: Pointer objeler classlarının methodlarına erişmek için . operatörü yerine -> operatörünü kullanır.

Reference

Pointerlara benzer şekilde reference da hafızada bir yeri gösterir. Pointerlarla aynı düzende ancak * yerine & kullanılarak tanımlanırlar. Hafızada gösterdikleri yer boş olamaz. Bu yüzden iyi tanımlanmamaları programda run time errorlere sebep olabilir. Dereference etmeye gerek kalmadan tuttukları değeri gösterebilirler.

List

List kütüphanesi ile eklenen listeler aslen array gibidirler. list<type> name şeklinde tanımlanırlar. Kendilerine özel çeşitli metotları olsa da en göze çarpan özellikleri hem baştan hem de sondan eklemeli bir yapı olmalarıdır. Listeler iteratorlar ile yazdırılırlar.

Map

Mapler özünde key ve value şeklinde ikili değerler tutan yapılardır. Map<type, type> name şeklinde tanımlanırlar. Key her değer için özel olmalıdır. Value için böyle bir kısıtlama yoktur. Kendilerine ait metotları mevcuttur. Onun dışında key değerleri map_adı->first, value değerleri ise map_adı->second şeklinde çağrılır ve mapler bir iterator ile yazdırılabilir.

Stack

Yine bir değer dizisi tutmaya yarayan stack yapısı stack<type> name şeklinde tanımlanır. Stacklerin en büyük özelliği last in first out yapısında olmalarıdır. Yani en son eklenen değer ilk ulaşılan olacaktır. Bu tarz bir yapıya ihtiyaç duyulduğunda stack yapısı ve metotları kullanılabilir.

Multithreading

Threadler bağımsız alt işlemlerdir. Kendilerine özgü bellek alanları vardır. Ana programdan bağımsız olarak yürütülebilirler. Thread kütüphanesi ile eklenir. Bir fonksiyon, functors (obje fonksiyonlar), lambda expression çağırmak için syntax thread name(func_name, parameters) şeklindedir. Burada parameters fonksiyona yollanacak olan parametrelerdir. Bir classın üye fonksiyonunu çağırmak içinse thread name(Class_name::Func_name, obj_name, parameters) yazımı kullanılır. Burada class_name fonksiyonun bulunduğu sınıf, obj_name ise fonksiyonu çağırmak için kullanılacak nesnenin adıdır. Bu yüzden thread çağrılmadan önce objenin yaratılması gerekir.

Bazı Thread Fonksiyonları:

joinable(): Threadin join edilip edilemediğini kontrol eder, edilebiliyorsa true döndürür.

join(): Ana kod bloğuna devam etmeden önce threadin işleminin bitmesini bekler.

detach(): Thread objesi ile thread execution işlemini birbirinden ayırır. Obje yıkılsa bile execution işlemi devam eder. Detach edilmiş threadler joinable değildir.

sleep_for(parameter): Girilen değer boyunca (değer zaman kütüphanelerine ait bir obje olmalı) threadi bekletir. Gerçek zamanlı uygulamalarda kullanılabilir.

Not: this_thread::func_name() şeklinde bir yapı ile içinde bulunulan threadle ilgili işlem yapılabilir.

Berke Can GÖKTAŞ

20.10.2021