**# Online C++ Programlama Dili Kursu İçeriği Ağustos 2022 Coşkun ERGAN**

**## C++ Dilinin Genel Tanıtımı**

✅ C++ dilinin tarihçesi

✅ C++ dili ve programlama paradigmaları

✅ C++ dili standartları

✅ C++98 – 03

✅ C++11

✅ C++14

✅ C++17

✅ C++20

✅ eski C++ ve modern C++

**## C Dili ve C++ İçindeki C Dili (C in C++)**

✅ C dilinden C++ diline geçiş

✅ işlev bildirimleri ve tanımlamalarına ilişkin farklılıklar

✅ türlere ve tür dönüşümlerine ilişkin farklılıklar

✅ C’de geçerli C++’da geçersiz durumlar

✅ C99 ve C++

**## Temel Kavramlar (Basic Concepts)**

✅ tamamlanmış ve eksik türler (complete & incomplete types)

✅ tek tanımlama kuralı (one definition rule)

✅ ifadelerin değer kategorileri (value categories)

✅ tanımsız davranış (undefined behavior)

✅ derleyiciye bağlı davranışlar. (implementation defined & implementaion specified)

✅ derleyici eklentileri (compiler extensions)

✅ kapsam (scope) ve isim arama (name lookup)

✅ erişim kontrolü (access control)

✅ çift anlamlılık hatası (ambiguity)

**## İlk Değer Verme (Initialization)**

✅ eş biçimli ilk değer verme (uniform initialization)

✅ daraltıcı dönüşümler (narrowing conversions)

✅ most vexing parse

✅ doğrudan ilk değer verme (direct intialization)

✅ değerle başlatma \_value initialization\_

✅ kopyalama ile ilk değer verme (copy initialization)

✅ varsayılan ilk değer verme (default initialization)

✅ bileşiklere ilk değer verme (aggregate initialization)

**## Tür Çıkarımı (Type Deduction)**

✅ auto belirteci ile tür çıkarımı (auto type deduction)

✅ decltype belirteci ile tür çıkarımı (decltype type deduction)

✅ sonradan gelen geri dönüş türü (trailing return type)

✅ auto geri dönüş değeri türü (auto return type)

✅ decltype auto tür çıkarımı (decltype auto)

**## kapsamlandırılmış enum türleri (scoped enums)**

✅ geleneksel enum türleri (conventional enum types)

✅ baz tür seçimi (underlying type)

✅ tür dönüşümleri (type conversions)

✅ enum türleri kapsam (enum classes & scope)

✅ using enum declarations (C++20)

**## Sabit İfadeleri (Constant Expressions)**

✅ const anahtar sözcüğü ve const semantiği (const keyword & const semantics)

✅ const nesneler (const objects)

✅ constexpr anahtar sözcüğü (constexpr keyword)

✅ constexpr işlevler (constexpr functions)

✅ consteval işlevler - C++20 (consteval functions - C++20)

✅ constinit anahtar sözcüğü - C++20

**## İşlevlerin Varsayılan Argüman Alması (Default Arguments)**

**## Referans Semantiği (Reference Semantics)**

✅ sol taraf referansları (L value references)

✅ sağ taraf referansları (R value references)

✅ referanslar ve const semantiği (references & const semantics)

✅ referanslar ile göstericilerin (pointer) karşılaştırılması

✅ parametresi referans olan işlevler

✅ referans döndüren işlevler

✅ referanslar ve \_life extension\_

**## İşlev Yüklemesi (Function Overloading)**

✅ genel kurallar

✅ yüklenmiş işlev çözümlenmesi (function overload resolution)

✅ const yüklemesi (const overloading)

✅ extern "C" bildirimi (extern C declarations)

✅ işlev yüklemesinde dikkat edilmesi gereken durumlar

**## Tür Dönüştürme Operatörleri (Type-cast Operators)**

✅ static\_cast<> operatörü

✅ const\_cast<> operatörü

✅ reinterpret\_cast<> operatörü

✅ dynamic\_cast<> operatörü (kalıtım başlığı altında)

**## inline işlevler ve inline değişkenler**

✅ inline işlevler (inline functions)

✅ inline değişkenler (C++17) (inline variables)

**## Sınıflara giriş (Introduction to Classes)**

✅ sınıf kapsamı (class scope)

✅ sınıflar ve isim arama (name lookup)

✅ erişim kontrolü (access control) ve veri gizleme (data hiding)

❇️ public öğeler (public members)

❇️ private öğeler (private members)

❇️ protected öğeler (protected members)

✅ sınıfların öğeleri (class members)

❇️ sınıfların veri öğeleri (data members)

⦿ non-static veri öğeleri

⦿ mutable veri öğeleri

⦿ static veri öğeleri

❇️ sınıfların üye işlevleri

⦿ non-static üye işlevler

⦿ const üye işlevler

⦿ static üye işlevler

❇️ this göstericisi ve \\*this

❇️ sınıfların tür öğeleri (type members)

✅ sınıfların kurucu işlevleri (constructors)

❇️ kurucu işlev ilk değer verme listesi (constructor initializer list)

❇️ delege eden kurucu işlevler (delegating constructors)

❇️ explicit kurucu işlevler (explicit constructors)

✅ sınıfların sonlandırıcı işlevleri (destructors)

✅ üye işlevlerin çağrılması

✅ sınıflar ve const doğruluğu (classes & const correctness)

❇️ const sınıf nesneleri (const objects)

❇️ const üye fonksiyonlar (const member fuctions)

✅ geçici sınıf nesneleri (temporary objects)

✅ otomatik tür dönüşümleri (implicit type conversions)

✅ mutable anahtar sözcüğü

✅ friend bildirimi (friend declarations)

❇️ friend bildirimi ve veri gizleme

❇️ global işlevlere friend bildirimi

❇️ sınıfların üye işlevlerine friend bildirimi

❇️ sınıflara friend bildirimi

❇️ attorney client idiyomu

**## Sınıfların Özel Üye İşlevleri ve Kopyalama İşlemleri**

**(Special Member Functions & Copy Control)**

✅ sınıfların özel işlevleri (special member functions)

❇️ default constructor (varsayılan kurucu işlev)

❇️ destructor (sonlandırıcı işlev)

❇️ copy constructor (kopyalayan kurucu işlev)

❇️ move constructor (taşıyan kurucu işlev)

❇️ copy assignment (kopyalayan atama işlevi)

❇️ move assignment (taşıyan atama işlevi)

✅ özel işlevlerin default edilmesi

✅ özel işlevlerin delete edilmesi

✅ sınıflar ve taşıma semantiği (move semantics)

✅ \_rule of zero\_

✅ \_rule of five\_

✅ kopyala takas et idiyomu (copy & swap idiom)

✅ kopyalamanın eliminasyonu (copy elision)

❇️ geçici nesneler yoluyla kopyalama eliminasyonu (copy elision by temporaries)

❇️ RVO (return value optimization)

❇️ NRVO (names return value optimization)

❇️ zorunlu kopyalama eliminasyonu (mandatory copy elision)

**## Operatör Yüklemesi (Operator Overloading)**

✅ operatör yüklemesine ilişkin genel kurallar

✅ üye operatör fonksiyonları (member operator functions)

✅ global operatör fonksiyonları (global operator functions)

✅ aritmetik operatörlerin yüklenmesi

✅ karşılaştırma operatörlerinin yüklenmesi

✅ 3 yollu karşılaştırma operatörü (3-way comparision operator) C++20

✅ "++" ve "--" operatörlerinin yüklenmesi

✅ ok operatörü ve içerik operatörlerinin yüklenmesi (overloading of arrow and dereferencing operators)

✅ [] operatörünün yüklenmesi (overloading of subscript operator)

✅ fonksiyon çağrı operatörünün yüklenmesi (overloading of function call operator)

✅ tür dönüştürme operatör fonksiyonları (type-cast operator functions)

✅ programcının tanımladığı sabitler (user-defined literals)

**## Dinamik Ömürlü Nesneler (Dynamic Objects)**

✅ new ve delete ifadeleri (new & delete expressions)

✅ new[] ve delete [] ifadeleri

✅ operator new işlevleri

✅ operator delete işlevleri

✅ operator new ve operator delete işlevlerinin yüklenmesi

✅ std::bad\_alloc

✅ std::set\_new\_handler ve std::get\_new\_handler işlevleri

✅ placement new operatörleri

✅ nothrow new

**## Tür Eş İsimleri (Type Alias)**

✅ typedef bildirimleri

✅ using bildirimleri

**## İsim Alanları (Namespaces)**

✅ isim alanlarının oluşturulması

✅ isim alanları ve isim arama (namespaces & name lookup)

✅ çözünürlük operatörü ve isim alanları (scope resoşution operator & namespaces)

✅ using bildirimi (using declaration)

✅ using namespace direktifi (using namespace directive)

✅ argümana bağlı isim arama (argument dependent lookup)

✅ isimsiz isim alanı (unnamed namespace)

✅ içsel isim alanları (nested namespaces)

✅ inline isim alanları (inline namespaces)

✅ isim alanı eş ismi (namespace alias)

✅ işlev yüklemesi ve isim alanları (function overloading & namespaces)

**## Sınıflar ve Kalıtım (Classes & Inheritance)**

✅ nesne yönelimli programlama ve kalıtım (OOP & inheritance)

✅ public kalıtımı (public inheritance)

✅ çalışma zamanı çok biçimliliği (runtime polymorphism)

❇️ dinamik ve statik tür bilgisi (static & dynamic type)

❇️ sanal işlevler (virtual function)

❇️ saf sanal işlevler (pure virtual function)

❇️ sanal sonlandırıcı işlev (virtual destructor)

❇️ sanal kurucu işlev idiyomu (virtual constructor idiom)

❇️ override bağlamsal anahtar sözcüğü

❇️ sanal gönderim mekanizmasının implementasyonu (implementation of virtual dispatch mechanism)

❇️ nesne dilimlenmesi (object slicing)

❇️ sanal olmayan arayüz idiyomu (non-virtual interface idiom)

✅ final bağlamsal anahtar sözcüğü (final contextual keyword)

❇️ final sınıflar (final classes)

❇️ \_final override\_

✅ çoklu kalıtım \_multiple inheritance)

❇️ çoklu kalıtımda kapsam ve isim arama(multiple inheritance & name lookup)

❇️ çoklu kalıtımda sınıfın özel işlevleri

❇️ elmas formasyonu (diamond formation)

❇️ sanal kalıtım (virtual inheritance)

❇️ çoklu kalıtım ve kalıtımla alınan kurucu işlevler

❇️ çoklu kalıtımda kopyalama ve taşıma işlemleri

✅ private kalıtımı (private inheritance)

❇️ implementaion inheritance

❇️ empty base optimization

✅ protected kalıtımı (protected inheritance)

✅ sınıf içi using bildirimi

✅ kalıtımla alınan kurucu işlevler (inherited constructors)

✅ mixin sınıflar (mixin classes)

**## Olağan Dışı Durumların İşlenmesi (Exception Handling)**

✅ exception güvenliği (exception safety)

✅ hata nesnelerinin gönderilmesi (throwing exception objects)

❇️ throw deyimi (throw statement)

❇️ rethrow deyimi (rethrow statement)

✅ try blokları

✅ catch blokları

❇️ catch all

✅ yakalanamayan hata nesnesi (uncaught exception)

✅ std::terminate ve std::set\_terminate işlevleri

✅ hata nesnesinin yeniden gönderilmesi (rethrow statement)

✅ yığının geri sarımı (stack unwinding)

✅ kurucu işlevlerden exception gönderimi

✅ sonlandırıcı işlevler ve hata gönderimi

✅ exception handling ve kalıtım (eception handling & inheritance)

✅ exception handling ve dinamik ömürlü sınıf nesneleri (eception handling & dynamic objects)

✅ exception güvenliği için akıllı göstericilerin kullanımı (eception handling & smart pointers)

✅ işlev try blokları (function try block)

✅ noexcept belirleyicisi (noexcept specifier)

✅ beklenmeyen hata nesnesi (unexpected excetion)

✅ std::unexpected\_exception

✅ std::exception sınıfı ve hiyerarşisi

❇️ std::exception sınıfı ve what sanal fonksiyonu

❇️ std::logic\_error

⦿ std::invalid\_argument, std::domain\_error, std::length\_error<, std::out\_of\_range, std::future\_error sınıfları

❇️ std::runtime\_error

⦿ std::range\_error, std::overflow\_error, std::underflow\_error sınıfları

❇️ std::system\_error

❇️ std::regex\_error

❇️ std::bad\_alloc

❇️ std::bad\_typeid

❇️ std::bad\_cast

❇️ std::bad\_exception

❇️ std::bad\_weak\_ptr

❇️ std::bad\_function\_call

✅ kendi hata sınıflarımızı oluşturmak (custom exception classes)

✅ exception garantileri (eception guarantees)

❇️ \_basic exception guarantee\_

❇️ \_strong exception guarantee\_

❇️ \_no throw guarantee\_

✅ std::current\_exception

✅ std::exception\_ptr

✅ std::rethrow\_exception

**## Çalışma Zamanında Tür Belirlenmesi (RTTI)**

✅ dynamic\_cast operatörü

✅ typeid operatörü

✅ std::typeinfo sınıfı

✅ std::bad\_typeid

**## std::string sınıfı**

✅ genel kavramlar

✅ string::size\_type

✅ string::npos

✅ arama işlevleri

✅ set işlemleri

✅ erişim işlemleri

✅ karşılaştırma işlevleri (comparision functions)

✅ sayısal dönüşüm işlevleri (numeric conversions)

✅ küçük string optimizasyonu (small string optimization)

✅ bir STL kabı olaral string sınıfı (string class as STL container)

**## Bileşik Nesneler (Composition)**

✅ öğe olan nesneler ve özel işlevler, kopyalama kontrolü.

✅ öğe olan nesneler ve erişim kontrolü

✅ bileşik nesnelerin kullanıldığı temalar

**## İçsel türler (Type Members)**

✅ sınıf içinde yapılan eş isim bildirimleri

✅ içsel sınıflar (nested classes)

❇️ erişim kontrolü (access control)

❇️ \_pimpl\_ idiyomu (pimpl idiom)

**## Şablonlar (Templates)**

✅ şablon tür parametreleri (template type parameters)

✅ şablon sabit parametreleri (template non-type parameters)

✅ şablon şablon parametreleri (template template parameters)

✅ şablon argümanları (template arguments)

❇️ belirtilmiş template argümanları (explicit template arguments)

❇️ varsayılan template argümanları (default template arguments)

✅ şablonlardan kod üretimi (template instantiation)

✅ fonksiyon şablonları (function templates)

❇️ fonksiyon şablonlarında tür çıkarımı (function template argument deduction)

❇️ fonksiyon şablonlarının yüklenmesi (function template overloading)

✅ sınıf şablonları (class templates)

✅ kurucu işlev ile tür çıkarımı (CTAD) (C++17)

✅ üye şablonlar (member templates)

✅ şablonların özelleştirilmesi (template specialization)

❇️ tam özelleştirme (explicit/full specialization)

❇️ kısmi özelleştirme (partial specialization)

✅ \_sfinae\_

✅ değişken sayıda parametreli şablonlar (variadic templates)

✅ mükemmel gönderim (perfect forwarding)

✅ katlama ifadeleri(C++17) (fold expressions)

✅ \_if constexpr\_

✅ değişken şablonları (variable templates)

✅ eş isim şablonları (alias templates)

**## İteratörler (Iterators)**

✅ aralık kavramı (ranges)

✅ iteratörlerin kategorileri (iterator categories)

❇️ input iterator

❇️ output iterator

❇️ forward iterator

❇️ bidirectional iterator

❇️ random access iterator

✅ kapların begin ve end işlevleri

✅ global begin ve end işlevleri

✅ iterator işlevleri

❇️ std::next

❇️ std::prev

❇️ std::iter\_swap

❇️ std::advance

❇️ std::distance

✅ iterator uyumlandırıcıları (iterator adaptors)

❇️ akım iteratörleri (stream iterators)

⦿ istream\_iterator

⦿ ostream\_iterator

⦿ istreambuf\_iterator

⦿ ostreambuf\_iterator

❇️ reverse iterators

❇️ move iterators

❇️ insert iterators

⦿ back\_insert\_iterator

⦿ front\_insert\_iterator

⦿ insert\_iterator

✅ iterator traits

**## Kaplar (Containers)**

✅ STL kapları ve veri yapıları (STL containers & data structures) \_

✅ sıralı kaplar (sequence containers)

❇️ std::vector

❇️ std::deque

❇️ std::string

❇️ std::array

❇️ std::list

❇️ std::forward\_list

✅ ilişkisel kaplar (associative containers)

❇️ std::set

❇️ std::multiset

❇️ std::map

❇️ std::multimap

✅ sırasız ilişkisel kaplar (unordered containers)

❇️ std::unordered\_set

❇️ std::unordered\_multiset

❇️ std::unordered\_map

❇️ std::unordered\_multimap

✅ kapların tür öğeleri (type members of containers)

✅ kapların emplace işlevleri

**## Kap Uyumlandırıcıları (Container Adaptors)**

✅ std::stack

✅ std::queue

✅ std::priority\_queue

**## Algoritmalar (Algorithms)**

✅ algoritmaların temel özellikleri ve genel ilkeler

✅ salt okuyan algoritmalar \_non-modifying algorithms)

✅ kap öğelerini değiştiren algoritmalar (modifying algorithms)

✅ kap öğelerini konumlandıran algoritmalar (mutating algorithms)

✅ sıralama ile ilgili algoritmalar (sorting algorithms)

✅ sıralanmış aralıklar üzerinde koşturulan algoritmalar (sorted range algorithms)

✅ nümerik algoritmalar (numeric algorithms)

✅ algoritmaların lambda ifadelerini kullanması

**## Lambda İfadeleri (Lambda Expressions)**

✅ kapanış türleri ve kapanış nesneleri (closure types and closure objects)

✅ lambda ifadeleri ve tür çıkarımı (lambda expressions and type deduction)

✅ lambda yakalama ifadeleri (lambda captures)

✅ \_lambda init capture\_

✅ capture this

✅ capture \_\*this\_

✅ mutable lambdalar

✅ trailing return type

✅ genelleştirilmiş lambda ifadeleri (generalized lambda expressions)

✅ algoritmalarda lambda ifadelerinin kullanımı

✅ lambda ifadeleri C++11/14/17/20

✅ lambda idiyomları (lambda idioms)

**## Akıllı Gösterici Sınıfları (Standard Smart Pointer Classes)**

✅ std::unique\_ptr sınıfı

❇️ std::make\_unique işlev şablonu

❇️ std::default\_delete ve custom deleters

❇️ tipik hatalar

✅ std::shared\_ptr sınıfı

❇️ referans sayımı (reference counting)

❇️ std::make\_shared işlev şablonu

❇️ std::weak\_ptr sınıfı

**## Standart Giriş Çıkış Kütüphanesi (iostream Library)**

✅ giriş çıkış akımlarına ilişkin standart sınıflar (standard stream classes)

✅ global akım nesneleri (global stream objects)

✅ formatlı giriş çıkış işlemleri (formatted input output)

✅ << ve >> operatörlerinin yüklenmesi (inserter & extractors)

✅ formatlama ve formatlama işlemleri (formatting)

✅ manipülatörler (manipulators)

✅ akımın durumu (condition states)

✅ string akımları (string streams)

✅ dosya işlemleri (file operations)

✅ formatsız giriş ve çıkış işlemleri (unformatted input output)

✅ bellek üstünde yapılan giriş çıkış işlemleri (in-memory input output operations)

**## Bazı önemli STL Öğelerinin Tanıtımı**

✅ std::pair

✅ std::tuple

✅ std::initializer\_list

✅ std::bitset

✅ std::regex

✅ type\_traits kütüphanesi

✅ std::allocator

✅ std::ratio

✅ std::chrono

✅ standart random kütüphanesi

✅ std::string\_view sınıfı (C++17)

✅ std::optional sınıfı (C++17)

✅ std::variant sınıfı (C++17)

✅ std::any sınıfı (C++17)

✅ std::byte (C++17)

✅ std::invoke

✅ std::apply

**## Tamamlayıcı Araçlar ve Sentaks Öğeleri**

✅ aralık tabanlı for döngüleri (range based for loops)

✅ ilk değer vermeli if ve switch deyimleri (if/switch with initializers)

✅ static\_assert

✅ decltype(auto) tür çıkarımı

✅ std::declval

✅ üye fonksiyon göstericileri (member function pointers)

✅ ham string sabitleri (raw string literals)

✅ ikilik sayı sisteminde yazılan sabitler (binary literals)

✅ basamak ayırıcısı (digit seperator)

✅ ilk değer vermeli if deyimi (C++17) (if with initializer)

✅ ilk değer vermeli switch deyimi (C++17) (if with initializer)

✅ alignas belirteci (alignas specifier)

✅ alignof operatörü (alignof operator)

✅ yapısal bağlama (C++17)(structural binding) \_

✅ attribute’lar (attributes)

**## Concurrency**

✅ memory model

✅ thread’ler ve thread yönetimi

✅ std::this\_thread isim alanı

✅ data race kavramı ve data\_race’den kaçınma

✅ standart mutex sınıfları ve mutex işlemleri

✅ lock\_guard ve unique\_lock sınıfları

✅ std::condition\_variable sınıfı

✅ std::future ve std::promise sınıfları

✅ std::async işlevi

✅ atomik türler (atomic types)

✅ görev tabanlı (task based) programlama

✅ std::packaged\_task sınıfı

✅ paralel STL algoritmaları