**T.C**

**NECMETTİN ERBAKAN ÜNİVERSİTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**



**C# PROGRAMLAMA DİLİNİN İNCELEME RAPORU**

**PROGRAMLAMA DİLLERİNİN KAVRAMLARI DERSİ PROJESİ**

**FIRAT KAYA**

**20010011075**

**PROJE DANIŞMANI: HÜSEYİN HAKLI**

**EKİM 2022 – ARALIK 2022**

## Giriş

## Bilmeniz Gereken Kavramlar Ve Açıklamaları

### Nesne Tabanlı Programlama Dili Nedir?

Nesne yönelimli programlama (NYP), (İngilizce: *Object - Oriented Programming*)(OOP), Her işlevin nesneler olarak soyutlandığı bir programlama yaklaşımıdır. NYP destekleyen programlama dilleri yüksek seviye diller olarak adlandırılır.

1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım dünyasında beliren bir bunalımın sonucudur. Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti zaman ve çaba olarak daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (modularity) benimsemesidir. NYP ayrıca, bilgi gizleme (information hiding), veri soyutlama (data abstraction), çok biçimlilik (polymorphism) ve kalıtım (inheritance) gibi yazılımın bakımını ve aynı yazılım üzerinde birden fazla kişinin çalışmasını kolaylaştıran kavramları da yazılım literatürüne kazandırmıştır. Sağladığı bu avantajlardan dolayı, NYP günümüzde geniş çaplı yazılım projelerinde yaygın olarak kullanılmaktadır.

NYP'nin altında yatan birimselliğin ana fikri, her bilgisayar programının (izlence), etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir, ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. Hâlbuki NYP'den önce var olan tek yaklaşımda (Yordamsal programlama), programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmektedirler.

Günümüzde çok çeşitli nesne tabanlı programlama dilleri olmasıyla beraber, en popüler diller sınıflar üzerine kurulmuşlardır (class-based). Bu dillerde nesneler sınıfların birer üyesidir ve nesnelerin tipini de bu sınıflar belirlerler.

### Ecma Nedir?

European Computer Manufacturers Association (ECMA), Avrupa Bilgisayar Üreticileri Birliği olarak bilinir. ECMA uluslararası özel bir kuruluş olup üyelik sistemine dayalıdır. Bilgi ve iletişim teknolojileri standartlarına yönelik bir dernektir. 1961 yılında kurulmuş olup ECMA standartlar birliği olarak bilinir.

### ISO/IEC Nedir?

International Organization for Standardization (ISO), dünya çapında standartları geliştirmek, uluslararası iletişim ve iş birliğini artırmak ve uluslararası ticaretin adil büyümesini teşvik etmek amacıyla 1947 yılında kurulmuş olan bir sivil toplum kuruluşudur.

IEC, uluslararası standartlar geliştiren ve elektro-teknoloji alanında uygunluk değerlendirme sistemlerini işleten, 1906 yılında İsviçre Kanunlarına göre kurulmuş, kar amacı gütmeyen uluslararası bir sivil toplum kuruluşudur.

### SMC (Simple Managed C) Nedir?

Çok basit bir şekilde ifade etmek gerekirse, yönetilen kod tam da şudur: yürütülmesi bir çalışma zamanı tarafından yönetilen kod. Bu durumda, söz konusu çalışma zamanına, uygulamadan bağımsız olarak (örneğin, Mono, .NET Framework veya .NET Core/.NET 5+) Common Language Runtime veya CLR adı verilir. CLR, yönetilen kodu almaktan, makine kodunda derlemekten ve ardından yürütmekten sorumludur. Bunun da ötesinde, çalışma zamanı, otomatik bellek yönetimi, güvenlik sınırları, tür güvenliği vb. gibi birçok önemli hizmet sunar

## Historical Background – Tarihsel Geçmişi

### Giriş

Microsoft’ta çalışan Anders Hejlsberg Ocak 1999 ‘da C-like Object Oriented Language (Cool) adını verdiği yeni bir nesne tabanlı programlama dili oluşturmak için ekip kurdu. .Net Framework’ün geliştirilmesi sırasında , sınıf kitaplığı olarak SMC adı verilen kod derleyici sistemi kullanılarak yazıldı. Microsoft Cool adını son ad olarak düşünürken telif hakları sorunları yüzünden adını değiştirmeye karar verdi. Adını C++ ‘ın geliştirilmiş ve yükseltilmiş bir devamı olarak nitelendirmek için adını C# ile değiştirdiler.

Microsoft’un Temmuz 2000 tarihinde yaptığı Profesyonel Geliştiriciler Konferansı’nda C# programlama dili kamuya duyuruldu ve sınıf kitaplıkları ve ASP.NET çalışma zamanı da C# ‘a taşındı.

Günümüze kadar C# programlama dili Microsoft tarafından destek verilmeye ve geliştirilmeye devam ediyor. Şuan ki son surum C# 10 olmasına rağmen 2022 yılının sonuna doğru C# 11 sürümünün çıkacağı tahmin ediliyor.

### C#’ın Baş Mimarı Anders Hejlsberg Kimdir?



Resim 1 Anders Hejlsberg

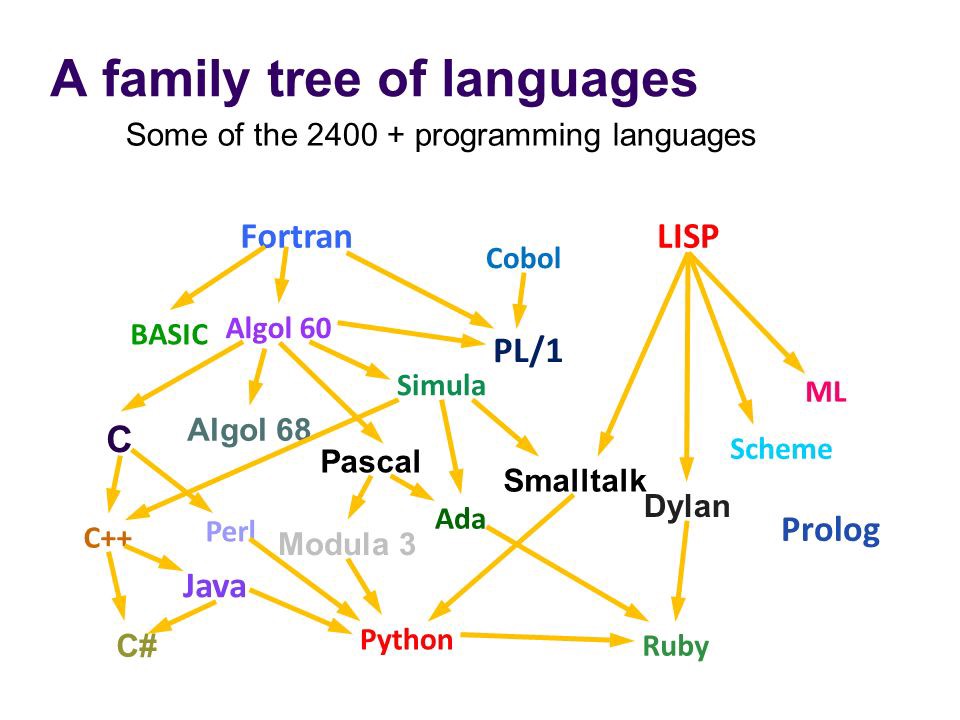
Anders Hejlsberg, 1960 yılında Danimarka’nın Kopenhag kentinde doğdu. Danimarka Teknik Üniversitesi’nde eğitim gördü fakat okulunu bitirmedi. Okurken Nascom mikrobilgisayar için programlar yazmaya başladı.

Danimarkalı Bilgisayar Mühendisi Hejlsberg, Borland’ın ilk çalışanlarından ve Turbo Pascal‘ı yazan kişidir. Birçok popüler ve başarılı programlama dilinin geliştirme araçlarının tasarlanmasına yardım etti. Delphi ürün hattının baş mimari olarak çalıştı. Daha sonra Microsoft’a geçiş yaptı.

Microsoft’ta Visual J++ ve Windows Foundation Classes’i geliştirmede önemli rol oynadı. Ardından COM+ ve VS.Net Framework ekibinin başında çalıştı. Bu ekiple birlikte C# dilini geliştirip derleyicisini yazdı.

### C# Soyağacı

C# programlama dili, C dilinden geliştirilmiş olan C++ ve C++ programlama dilinden geliştirilen Java programlama dilinden etkilenerek ve ilham alınarak geliştirilmiş bir dildir.



Resim 2 - Programla dilleri soyağacı

C# programlama dilinin tarihini tam anlamıyla kavramak için en azından Java ve C++ hakkında biraz bilgiye sahip olmamız gerekiyor.

#### C++ Programlama Dili Nedir?

1979 yılında Bell Laboratuvarlarında Bjarne Stroustrup tarafından geliştirilmiştir. Başlangıçta “C ile Sınıflar” olarak adlandırıldı, ancak daha sonra 1983'te C++ olarak yeniden adlandırıldı. İlk adından da anlaşılacağı gibi, C'nin bu çocuğu nesne yönelimli olmaya odaklanmıştır. Bu programlama dili, nesne yönelimli köklerinin yanı sıra genel programlamayı da kapsayacak şekilde büyümüştür.

#### Java Programlama Dili Nedir?

1990'ların başında James Gosling tarafından geliştirilen Nesne Yönelimli bir programlama dilidir . Ekip, set üstü kutular, televizyon vb. gibi dijital cihazlar için bir dil geliştirmek için bu projeyi başlattı. Başlangıçta projede C++ kullanılması düşünülüyordu ancak fikir çeşitli nedenlerle reddedildi (Örneğin C++ daha fazla bellek gerektiriyordu). Gosling, C++' ı değiştirmek ve genişletmek için çabaladı, ancak çok geçmeden Green adlı başka bir aşamayı yapmaya başladı. James Gosling ve ekibi projelerini “ Greentalk ” olarak adlandırdı ve dosya uzantısı .gt idi ve daha sonra “ OAK ” olarak tanındı. Ancak daha sonra Oak Technologies'in bir ticari markası olduğu için “JAVA” olarak yeniden adlandırmak zorunda kalmışlardır. “JAVA” Gosling ve ekibi bir beyin fırtınası yaptı ve oturumdan sonra JAVA, DNA, SILK, RUBY gibi birçok isim buldular. Java ismine çok özgün olduğu için çok tartışıldıktan sonra karar verildi. Java, Robust, Portable, Platform Independent, High Performance, Multithread vb. ilkeler üzerine oluşturuldu ve TIME MAGAZINE tarafından 1995'in En İyi On Ürününden biri olarak adlandırıldı. Şu anda Java internet programlamasında, mobil cihazlarda, oyunlarda, e-iş çözümlerinde vb. kullanılmaktadır.

### C# Programlama Dilinin Doğuşu İle Ortaya Çıkan Tartışmalar

C# programlama dili, C++ ve Java programlama dillerinden miras aldığı özellikler ile ilk çıktığında yazılım dünyasında birkaç programlama dili geliştiricisi tarafından yenilikten yoksun ve yeni değil olarak nitelendirildi. Sözleri ile en çok dikkat çeken isimler ise Java programlama dilinin yaratıcılarından James Gosling ve Bill Joy ‘du. Bill Joy C# için “Javanın basit bir klonu” demiştir. James Gosling C# programlama dili için “Güvenilirliği, üretkenliği ve güvenliği silinmiş bir tür Java'dır” demiştir. Bu tür söylemler ve tartışmalar diğer geliştiricileri ve programlama dilleri hakkında araştırma yapan ve yazılar, makaleler ve kitaplar yazan kişileri de içine çekmiştir. En popüler olanlarından C++ Akış (Stream) kitabının yazarları Klaus Kreft ve Angelika Langer yayınladıkları bir blog yazısında "Java ve C# neredeyse aynı programlama dilleridir. Yenilikten yoksun sıkıcı tekrarlar" ifadelerini sıkça kullandılar.

Anders Hejlsberg bu tür söylemler için verdiği bir röportajda “Her şeyden önce, C# bir Java klonu değildir. C# tasarımında birçok dile baktık. C++'a baktık, Java'ya, Modula 2, C'ye ve Smalltalk'a baktık. Derin nesne yönelimi, nesne basitleştirme vb. gibi ilgilendiğimiz temel fikirlere sahip çok sayıda dil var.

C# ile bu diğer diller, özellikle Java arasındaki temel farklardan biri, tasarımımızda C++'a çok daha yakın olmaya çalışmamızdır. C#, operatörlerinin, anahtar sözcüklerinin ve deyimlerinin çoğunu doğrudan C++'dan ödünç alır. Ayrıca Java'nın bıraktığı bir dizi dil özelliğini de koruduk. Örneğin, Java'da neden numaralandırma yok? Yani, bunları kesmenin mantığı nedir? Numaralandırmalar açıkça C++'da anlamlı bir kavramdır. Numaralandırmaları C#'da koruduk ve aynı zamanda tip açısından güvenli hale getirdik. C#'da numaralandırmalar yalnızca tamsayılar değildir. Aslında bunlar, .NET temel sınıf kitaplığındaki System.Enum'dan türetilen, kesin olarak yazılmış değer türleridir. "bar" türündeki bir numaralandırma, döküm olmadan "bar" türündeki bir numaralandırma ile değiştirilemez. Bence bu önemli bir fark. Biz' ayrıca operatör aşırı yüklemesini ve tür dönüşümlerini de korudu. Ad alanları için tüm yapımız C++'a çok daha yakındır.” demiştir.

### C# Sürümleri Ve Gelişim Süreci

Tablo 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Language specification | | | Date | .NET | Visual Studio |
| Ecma | ISO/IEC | Microsoft |
| C# 1.0 | ECMA-334:2003, December 2002 | ISO/IEC 23270:2003, April 2003 | January 2002 | January 2002 | .NET Framework 1.0 | Visual Studio .NET 2002 |
| C# 1.1 C# 1.2 | October 2003 | April 2003 | * .NET Framework 1.1 | Visual Studio .NET 2003 |
| C# 2.0 | ECMA-334:2006, June 2006 | ISO/IEC 23270:2006, September 2006 | September 2005 | November 2005 | * .NET Framework 2.0 * .NET Framework 3.0 | Visual Studio 2005 Visual Studio 2008 |
| C# 3.0 | None | | August 2007 | November 2007 | * .NET Framework 2.0 (Except LINQ) * .NET Framework 3.0 (Except LINQ) * .NET Framework 3.5 | Visual Studio 2008 |
| C# 4.0 | April 2010 | April 2010 | * .NET Framework 4 | Visual Studio 2010 |
| C# 5.0 | ECMA-334:2017, December 2017 | ISO/IEC 23270:2018, December 2018 | June 2013 | August 2012 | * .NET Framework 4.5 | Visual Studio 2012 Visual Studio 2013 |
| C# 6.0 | ECMA-334:2022, June 2022 | None | Draft | July 2015 | * .NET Framework 4.6 * .NET Core 1.0 * .NET Core 1.1 | Visual Studio 2015 |
| C# 7.0 | None | | Specification proposal | March 2017 | * .NET Framework 4.7 | Visual Studio 2017 version 15.0 |
| C# 7.1 | Specification proposal | August 2017 | * .NET Core 2.0 | Visual Studio 2017 version 15.3[52] |
| C# 7.2 | Specification proposal | November 2017 |  | Visual Studio 2017 version 15.5 |
| C# 7.3 | Specification proposal | May 2018 | * .NET Core 2.1 * .NET Core 2.2 * .NET Framework 4.8 | Visual Studio 2017 version 15.7 |
| C# 8.0 | Specification proposal | September 2019 | * .NET Core 3.0 * .NET Core 3.1 | Visual Studio 2019 version 16.3 |
| C# 9.0 | Specification proposal | November 2020 | * .NET 5.0 | Visual Studio 2019 version 16.8 |
| C# 10.0 | Specification proposal | November 2021 | * .NET 6.0 * .NET 6.0.1 | Visual Studio 2022 version 17.0 |

#### C# sürüm 1.0

Geri dönüp baktığınızda Visual Studio .NET 2002 ile yayınlanan C# sürüm 1.0, Java'ya çok benzerdi. ECMA için belirtilen tasarım hedeflerinin bir parçası olarak, "basit, modern, genel amaçlı nesne odaklı bir dil" olmaya çalıştı. O zamanlar Java gibi göründüğünde bu erken tasarım hedeflerine ulaşıldığı anlamına geliyordu.

Ancak şimdi C# 1.0'a bakarsanız, biraz başınızı dönersiniz. Yerleşik zaman uyumsuz özelliklerden ve verilen genel türlerle ilgili bazı kaynmış işlevlerden yoksun. Aslında, genel olarak tamamen eksik. Ve LINQ? Henüz kullanılamıyor. Bu eklemelerin ortaya çıkması birkaç yıl sürer.

C# sürüm 1.0, bugünle karşılaştırıldığında özelliklerden arındırılmış görünüyordu. Kendini ayrıntılı bir kod yazarken bulurdun. Ama yine de, bir yerden başlaman gerekiyor. C# sürüm 1.0, Windows platformunda Java'ya uygun bir alternatifti.

C# 1.0'ın başlıca özellikleri şunlardır:

* Sınıflar
* Yapılar
* Arabirimler
* Ekinlikler
* Özellikler
* Temsilciler
* İşleçler ve ifadeler
* Deyimler
* Öznitelikler

#### C# sürüm 1.2

C# sürüm 1.2, Visual Studio .NET 2003 ile birlikte gönderildi. Dille ilgili birkaç küçük geliştirme içeriyor. En önemli olan, bu sürümden başlayarak, uygulandığında üzerinde adlı Dispose bir foreachIEnumerator döngüde oluşturulan kodun olmasıdır IEnumeratorIDisposable.

#### C# sürüm 2.0

Şimdi işler ilginçleşiyor. Visual Studio 2005 ile birlikte 2005'te yayımlanan C# 2.0'ın bazı önemli özelliklerine göz atalım:

* Genel Türler
* Kısmi türler
* Anonim yöntemler
* Boş değer atanabilen değer türleri
* Yineleyicilerde
* Kovaryans ve kontravaryans

Diğer C# 2.0 özellikleri, mevcut özelliklere özellikler ekledi:

* Getter/setter ayrı erişilebilirlik
* Yöntem grubu dönüştürmeleri (temsilciler)
* Statik sınıflar
* Temsilci çıkarımı

C# genel bir Object-Oriented (OO) dili olarak başlamış olsa da, C# sürüm 2.0 bunu aceleyle değiştirdi. Ayakları altlarında olduktan sonra, bazı ciddi geliştirici sorunlarına göz atmışlardı. Ve onlar, önemli bir şekilde peşlerine düştüler.

Genel türler ile türler ve yöntemler, tür güvenliğini korurken rastgele bir tür üzerinde çalışabilir. Örneğin, sahip List<T> olmak List<string> , bu dizelerde veya List<int> tamsayılarda yineleme yaparken tür açısından güvenli işlemler yapmanıza veya gerçekleştirmenize olanak tanır. Genel türleri kullanmak, her işlem için türünden ArrayList türetilen veya türeten türetilen bir ListInt tür oluşturmaktan Object daha iyidir.

C# sürüm 2.0 yineleyiciler getirdi. Bunu kısaca ifade etmek gerekirse, yineleyiciler döngüyle foreach bir List (veya diğer Numaralandırılabilir türler) içindeki tüm öğeleri incelemenize olanak sağlar. Yineleyicilerin dilin birinci sınıf bir parçası olması, dilin okunabilirliğini ve insanların kod hakkında düşünme becerisini önemli ölçüde artırdı.

Yine de C# Java'yı biraz yakalamaya devam etti. Java, genel sürümleri ve yineleyicileri içeren sürümleri zaten yayımlamıştı. Ancak diller birbirinden ayrılmaya devam ettikçe bu durum yakında değişecektir.

#### C# sürüm 3.0

C# sürüm 3.0, Visual Studio 2008 ile birlikte 2007'nin sonlarında geldi, ancak tam sürüm dil özellikleri aslında .NET Framework sürüm 3.5 ile gelecekti. Bu sürüm, C# büyümesinde önemli bir değişiklik olduğunu işaret etti. C# dilini gerçekten çok önemli bir programlama dili olarak kurdu. Şimdi bu sürümdeki bazı önemli özelliklere göz atalım:

* Otomatik uygulanan özellikler
* Anonim türler
* Sorgu ifadeleri
* Lambda ifadeleri
* İfade ağaçları
* Uzantı yöntemleri
* Örtülü olarak belirtilmiş yerel değişkenler
* Kısmi yöntemler
* Nesne ve koleksiyon başlatıcıları

Geçmişe bakıldığında, bu özelliklerin çoğu hem kaçınılmaz hem de ayrılmaz görünüyor. Hepsi stratejik olarak birbirine uyuyor. C# sürümünün en iyi özelliğinin Language-Integrated Sorgusu (LINQ) olarak da bilinen sorgu ifadesi olduğu düşünülüyor.

Daha ince bir görünüm, LINQ'in oluşturulduğu temel olarak ifade ağaçlarını, lambda ifadelerini ve anonim türleri inceler. Ancak her iki durumda da C# 3.0 devrimsel bir kavram sundu. C# 3.0, C# dilini karma bir Object-Oriented / İşlev diline dönüştürmenin temelini oluşturmaya başlamıştı.

Özellikle, artık koleksiyonlar üzerinde işlemler gerçekleştirmek için SQL stilinde bildirim temelli sorgular yazabilirsiniz. Tamsayı listesinin ortalamasını hesaplamak için bir döngü yazmak for yerine, bunu gibi basit bir şekilde list.Average()yapabilirsiniz. Sorgu ifadeleri ve uzantı yöntemlerinin birleşimi, tamsayı listesinin çok daha akıllı hale geldiğini gösteriyordu.

İnsanların kavramı gerçekten kavraması ve tümleştirmesi zaman aldı, ancak aşamalı olarak bunu yaptılar. Ve şimdi, yıllar sonra kod çok daha kısa, basit ve işlevsel.

#### C# sürüm 4.0

Visual Studio 2010 ile yayınlanan C# sürüm 4.0, sürüm 3.0'ın temel oluşturan durumuna kadar zor zamanlar geçirmişti. Sürüm 3.0 ile C# dili Java'nın gölgesinden sıkı bir şekilde çıkararak öne çıkarmıştı. Dil hızla zarifleşiyordu.

Sonraki sürüm bazı ilginç yeni özellikleri tanıttı:

* Dinamik bağlama
* Adlandırılmış/isteğe bağlı bağımsız değişkenler
* Genel kovaryan ve kontravaryan
* Katıştırılmış birlikte çalışma türleri

Katıştırılmış birlikte çalışma türleri, uygulamanız için COM birlikte çalışma derlemeleri oluşturmanın dağıtım acısını kolaylaştırdı. Genel kovaryans ve kontravaryans, genel türleri kullanmanız için size daha fazla güç verir, ancak bunlar biraz akademiktir ve büyük olasılıkla çerçeve ve kitaplık yazarları tarafından takdir edilir. Adlandırılmış ve isteğe bağlı parametreler birçok yöntem aşırı yüklemelerini ortadan kaldırmanıza ve kolaylık sağlamanıza olanak sağlar. Ancak bu özelliklerin hiçbiri tam olarak paradigma değiştirmiyor.

En önemli özellik anahtar sözcüğün dynamic kullanıma sunulmasıydı. dynamic C# sürüm 4.0'da derleme zamanı yazmada derleyiciyi geçersiz kılma özelliğine sahip anahtar sözcüğü. Dinamik anahtar sözcüğünü kullanarak JavaScript gibi dinamik olarak yazılan dillere benzer yapılar oluşturabilirsiniz. Bir oluşturabilir dynamic x = "a string" ve sonra buna altı tane ekleyebilir ve bundan sonra ne olacağını sıralamak için çalışma zamanına bırakabilirsiniz.

Dinamik bağlama, hata olasılığı sunar ancak aynı zamanda dil içinde büyük bir güç sağlar.

#### C# sürüm 5.0

Visual Studio 2012 ile yayımlanan C# sürüm 5.0, dilin odaklanmış bir sürümüdür. Bu sürüm için neredeyse tüm çaba, zaman uyumsuz programlama için ve await modeli olmak üzere çığır açan başka bir dil kavramına async dönüştü. Önemli özellikler listesi aşağıdadır:

* Zaman uyumsuz üyeler
* Arayan bilgisi öznitelikleri

Arayan bilgisi özniteliği, bir ton ortak yansıma koduna başvurmadan çalıştırdığınız bağlam hakkındaki bilgileri kolayca almanıza olanak tanır. Tanılama ve günlüğe kaydetme görevlerinde birçok kullanımı vardır.

await Ama async bu yayının gerçek yıldızları. Bu özellikler 2012'de ortaya çıktığında, C# birinci sınıf bir katılımcı olarak asenkronu dile pişirerek oyunu yeniden değiştirdi. Uzun süre çalışan işlemler ve geri çağırma web'lerinin uygulanmasıyla ilgilendiyseniz, büyük olasılıkla bu dil özelliğini çok sevdiniz.

#### C# sürüm 6.0

3.0 ve 5.0 sürümleriyle C# nesne odaklı bir dilde önemli yeni özellikler eklemişti. Visual Studio 2015 ile yayınlanan sürüm 6.0 ile baskın bir katil özelliğinden uzaklaşır ve bunun yerine C# programlamasını daha üretken hale getiren birçok küçük özellik yayınlar. Bunlardan bazıları şunlardır:

* Statik içeri aktarmalar
* Özel durum filtreleri
* Otomatik özellik başlatıcıları
* Gövdeli ifade üyeleri
* Null yayıcı
* Dize ilişkilendirme
* Nameof işleci

Diğer yeni özellikler şunlardır:

* Dizin başlatıcıları
* Catch/finally bloklarında bekle
* Yalnızca alma özellikleri için varsayılan değerler

Bu özelliklerin her biri kendi başına ilginçtir. Ama bunlara tamamen bakarsanız, ilginç bir desen görürsünüz. Bu sürümde C# kodu daha ters ve okunabilir hale getirmek için dil ortak ortaklarını ortadan kaldırmıştı. Yani temiz, basit kod hayranları için, bu dil sürümü büyük bir kazanç oldu.

Kendi başına geleneksel bir dil özelliği olmasa da bu sürümle birlikte bir şey daha yaptılar. Derleyici roslyn'i bir hizmet olarak yayımladılar. C# derleyicisi artık C# dilinde yazılmıştır ve derleyiciyi programlama çabalarınızın bir parçası olarak kullanabilirsiniz.

#### C# sürüm 7.0

C# sürüm 7.0, Visual Studio 2017 ile yayımlandı. Bu sürümde C# 6.0'ın damarında bazı evrimsel ve seyrek erişimli şeyler bulunur, ancak derleyici hizmet olarak kullanılmamıştır. Yeni özelliklerden bazıları şunlardır:

* Out değişkenleri
* Tanımlama demetleri ve yapısızlaştırma
* Desen eşleştirme
* Yerel işlevler
* Genişletilmiş ifade gövdeli üyeler
* Başvuru yerel öğeleri
* Başvuru dönüşleri

Dahil olan diğer özellikler:

* Atılanlar
* İkili Değişmez Değerler ve Basamak Ayırıcıları
* Throw ifadeleri

Bu özelliklerin tümü geliştiriciler için harika yeni özellikler ve her zamankinden daha temiz kod yazma fırsatı sunar. Vurgulama, anahtar sözcükle kullanılacak değişkenlerin bildirimini out daraltmak ve tanımlama grubu aracılığıyla birden çok dönüş değerine izin vermektir.

Ancak C# daha geniş kullanıma sunuldu. .NET Core artık herhangi bir işletim sistemini hedeflemektedir ve gözlerini buluta ve taşınabilirliğe sıkıca bakmaktadır. Bu yeni özellikler, yeni özelliklerin yanı sıra dil tasarımcılarının düşüncelerini ve zamanını da kaplar.

#### C# sürüm 7.1

C#, C# 7.1 ile *nokta sürümlerini* yayınlamaya başladı. Bu sürümde dil sürümü seçimi yapılandırma öğesi, üç yeni dil özelliği ve yeni derleyici davranışı eklendi.

Bu sürümdeki yeni dil özellikleri şunlardır:

* asyncMain Yöntem
  + Bir uygulamanın giriş noktası değiştiriciye async sahip olabilir.
* default değişmez değer ifadeleri
  + Hedef tür çıkarılabildiğinde varsayılan değer ifadelerinde varsayılan değişmez değer ifadelerini kullanabilirsiniz.
* Çıkarsanan demet öğe adları
  + Tanımlama grubu öğelerinin adları, çoğu durumda tanımlama grubu başlatma işleminden çıkarılabilir.
* Genel tür parametrelerinde desen eşleştirme
  + Türü genel tür parametresi olan değişkenler üzerinde desen eşleştirme ifadeleri kullanabilirsiniz.

Son olarak, derleyicinin iki seçeneği -refout vardır ve -refonly bu seçenek başvuru derlemesi oluşturmayı denetler

#### C# sürüm 7.2

C# 7.2 birkaç küçük dil özelliği ekledi:

* Dizilerdeki stackalloc başlatıcılar.
* Deseni destekleyen herhangi bir türle deyimleri kullanın fixed .
* Sabitlemeden sabit alanlara erişin.
* Yerel değişkenleri yeniden atayın ref .
* Bir yapı sabit olduğunu ve üye yöntemlerine parametre in olarak geçirilmesi gerektiğini belirtmek için türleri bildirinreadonly struct.
* Bir bağımsız değişkenin in başvuru tarafından geçirildiğini ancak çağrılan yöntem tarafından değiştirilmediğini belirtmek için parametrelere değiştiriciyi ekleyin.
* Yöntemin ref readonly değerini başvuruya göre döndürdüğünü ancak bu nesneye yazma işlemine izin vermediğini belirtmek için yöntem dönüşlerinde değiştiriciyi kullanın.
* Yapı türünün yönetilen belleğe doğrudan eriştiği ve her zaman yığın olarak ayrılması gerektiğini belirtmek için türleri bildirin ref struct .
* Ek genel kısıtlamalar kullanın.
* Girintili olmayan adlandırılmış bağımsız değişkenler
  + Adlandırılmış bağımsız değişkenlerin ardından konumsal bağımsız değişkenler gelebilir.
* Sayısal değişmez değerlerde önde gelen alt çizgi
  + Sayısal değişmez değerler artık yazdırılan basamaklardan önce önde gelen alt çizgilere sahip olabilir.
* private protected erişim değiştiricisi
  + Erişim değiştiricisi private protected , aynı derlemedeki türetilmiş sınıflar için erişimi etkinleştirir.
* Koşullu ref ifadeler
  + Koşullu ifadenin (?:) sonucu artık bir başvuru olabilir.

#### C# sürüm 7.3

C# 7.3 sürümünün iki ana teması vardır. Temalardan biri, güvenli kodun güvenli olmayan kod kadar yüksek performanslı olmasını sağlayan özellikler sağlar. İkinci tema, mevcut özelliklerde artımlı geliştirmeler sağlar. Bu sürümde yeni derleyici seçenekleri de eklendi.

Aşağıdaki yeni özellikler güvenli kod için daha iyi performans temasını destekler:

* Sabit alanlara sabitlemeden erişebilirsiniz.
* Yerel değişkenleri yeniden atayabilirsiniz ref .
* Dizilerde stackalloc başlatıcıları kullanabilirsiniz.
* Deyimleri, deseni destekleyen herhangi bir türle kullanabilirsiniz fixed .
* Daha genel kısıtlamalar kullanabilirsiniz.

Mevcut özelliklerde aşağıdaki geliştirmeler yapılmıştır:

* Tanımlama grubu türleriyle ve != test == edebilirsiniz.
* İfade değişkenlerini daha fazla konumda kullanabilirsiniz.
* Otomatik uygulanan özelliklerin yedekleme alanına öznitelikler ekleyebilirsiniz.
* Bağımsız değişkenler farklı in olduğunda yöntem çözümlemesi geliştirildi.
* Aşırı yükleme çözümlemesinde artık daha az belirsiz durum var.

Yeni derleyici seçenekleri şunlardır:

* -publicsign derlemelerinin Açık Kaynak Yazılım (OSS) imzasını etkinleştirmek için.
* -pathmap kaynak dizinleri için eşleme sağlamak için.

#### C# sürüm 8.0

C# 8.0, özellikle .NET Core'a yönelik ilk büyük C# sürümüdür. Bazı özellikler yeni CLR özelliklerini, diğerleri ise yalnızca .NET Core'a eklenen kitaplık türlerini kullanır. C# 8.0, C# diline aşağıdaki özellikleri ve geliştirmeleri ekler:

* Salt okunur üyeler
* Varsayılan arabirim metotları
* Desen eşleştirme geliştirmeleri:
  + İfadeleri değiştirme
  + Özellik desenleri
  + Demet desenleri
  + Konumsal desenler
* Bildirimleri kullanma
* Statik yerel işlevler
* Tek kullanımlık başvuru yapıları
* Boş değer atanabilir başvuru türleri
* Zaman uyumsuz akışlar
* Dizinler ve aralıklar
* Null birleşim ataması
* Yönetilmeyen yapılandırılmış türler
* İç içe ifadelerde Stackalloc
* İlişkilendirilmiş düz metin dizelerinin geliştirilmesi

Varsayılan arabirim üyeleri CLR'de geliştirmeler gerektirir. Bu özellikler .NET Core 3.0 için CLR'ye eklendi. Aralıklar, dizinler ve zaman uyumsuz akışlar için .NET Core 3.0 kitaplıklarında yeni türler gerekir. Derleyicide uygulanırken null atanabilir başvuru türleri, bağımsız değişkenlerin ve dönüş değerlerinin null durumuyla ilgili anlamsal bilgiler sağlamak için kitaplıklara açıklama eklendiğinde çok daha kullanışlıdır. Bu ek açıklamalar .NET Core kitaplıklarına ekleniyor.

#### C# sürüm 9

C# 9, .NET 5 ile yayımlandı. .NET 5 sürümünü hedefleyen tüm derlemeler için varsayılan dil sürümüdür. Aşağıdaki yeni ve gelişmiş özellikleri içerir:

* Kayıtlar
* Yalnızca init ayarlayıcılar
* Üst düzey deyimler
* Desen eşleştirme geliştirmeleri
* Performans ve birlikte çalışma
  + Yerel boyutlu tamsayılar
  + İşlev işaretçileri
  + Localsinit bayrağını göstermeyi engelleme
* Özellikleri sığdırma ve bitirme
  + Hedef türündeki new ifadeler
  + static anonim işlevler
  + Hedef türündeki koşullu ifadeler
  + Birlikte değişken dönüş türleri
  + Döngüler için foreach uzantı GetEnumerator desteği
  + Lambda atma parametreleri
  + Yerel işlevlerlerde öznitelikler
* Kod oluşturucu desteği
  + Modül başlatıcılar
  + Kısmi yöntemler için yeni özellikler

C# 9 önceki sürümlerdeki üç temadan devam eder: seremoniyi kaldırma, algoritmalardan verileri ayırma ve daha fazla yerde daha fazla desen sağlama.

Üst düzey deyimler , ana programınızın daha kolay okunduğu anlamına gelir. Seremoni için daha az ihtiyaç vardır: ad alanı, Program sınıf ve static void Main() hepsi gereksizdir.

girişinde records , eşitlik için değer semantiğini izleyen başvuru türleri için kısa bir söz dizimi sağlanır. Genellikle en düşük davranışı tanımlayan veri kapsayıcılarını tanımlamak için bu türleri kullanacaksınız. Yalnızca init ayarlayıcıları , kayıtlarda yıkıcı olmayan mutasyon (with ifadeler) yeteneği sağlar. C# 9 ayrıca türetilmiş kayıtların sanal yöntemleri geçersiz kılıp temel yöntemin dönüş türünden türetilmiş bir tür döndürebilmesi için birlikte değişken dönüş türleri ekler.

Desen eşleştirme özellikleri çeşitli yollarla genişletilmiştir. Sayısal türler artık *aralık desenlerini* destekliyor. Desenler , orve not desenleri kullanılarak andbirleştirilebilir. Daha karmaşık desenleri netleştirmek için parantezler eklenebilir.

Bir diğer özellik kümesi de C# dilinde yüksek performanslı bilgi işlemi destekler:

* nint ve nuint türleri, hedef CPU'daki yerel boyutlu tamsayı türlerini modeller.
* İşlev işaretçileri temsilci benzeri işlevler sağlarken, temsilci nesnesi oluşturmak için gereken ayırmaları önler.
* Yönergeleri localsinit kaydetmek için yönerge atlanabilir.

Bir diğer geliştirme kümesi de *kod oluşturucuların* işlevsellik eklediği senaryoları destekler:

* Modül başlatıcıları , bir derleme yüklendiğinde çalışma zamanının çağıran yöntemlerdir.
* Kısmi yöntemler yeni erişilebilir değiştiricileri ve geçersiz olmayan dönüş türlerini destekler. Böyle durumlarda bir uygulama sağlanmalıdır.

C# 9, hem kod yazma hem de okuma amacıyla geliştirici üretkenliğini geliştiren diğer birçok küçük özelliği ekler:

* Hedef tür new ifadeleri
* static anonim işlevler
* Hedef türü koşullu ifadeler
* Döngüler için foreach uzantı GetEnumerator() desteği
* Lambda ifadeleri atma parametrelerini bildirebilir
* Öznitelikler yerel işlevlere uygulanabilir

C# 9 sürümü, C# dilini modern, genel amaçlı bir programlama dili olarak tutmak için çalışmaya devam eder. Özellikler modern iş yüklerini ve uygulama türlerini desteklemeye devam eder.

#### C# sürüm 10

C# 10, C# diline aşağıdaki özellikleri ve geliştirmeleri ekler:

* Kayıt yapıları
* Yapı türleri geliştirmeleri
* İlişkilendirilmiş dize işleyicileri
* Global using Yönerge -leri
* Dosya kapsamlı ad alanı bildirimi
* Genişletilmiş özellik desenleri
* Lambda ifadelerinde geliştirmeler
* İlişkilendirilmiş dizelere izin ver const
* Kayıt türleri mühürleyebilir ToString()
* İyileştirilmiş kesin atama
* Aynı yapısızlaştırmada hem atamaya hem de bildirime izin ver
* Yöntemlerde özniteliğine izin ver AsyncMethodBuilder
* CallerArgumentExpression özniteliği
* #line Geliştirilmiş pragma

*Ek özellikler önizleme* modunda kullanılabilir. Bu özellikleri denemeniz ve bunlar hakkında geri bildirim sağlamanız teşvik edilir. Son sürümden önce değişebilirler. Bu özellikleri kullanmak için projenizde olarak ayarlamanız <LangVersion>Preview gerekir:

* Bu makalenin devamında genel öznitelikler.
* arabirimlerdeki statik soyut üyeler

C# 10, .NET Çalışma Zamanı için seremoni kaldırma, algoritmalardan veri ayırma ve gelişmiş performans temaları üzerinde çalışmaya devam eder.

Özelliklerin çoğu, aynı kavramları ifade etmek için daha az kod yazabileceğiniz anlamına gelir. *Kayıt yapıları,kayıt sınıflarının* yaptığı yöntemlerin çoğunu sentezler. Yapılar ve anonim türler *ifadelerle birlikte* desteklenir. *Genel kullanım yönergeleri* ve *dosya kapsamlı ad alanı bildirimleri* , bağımlılıkları ve ad alanı kuruluşunu daha net ifade ettiğiniz anlamına gelir. *Lambda geliştirmeleri* , lambda ifadelerini kullanıldıkları yerde bildirmeyi kolaylaştırır. Yeni özellik desenleri ve yapısızlaştırma geliştirmeleri daha kısa kodlar oluşturur.

Yeni ilişkilendirilmiş dize işleyicileri ve AsyncMethodBuilder davranışı performansı artırabilir. .NET 6'da performans geliştirmeleri elde etmek için .NET Çalışma Zamanı'nda bu dil özelliklerinden yararlanılmıştır.

C# 10 ayrıca .NET sürümleri için yıllık tempoya daha fazla geçiş işareti ekler. Her özellik yıllık bir zaman diliminde tamamlanamadığından, C# 10'da birkaç "önizleme" özelliğini deneyebilirsiniz. Arabirimlerdeki hem *genel öznitelikler* hem *de statik soyut üyeler* kullanılabilir, ancak bunlar önizleme özellikleridir ve son sürümlerinden önce değişebilir.

## Design Process – Tasarım Süreci