**EGEDERT++ DİLİ İÇİN KULLANIM KILAVUZU**



**Tasarlayanlar**

Caner Aktaş – 05210000288

Salih Ali Kamacı – 05210000281

Berkay Tülü – 05210000227

Arda Bozkurt - 05200000099

**GİRİŞ**

EGEDERT++ programlama dili spesifik olarak Türk programlama öğrencileri için tasarlanmıştır. EGEDERT++ programlama dili Bilişim Teknolojileri üstüne uzmanlaşmak isteyen lise öğrencileri, üniversite öğrencileri veya çoktan kendini Bilişim Teknolojilerinde geliştirmiş kişilere bile uygun olacak şekilde tasarlanmıştır. Bilişim Teknolojilerinde uzmanlaşmayan kişiler, hatta programlamaya yeni başlamak isteyen farklı uzmanlık alanlarında çalışan kişiler de dahil olmak üzere tüm kişilere uygun olması amaçlanmıştır. Bu kullanım kılavuzu, EGEDERT++ dilinin sözdizimini ve anlamlarını açıklamayı ve kullanıcıların dili sağlam bir şekilde anlamalarını amaçlamaktadır.

***İÇİNDEKİLER***

**BÖLÜM 1: EGEDERT++ Diline Giriş**

* 1.1: EGEDERT++ Dili Hakkında
* 1.2: EGEDERT++ Dilini Nasıl Yüklerim?
* 1.3: EGEDERT++ Dilinde HELLO WORLD!

**BÖLÜM 2: EGEDERT++ Dilinin Sözdizimi**

* 2.1: EGEDERT++ Dilinin İfadeleri
* 2.2: EGEDERT++ Dilinde Değişkenler ve Veri Tipleri
* 2.3: EGEDERT++ Dilinde Operatörler
* 2.4: EGEDERT++ Dilinde Kontrol Akışı

**BÖLÜM 3: EGEDERT++ Dilinde Fonksiyonlar**

* 3.1: EGEDERT++ Dilinde Fonksiyonlara Giriş
* 3.2: EGEDERT++ Dilinde Kullanıcı Tanımlı Fonksiyonlar

**BÖLÜM 4: EGEDERT++ Dilinde Diziler**

* 4.1: EGEDERT++ Dilinde Dizilere Giriş

**BÖLÜM 5: EGEDERT++ Dilinde Girdi ve Çıktılar**

* 5.1: EGEDERT++ Dilinde Kullanıcıdan Girdi Almak
* 5.2: EGEDERT++ Dilinde Kullanıcıya Çıktı Vermek

**BÖLÜM 6: EGEDERT++ Dili için Kütüphaneler**

* 6.1: EGEDERT++ Dilinde Hazır Bulunan Kütüphaneler
* 6.2: EGEDERT++ Diline Dışarıdan Kütüphane Yüklemek

**BÖLÜM 7: EGEDERT++ Dilinin Grammerı**

**BÖLÜM 1: EGEDERT++ Diline Giriş**

Öncelikle dilimize gösterdiğiniz ilgi için teşekkür ederiz! Umarız ki bu kullanım kılavuzu dilimizi daha çabuk kavramanıza ve bu dille gelecekte çok iyi işlere imza atmanıza yardımcı olur. İyi eğlenceler!

**1.1: EGEDERT++ Dili Hakkında**

EGEDERT++ programlama dili, Türk programlama öğrencileri için özel olarak tasarlanmış bir dil olarak karşımıza çıkmaktadır. EGEDERT++ dili, Bilişim Teknolojileri üzerine uzmanlaşmak isteyen lise ve üniversite öğrencileri için ideal bir başlangıç dilidir. Ancak, dilin tasarımı, Bilişim Teknolojilerinde uzmanlaşmayan kişilere, hatta programlamaya yeni başlamak isteyen farklı uzmanlık alanlarında çalışan kişilere de uygun olacak şekilde yapılmıştır.

EGEDERT++ dilinin amacı, kullanıcılarına programlama kavramlarını anlamalarını sağlayarak temel programlama becerilerini geliştirmelerine yardımcı olmaktır. Dilin tasarımı ve sözdizimi, programlamaya yeni başlayanların rahatlıkla öğrenip anlayabileceği şekilde basit ve anlaşılır bir yapıya sahiptir. Kullanıcı dostu bir dil olması, öğrenme sürecini kolaylaştırmakta ve dilin yaygın olarak kullanılmasını teşvik etmektedir.

EGEDERT++ dilinin gelecekteki planları, Türk programlama öğrencilerine daha fazla kaynak ve eğitim materyali sağlamak için sürekli olarak geliştirilmektedir. Kullanıcıların dil hakkında daha fazla bilgi edinebilmeleri ve becerilerini geliştirebilmeleri için kapsamlı bir kullanım kılavuzu ve eğitim kaynakları sunulmaktadır. Ayrıca, dilin kullanıcı kitlesinin geri bildirimlerini dikkate alarak dilin iyileştirilmesi ve yeni özelliklerin eklenmesi de planlanmaktadır.

EGEDERT++ dilinin hedefi, Türk programlama öğrencilerinin ve ilgilenen herkesin programlama konusunda güçlü bir temel oluşturmasını sağlamaktır. Bu dil sayesinde kullanıcılar, mantıksal düşünme becerilerini geliştirerek problemleri çözebilme yeteneklerini artırabilir ve gelecekteki kariyerleri için güçlü bir temel atabilirler.

EGEDERT++ dilinin kullanıcı dostu yapısı, Türk programlama öğrencilerinin programlamayı öğrenme sürecini kolaylaştırırken aynı zamanda programlama alanına ilgi duyan herkesin de dilin faydalarından yararlanmasını sağlar. Bu dil, programlamaya yeni başlayanların heyecanını ve merakını korurken aynı zamanda ileri seviye öğrencilerin de daha karmaşık projeler geliştirmelerine olanak sağlar.

EGEDERT++ dilinin gelecekteki planları, kullanıcıların dil hakkındaki geri bildirimlerini dikkate alarak dilin geliştirilmesi ve yaygınlaştırılmasıdır.

**1.2: EGEDERT++ Dilini Nasıl Yüklerim?**

* **Adım 1: EGEDERT++ Yükleyicisini İndirin**

www.egedertplusplus.com/downloads web sitesine gidin ve en son sürüm yükleyicimizi indirin.

* **Adım 2: Yükleyiciyi Çalıştırma**

İndirdiğiniz dosyayı bulun ve çalıştırın.

* **Adım 3: Yükleyici Seçenekleri**

Yükleyici, size EGEDERT++ programlama dilini sisteminize kurmak için gereken adımları gösterecektir. Adım adım talimatları izleyerek kurulum sürecini kişisel tercihlerinize göre tamamlayın.

* **Adım 4: Yükleme Tamamlandı**

Kurulum tamamlandığında, EGEDERT++ programlama dilinin başarıyla yüklendiği hakkında bir onay mesajı alacaksınız. Artık EGEDERT++ dilini kullanmaya hazırsınız!

**1.3: EGEDERT++ Dilinde MERHABA DÜNYA!**

EGEDERT++ dilinin programlamaya yeni başlayanlar için bile uygun olduğunu belirtmiştik. Şimdi ise kanıtlama zamanı.

EGEDERT++ dilinde “Merhaba Dünya!” yazısını konsola yazdırmak için basitçe;

yaz "Merhaba, Dünya!"

kullanmanız yeterli olacaktır. Kullandığınız bu komut konsola Merhaba, Dünya! cümlesini sizin için yazdıracak.

Burada kullandığımız yaz kelimesini bir fonksiyon olarak adlandırıyoruz. yaz EGEDERT++ dilindeki onlarca fonksiyondan sadece bir tanesi. Kendinizi hazırlayın, daha yeni başlıyoruz.

Merhaba, Dünya yazısının neden çift tırnak arasında olduğunu merak ediyor olabilirsiniz. Bunun sebebi EGEDERT++ dilinde metin (string) tanımının bu şekilde yapılıyor olmasıdır. Bu türleri ileride daha detaylı olarak anlatacağız.

**BÖLÜM 2: EGEDERT++ Dilinin Sözdizimi**

**2.1: EGEDERT++ Dilinin İfadeleri**

* **İfadeler**

EGEDERT++ dilinde belirli bir eylemi gerçekleştiren talimata ifade adı verilir. EGEDERT++ dilinde desteklenen bazı ifadeler şunlardır:

* + **Tanımlama İfadesi**

Tanımlama ifadesi, bir değişkene değer atar. Aşağıdaki sözdizimini takip eder;

<tanımlayıcı> “=” <ifade>

Örneğin;

x = 5

* + **Yazdırma İfadesi**

Yazdır ifadesi sonrasında gelen ifadeyi konsola yazdırma işlemini yapar. Aşağıdaki sözdizimini takip eder;

yaz <ifade>

İfade değerlendirilir ve konsola yazdırılır.

EGEDERT++’ta ifadeler hesaplamaları veya değerleri temsil eder. Sabit değerler, değişkenler veya aritmetik işlemlerden oluşabilirler. Desteklenen bazı ifade türleri şunlardır;

* + **Aritmetik İfadeler**

Bir aritmetik ifade toplama (+), çıkarma (-), çarpma (x) ve bölme (/) gibi aritmetik operatörleri kullanarak matematiksel hesaplamalar yapar. İfadeler bu işleçlerle birleştirilmiş terimlerden ve faktörlerden oluşabilir. Örneğin;

5 + 7 x 8

* + **Karşılaştırma İfadeleri**

Bir karşılaştırma ifadesi iki değeri karşılaştırır ve bir boole değeri (doğru veya yanlış) olarak geri döndürür. “==” (eşittir), “!=” (eşit değildir), “>” (büyüktür), “<” (küçüktür), “>=” (büyüktür veya eşittir), “<=” (küçüktür veya eşittir) gibi karşılaştırma operatörlerini kullanır.

* + **Parantez İçi**

Parantez içindeki ifadeler işlem sırasını veya değerlendirme sırasını değiştirmek için kullanılır. Örneğin;

5 + 7 x 8 işleminin sonucu 61 iken,

(5 + 7) x 8 işleminin sonucu 96 dır.

* + **Sabit**

Sabit değer sayı veya dize gibi sabit bir değeri temsil eder. Örneğin;

5

“Merhaba, Dünya”

* + **Değişken**

Değişken bir değeri tutan adlandırılmış bir depolama konumunu temsil eder. Değişkenler genelde bir tanımlama ifadesi kullanılarak atanırlar ve ifadelerde kullanılabilirler. Örneğin;

x = 5 tanımlandığında x bir değişken haline gelir

15 + x ise 20 sonucunu verir

**2.2: EGEDERT++ Dilinde Değişkenler ve Veri Tipler**

EGEDERT++ dilinde, değişkenler ve veri tiplerine giriş;

* **Değişkenler**
  + **İsimlendirme**

EGEDERT++ dilindeki değişkenler değerleri tutan adlandırılmış bir depolama konumunu temsil eder. Programın yürütülmesi boyunca değişkenlere farklı değerler tanımlanabilir. Değişkenleri adlandırırken aşağıdaki kurallar geçerlidir;

* + - Değişken isimleri harf ile başlamak zorundadır.
    - Harf, sayı ve alt çizgi içerebilirler.
    - Değişken isimler büyük ve küçük harfe duyarlıdır. Yani “değişken” ve “Değişken” farklı değişkenleri temsil ederler.
  + **Tanımlama ve Öndeğer Atama**
    - EGEDERT++ dilinde bir değişkene öndeğer atamak için veri tipini belirtmenize gerek yoktur. Örneğin;

EGEDERT++ x = 5 ‘in bir sayı değeri olduğunu bilir

Aynı şekilde x = “5” ‘in bir metin olduğunu bilir

* + - Değişkenleri tanımlamak için öndeğer atamanın zorunlu olmadığını belirttik, ancak değer atanmadığı zaman değişkenin veri türünün belirtilmesi gerekir. Örneğin;

tamsayı x

metin “deneme”

Yukarıdaki örneklerde “tamsayı” ve “metin” EGEDERT++’ta bulunan birer veri türüdür.

* + - EGEDERT++’ta “tamsayı”, “ondalıksayı”, “doğru”, “yanlış”, “metin” gibi veri türleri bulunur.
  + **Desteklenen Veri Türleri**

EGEDERT++ aşağıdaki veri türlerini destekler;

* + - **Tam sayılar**

Tam sayıları herhangi bir sınırlama olmadan depolar. Örneğin: 5, 10, 12, -5, -8…

* + - **Ondalık Sayılar**

Ondalık sayıları 10 basamağa kadar depolayabilir. Örneğin; 3.141592653

* + - **Metinler**

Metinler, birer karakter dizileridir. EGEDERT++, Unicode karakterleri destekler. EGEDERT++’ta metinler çift tırnak kullanarak temsil edilirler. Örneğin; “Bu bir metin!” veya “Merhaba, Dünya!”

* + - **Doğru ve Yanlış (Boole)**

EGEDERT++’ta değişkenler verileri depolamak ve işlemek için gereklidir. Değerleri gerektiği gibi saklamanıza ve güncellemenize olanak vererek programınıza esneklik sağlayabilirsiniz.

**2.3: EGEDERT++ Dilinde Operatörler**

* **Aritmetik Operatörler**

EGEDERT++ sayısal değerler üzerinde matematiksel hesaplamalar yapmak için aşağıdaki aritmetik operatörleri destekler;

* + Toplama (+): İki değeri toplar
  + Çıkartma (-): İkinci değeri ilk değerden çıkartır
  + Çarpma (\*): İki değeri birbiriyle çarpar
  + Bölme (/): Birinci değeri ikinci değere böler
  + Üs alma (^): İlk değerin ikinci değer katını alır
  + Mod alma (%): Birinci değer ile ikinci değer arasındaki bölümün kalanını hesaplar

Örneğin;

x = 10

y = 5

gibi tanımlanmış değerleriniz varsa;

z = x + y

işlemi z değişkenini x ve y’nin toplamına tanımlar. x ve y değişkenlerinin değerlerine göre z burada 15’e tanımlanmış olur. Çıkartma için ise;

z = x – y

işlemi yapıldığında z daha önce tanımlanmış olduğu halde **2.2 Değişkenler** kısmında bahsedildiği gibi daha sonra tekrar farklı bir değere atanabilir. Bu örnekte z 15 değerinden 10 değerine tanımlandı. Çarpma için ise yeni bir değer tanımlayalım;

tamsayı k

k = x \* y

işleminde k’yi bir tamsayı olarak tanımladık ancak bir tamsayı değeri vermedik. Daha sonra k’yi x ve y’nin çarpımı olarak tanımladığımız için k burada 50 olarak tanımlandı. Kısacası EGEDERT++’da matematik işlemi yapabilmeniz için tanımlanmış değer veya sayı kullanmanız gerekiyor. Eğer bir değişkenin sadece veri tipini belirtip işlem yapmaya çalışırsanız hata ile karşılaşırsınız;

tamsayı m

yaz m + 5

Yukarıdaki kod bloğu konsola hata döndürür çünkü m’e bir değer ataması yapılmadı. Hatalar ve hata kontrolleri başka bir başlık altında daha detaylı incelenecek.

* **Mantık Operatörleri**

EGEDERT++ mantık operatörlerini destekler. Mantık operatörleri boole değerler ile çalışır. EGEDERT++ ‘ın desteklediği mantıksal operatörler şunlardır;

* + **VE**

Eğer her iki işlenen de doğruysa, doğru değerini döndürür. Örneğin;

a = doğru

b = yanlış

gibi iki boole değeriniz var ise VE işlemini aşağıdaki gibi yapabilirsiniz;

c = a ve b

Bu işlem sonrası c değeri yanlış olarak geri döner.

* + **VEYA**

İşlenenlerden en az biri doğruysa doğru değerini döndürür. Örneğin;

a = doğru

b = yanlış

gibi iki boole değeriniz var ise VE işlemini aşağıdaki gibi yapabilirsiniz;

d = a veya b

Bu işlem sonrası d değeri doğru olarak geri döner.

* + **DEĞİL**

İşlenenin değerinin tersini alır. İşlenen değer doğru ise yanlış, yanlış ise doğru döndürür. Örneğin;

a = doğru

e = değil a

Bu işlem sonrası e değeri yanlış olarak geri döner.

* **Tanımlama Operatörü**

EGEDERT++’ta tanımlama operatörü olan “=” değişkenlere değer atamak için kullanılır. Örneğin;

x = 314

Bu örnekte “=” operatörü x değişkenine değerini tanımlıyor.

Bu operatörler EGEDERT++ kullanarak yazdığınız programlarınızda hesaplamalar yapmanıza olanak tanır. Bu operatörleri anlamak çeşitli hesaplamaları gerçekleştirmek için önemlidir.

**2.4: EGEDERT++ Dilinde Kontrol Akışı**

EGEDERT++ dilindeki kontrol akışı ifadeleri programınızın akışını kontrol etmenize olanak tanır. Belirli koşullara veya döngü gereksinimlerine göre ifadelerin yürütülme sırasını belirlerler. EGEDERT++’ta desteklenen kontrol akışı ifadeleri şunlardır;

* + **Eğer İfadesi**

Eğer ifadesi bir kod parçasının koşuluna bağlı olarak yürütülmesini kontrol eder. Belirli bir koşul doğruysa belirli bir kod bloğunu, veya koşul yanlışsa başka bir kod bloğunu yürütür. Aşağıdaki sözdizimini takip eder;

eğer <koşul> ise

<kod bloğu>

aksi\_takdirde

<kod bloğu>

bitir

Bu kod parçası eğer durum geçerli ise kod bloğundaki işlemleri yürütür. Eğer durum geçerli değilse aksi takdirde kısmındaki kod bloğundaki işlemler yürütülür. <koşul> ifadesinin sonucu bir boole ifade olmalıdır. Çünkü eğer ifadesi boole ifadesinin doğruluğuna veya yanlışlığına göre kod bloklarını çalıştırır.

Eğer ifadesinde aksi takdirde kısmı tercihe bağlıdır ve her eğer ifadesine eklenmek zorunda değildir. Kodu okuyan okuyucu bitir ifadesine geldiği zaman yukarıdaki işlemleri okur ve istenen işlemleri tamamlar.

* + **Sırasında Döngüsü**

Sırasında döngüsü sağlanan koşul geçerli olduğu süre boyunca içinde bulunan kod bloğunu sürekli çalıştırır. Aşağıdaki sözdizimini takip eder;

sırasında <koşul> ise

<kod bloğu>

bitir

<koşul> her yinelemeden önce değerlendirilen bir boole ifadesini temsil eder. Koşul doğruysa kod bloğu yürütülür. Döngü koşul yanlış olana kadar devam eder. Kod bloğu içerisinde koşul kısmını etkileyen durumlar varsa her bitir’e ulaşıldığında koşul yeniden değerlendirilir ve eğer sağlanılıyorsa kod bloğu yeniden yürütülür.

* + **İçin Döngüsü**

İçin döngüsü belirtilen koşullar sağlanana kadar devam eden bir döngü türüdür. Aşağıdaki sözdizimini takip eder;

i = <sayı> için i = <sayı> olana kadar i <operatör> <sayı>

<kod bloğu>

bitir

Örneğin;

i = 0 için i = 5 olana kadar i + 1

<kod bloğu>

Bitir

Yukarıdaki örnekte i değerini 0 olarak tanımladık. İçin döngüsü belirtildiği gibi i değeri 5’e ulaşana kadar her döngü aşamasında i’yi 1 kere arttırır ve <kod bloğu> içinde bulunan işlemleri her döngüde uygular. i değeri 5’e ulaştığında <kod bloğu> çalışmaz çünkü için döngüsüne i değeri 5 olana çalışması söylendi.

**BÖLÜM 3: EGEDERT++ Dilinde Fonksiyonlar**

**3.1: EGEDERT++ Dilinde Fonksiyonlara Giriş**

Fonksiyonlar, kodun yeniden kullanımını sağlarlar. Bir kod bloğunu birden fazla yerde kullanmayı düşünüyorsanız, fonksiyon kullanmayı deneyebilirsiniz. Basitçe anlatmak gerekirse, fonksiyonlar bir program boyunca birden çok kez çağrılabilen ve yürütülebilen bir dizi talimata verilen addır.

* **Fonksiyonlar ve Kodun Yeniden Kullanılabilirliği**

Fonksiyonlar gerektiğinde çağırabileceğiniz belirli talimatları depolayarak aynı kodun yeniden kullanılabilirliğini destekler.

Aynı kodu birden çok kez yazmak yerine, bir işlevi bir kez tanımlayıp programınızın çeşitli yerlerinde yeniden kullanabilirsiniz. Bu, yalnızca geliştirme süresinden tasarruf etmekle kalmaz, aynı zamanda kodun bakımını ve okunabilirliğini de arttırır. Kodun fonksiyonlar halinde geliştirilmesi aynı zamanda takım çalışmasında oluşabilecek karışıklıkları da ortadan kaldırır.

* **EGEDERT++’ta “yaz” Fonksiyonu**

Mesajları ekranda görüntülemek için kullanılan bu fonksiyon yaz kelimesinin ardından gelen <ifade> ile çalışır. Biz fonksiyondan sonra gelen ifadelere parametre diyoruz.

Fonksiyonlar, parametrelerin makinenin girdileri olduğu, belirli bir görevi yerine getiren küçük makineler olarak düşünülebilir. Bazı fonksiyonların çağırılabilmesi için parametrelere ihtiyacı yoktur. yaz fonksiyonu da bu fonksiyonlardan biridir.

EGEDERT++’ta fonksiyonları açıklamak için biraz daha derinlere inelim.

* **Dışarıdan Fonksiyon Aktarma**

EGEDERT++’ta yerleşik olarak bulunan ve hemen kullanıma sunulan birçok fonksiyon vardır. Ancak diğerlerini kullanabilmeniz için önce içe aktarmanız gerekir.

EGEDERT++’ta içe fonksiyon aktarabilmeniz için aktarmak istediğiniz kütüphaneyi ve fonksiyonun ismini bilmeniz yeterlidir.

Kütüphaneyi ve fonksiyonu bildikten sonra basitçe;

<kütüphane> den <fonksiyon> aktar

yazmanız yeterlidir. Bu komutu yazdıktan sonra fonksiyonunuzu kodunuzun istediğiniz kısmında kullanabilirsiniz.

**3.2: EGEDERT++ Dilinde Yeni Fonksiyon Tanımlama**

EGEDERT++ dilinde yeni bir fonksiyon tanımlanması ve kullanılması belirli adımları takip eder. Aşağıdaki adımlardan sonra EGEDERT++’ta nasıl fonksiyon tanımlandığını öğrenmiş olacaksınız.

* **1.Adım: Fonksiyon Tanımlamak**

“fonksiyon” anahtar kelimesi, ardından fonksiyon adı daha sonra ise fonksiyon parametrelerini parantez içinde belirtin. EGEDERT++’ın kod bloğuna ulaştığını anlaması için “başla” kelimesini kullanın. “başla” kelimesinden sonra istediğiniz kod bloğunu fonksiyonunuzun içine yazabilirsiniz. Kod bloğunuzu tamamladığınız zaman fonksiyonun sonunu işaretlemek için “bitir” anahtar kelimesini kullanın. EGEDERT++’ta fonksiyon tanımlamak için aşağıdaki sözdizimini takip edebilirsiniz;

fonksiyon <fonksiyon adı> (<parametreler>)

başla

<kod bloğu>

bitir

* **2.Adım: Fonksiyonu Çağırmak**

Tanımladığınız fonksiyonu kullanmak için fonksiyonunuzun adını yazıp daha sonra eğer varsa parantez içinde parametrelerini belirtmeniz yeterlidir. 2 tane metni birleştiren bir fonksiyon tanımladığınızı düşünün. Bu fonksiyonu çağırmak için öncelikle uygun parametreleri tanımlamalısınız;

x = “Merhaba,”

y = “ Dünya!”

Fonksiyonunuzu çağırmak için gerekli parametreleriniz olduktan sonra basitçe fonksiyonunuzun adını yazıp parametreleri yerleştirmeniz yeterli olacaktır. Fonksiyonumuzun adının “birleştir” olduğunu varsayarsak;

yaz birleştir(x, y)

Yukarıdaki satır konsola “Merhaba, Dünya!” çıktısını verecektir. Yukarıda yaz fonksiyonunu da kullandığımızı ve yaz fonksiyonunun parametresi olarak birleştir fonksiyonunu kullandığımızı fark etmiş olabilirsiniz. Burada yaz fonksiyonunun parametresi olarak birleştir kullanmış olsak da birleştir, yaz fonksiyonunun döndüreceği bir veri türü değildir. Birleştir fonksiyonu bize 2 metinin birleşimi olan 1 metin döndürür ve yaz fonksiyonu da metin veri türünü konsola yazdırır.

Bu 2 adım sonrası fonksiyonunuzu tanımlayıp çağırmaya hazırsınız!

Şimdi ise parametrelere ve geri döndürülen tipleri detaylıca inceleyelim.

* **Fonksiyon Parametreleri ve Dönüş Değerleri**
  + **Parametreler**

Fonksiyon parametreleri, parametre olarak belirtilen değerleri tutarlar. Parametreler, fonksiyonların girdileri kabul etmesine ve dinamik verilerle çalışmasına izin verirler. Birden fazla parametre kullanıyorsanız virgül ile ayırabilirsiniz.

* + **Dönüş Değerleri**

Dönüş değerleri bir fonksiyonun ürettiği değerlerdir. Fonksiyonların dönüş değerleri olabilir veya olmayabilir. EGEDERT++’ta bir dönüş değerini belirtmek için “döndür” anahtar kelimesini ve ardından geri döndürülecek değeri kullanın. Döndürülecek değer, EGEDERT++’ta desteklenen herhangi bir veri türünde olabilir.

Parametreler ve dönüş değerleri fonksiyonların önemli birer parçalarıdır. Parametre ve dönüş değerlerini örnekler üzerinde inceleyelim. Bir toplama fonksiyonu tanımlamak istediğinizi düşünün. Fonksiyona 2 adet sayı vermek ve fonksiyondan bu 2 sayıyı toplamasını istiyorsunuz. Fonksiyondan da dönüş olarak bu iki sayının toplamını istiyorsunuz. Bu isteğimizi şimdi koda dökelim;

fonksiyon topla(a, b)

başla

toplam = a + b

döndür toplam

bitir

Yukarıda parametreli ve dönüş değeri olan bir fonksiyon oluşturduk. Şimdi ise parametreli ancak dönüş değeri olmayan bir fonksiyon oluşturalım. Bir metin birleştirme fonksiyonu oluşturmak istediğinizi düşünün. Fonksiyona 2 adet metin vermek ve fonksiyondan bu 2 metni birleştirmesini istiyorsunuz. Daha sonra fonksiyonun birleşmiş metni konsola yazdırmasını istiyorsunuz. Bu isteğimizi şimdi koda dökelim;

fonksiyon birleştir(“metin1", “metin2”)

başla

yaz “metin1”, “metin2”

bitir

Bu ise parametreli ve dönüş değeri olmayan bir fonksiyon örneği.

Fonksiyon tanımlamak programcının yeniden kullanılabilir kod blokları oluşturmasına olanak sağlar. Yukarıdaki adımları izleyerek EGEDERT++ dilinde fonksiyonları tanımlayabilir ve kullanabilirsiniz!

**BÖLÜM 4: EGEDERT++ Dilinde Diziler**

**4.1: EGEDERT++ Dilinde Dizilere Giriş**

Diziler EGEDERT++ dili de dahil olmak üzere programlama dillerinde temel veri yapılarıdır. Aynı türdeki birden çok değeri tek bir değişkende depolamak için uygun bir yol sağlarlar. Diziler ilgili verileri birlikte gruplandırmanıza olanak tanıyarak büyük miktarda veriyi düzenlemeyi ve işlemeyi kolaylaştırır. İşte dizilere ve birden çok değeri depolamadaki rollerine genel bir bakış:

* **Diziler ve Veri Depolama**

Diziler, aynı veri türündeki öğelerin bir koleksiyonunu depolamak için kullanılır.

Bir dizideki her öğeye, 0'dan başlayarak benzersiz bir dizin veya konum atanır.

Bir dizinin boyutu veya uzunluğu, tutabileceği maksimum eleman sayısını belirler.

Diziler, indekslerine dayalı olarak öğelere rasgele erişim sağlayarak verimli geri alma ve değiştirme olanağı sağlar.

* **Dizilerin Tanımlanması**

EGEDERT++'da bir dizi bildirmek için, öğelerin veri tipini, ardından dizi adını ve köşeli parantezleri [] belirtin.

İsteğe bağlı olarak, tutabileceği öğe sayısını belirterek dizinin boyutunu sağlayabilirsiniz.

* **Dizilerin Elemanlarına Ulaşmak**

Dizi elemanlarına köşeli parantez içindeki indeksleri kullanılarak erişilebilir.

Bir dizi öğesine erişmek veya onu değiştirmek için dizi adını ve ardından köşeli parantez içindeki dizini kullanın.

EGEDERT++’ta dizi tanımlamak için aşağıdaki syntaxı kullanmalısınız;

<veri türü>[] <dizi ismi>

<veri türü>, tamsayı ve metin gibi dizinin tutacağı öğelerin türünü temsil eder. <dizi ismi> ise dizi değişkenine verilen addır.

EGEDERT++’ta diziye eleman tanımlamak için ise aşağıdaki syntaxı kullanmalısınız;

<veri türü>[] <dizi ismi> = [<dizi elementleri>]

<elementler> dizi öğelerine atanacak ilk değerleri temsil eder.

EGEDERT++’ta dizi elemanlarına erişmek için ise aşağıdaki syntaxı kullanmalısınız;

<dizi ismi> [<index>]

<index>, elemanın dizideki konumunu temsil eder.

Diziler, birden çok değeri yapılandırılmış bir şekilde depolamak ve işlemek için güçlü bir mekanizmadır. EGEDERT++ dilinde dizilerin nasıl çalıştığını anlayarak veri depolama işlemlerinde kendinizi geliştirebilirsiniz!

**BÖLÜM 5: EGEDERT++ Dilinde Girdi – Çıktı**

**5.1: EGEDERT++ Dilinde Girdi**

EGEDERT++’ta kullanıcı girdisi almak çok basit! Girdiler programınızın girdi isteyerek ve girilen değerleri yakalayarak kullanıcıyla etkileşim kurmasını sağlar. EGEDERT++’ta kullanıcı girdisi almak için kullanılan yöntem şöyledir:

* **girdi() metodu**

girdi() metodu kullanıcıdan girdi almak için kullanılır. Kullanıcı klavyeden girdiyi yazıp Enter tuşuna basana kadar bekler. Kullanıcı girdiyi girdikten sonra enter tuşuna bastığında ise girdiyi belirtilen değişkene tanımlar. Örneğin;

ad = girdi(“Adınız nedir?”)

Bu kod bloğu ad değişkenine girdi sonucuna göre bir değer tanımlar. girdi satırı ile karşılaşıldığında kullanıcı aşağıdakine benzer gir çıktı görür;

Adınız nedir?

Kullanıcı cevabını yazdıktan sonra enter tuşuna bastığı zaman bilgi ad değişkenine tanımlanır.

Bu örneklerde program kullanıcıdan girdi( ) yöntemini kullanarak girdi istemektedir. Girilen değerler değişkende saklanır ve daha sonra gerektiğinde programda kullanılabilir. Örneğin;

yaz “Yoksa adınız “ + ad + “mi???”

EGEDERT++’ta girdi() komutunu kullanarak kullanıcı girdisine dinamik olarak yanıt veren ve bunu programın yürütme akışına dahil eden etkileşimli programlar oluşturabilirsiniz.

**5.2: EGEDERT++ Dilinde Çıktı**

EGEDERT++, kullanıcıya çıktıyı görüntülemek için çeşitli yöntemler sağlar. Bu yöntemler, programın yürütülmesi sırasında kullanıcıya bilgileri, sonuçları ve mesajları iletmenizi sağlar. EGEDERT++'da çıktıyı görüntülemek için yaygın olarak kullanılan yöntemlere genel bir bakış yapalım:

* **yaz() metodu**

yaz() metodunu daha önce incelemiştik. yaz() metodu değerleri veya mesajları standart çıktıya yazdırmak için kullanılır.

Değişmez metni, değişkenleri veya ifadeleri görüntüleyebilir.

+ operatörü kullanılarak birden fazla değer birleştirilebilir.

* **formatlıyaz() metodu**

formatlı yaz() metodu, değerler için yer tutucular belirleyerek çıktıyı biçimlendirmenizi sağlar.

Yer tutucular {} ile temsil edilir ve karşılık gelen değerlerle değiştirilebilir.

formatlı yaz() yöntemi, format dizgisini ve ardından girilecek değerleri alır. Örneğin;

ad = “Caner”

yaş = 21

formatlıyaz(“Adınız {}!!, yaşınız ise {}. Yaşlanmışsınız.”, ad, yaş)

{} olarak belirttiğimiz yer tutucuları metninizden sonra virgül kullanarak {} yerine gelmesini istediğiniz değişkeni yazarak değiştirebilirsiniz! Biçim dizesindeki yer tutucular {}, ad ve yas değişkenlerinin değerleriyle değiştirilir.

Bu çıktı yöntemlerini kullanarak, EGEDERT++'ta bilgileri ve sonuçları açık ve düzenli bir şekilde kullanıcıya etkili bir şekilde iletebilirsiniz.

**BÖLÜM 6: EGEDERT++ Dilinde Kütüphaneler**

**6.1: EGEDERT++ Dilinde Bulunan Kütüphaneler**

EGEDERT++, ortak programlama görevlerini basitleştirmek için bir dizi işlev ve yardımcı program sağlayan bir dizi standart kütüphanelerle birlikte gelir. Bu kütüphaneler, dilin işlevselliğini genişletir ve çeşitli işlemleri sıfırdan uygulamak zorunda kalmadan gerçekleştirmenize olanak tanır. EGEDERT++ içinde yaygın olarak kullanılan kitaplıklara bir giriş yapalım:

* **Matematik Kütüphanesi**

Matematik kütüphanesi, matematiksel işlemler ve hesaplamalar için fonksiyonlar sağlar.

Temel aritmetik işlemler, trigonometrik fonksiyonlar, logaritmik fonksiyonlar, rasgele sayı üretimi vb. için fonksiyonlar içerir.

Bu kütüphanedeki işlevler, karmaşık matematiksel hesaplamaları kolaylıkla yapmanızı sağlar. Matematik kütüphanesinde kullanılan bazı fonksiyonlar şunlardır;

* + karekök(x) 🡪 x sayısının karekökünü hesaplar
  + sin(x) 🡪 x açısının sinüs değerini hesaplar
  + cos(x) 🡪 x açısının cosinüs değerini hesaplar
* **Zaman Kütüphanesi**

Zaman kütüphanesi tarih veya saat ile etkin bir şekilde çalışabilmeniz için fonksiyonlar sunar. Geçerli tarih ve saati almak, saat farklarını hesaplamak, tarih ve saatleri biçimlendirmek gibi işlemleri zaman kütüphanesi ile kolayca yapabilirsiniz. Zaman kütüphanesi, zamanla ilgili işlevleri programlarınıza dahil etmenize ve tarih ve saatleri içeren işlemleri gerçekleştirmenize olanak tanır. Zaman kütüphanesinde kullanılan bazı fonksiyonlar şunlardır;

* şimdikiTarih() 🡪 Çalıştırıldığı andaki tarihi döndürür
* şimdikiSaat() 🡪 Çalıştırıldığı andaki saati döndürür
* zamanFarkı(saatBir, Saat2) 🡪 Verilen 2 saat arasındaki zaman farkını döndürür.

Bu kütüphaneler EGEDERT++’ta bulunan kütüphanelerden sadece bazıları. Her kütüphane, belirli işlemleri gerçekleştirmek için çok çeşitli işlevler sunarak programlama görevlerinizi daha verimli ve kullanışlı hale getirir. İşinize yarayan kitaplığı EGEDERT++’ta bulamadınız mı? Merak etmeyin onun için de bir çözümümüz var.

**6.2: EGEDERT++ Dilinde Harici Kütüphaneler**

EGEDERT++, programlarınızın işlevselliğini geliştirmek için harici kütüphaneler dahil etmenize olanak tanır. Dış kütüphaneler, standart EGEDERT++ dilinde bulunmayan ek işlevler ve yardımcı programlar sağlar. EGEDERT++'da harici kütüphanelerden işlevleri içe aktarmak ve kullanmak için sırasıyla aşağıdaki adımlar izleyebilirsiniz;

* **Hangi Harici Kütüphaneyi Kullanmak İstiyorsunuz?**

İhtiyaç duyduğunuz fonksiyonların bulunduğu harici kütüphaneye karar verdikten sonra kütüphanenin EGEDERT++ ile uyumlu olduğundan ve istediğiniz özelliklere sahip olduğundan emin olun.

* **Harici Kütüphaneyi Yükleyin**

Harici kütüphaneyi bilgisayarınıza indirin ve kurun. Kütüphanenin eğer varsa web sitesinden kurulum talimatlarına göz atmanız sizin için iyi olabilir. Kütüphanenin sisteminize doğru şekilde yüklendiğinden emin olun.

* **Harici Kütüphaneyi EGEDERT++’ta kullanın**

Kütüphanenin sorunsuz şekilde yüklendiğinden emin olduktan sonra EGEDERT++’ta kullan anahtar kelimesinden sonra çift tırnak içerisinde kütüphanenizin ismini yazmanız kütüphanenin sağladığı tüm faydalardan faydalanmanızı sağlayacaktır. Örneğin;

kullan “Kütüphanem”

Yukarıdaki kod satırı Kütüphanem isimli kütüphaneyi ve fonksiyonlarını EGEDERT++’ta kullanabilmenizi sağlar.

* **Harici Kütüphane Fonksiyonları**

Kütüphaneyi içe aktarıldıktan sonra, fonksiyonlarını programınızda kullanabilirsiniz.

Fonksiyonları kullanmak ve kullanımlarını anlamak için kullandığınız kütüphanenin sitesine göz atabilirsiniz.

Uygun sözdizimini ve parametreleri kullanarak fonksiyonları harici kütüphaneden çağırmaya hazırsınız!

Harici kütüphaneleri EGEDERT++ programlarınıza dahil ederek, ek fonksiyonlardan yararlanabilir, kodunuzu genişletebilir ve her şeyi sıfırdan uygulamak zorunda kalmadan daha karmaşık görevleri başarabilirsiniz.

**BÖLÜM 7: EGEDERT++ Dilinin Grammerı**

<program> ::= <deklarasyon-listesi>

<deklarasyon-listesi> ::= <deklarasyon> | <deklarasyon> <deklarasyon-listesi>

<deklarasyon> ::= <değişken-deklarasyonu> | <fonksiyon-deklarasyonu> | <kütüphane-ithali>

<değişken-deklarasyonu> ::= <tip> <tanımlayıcı>

| <tanımlayıcı> = <ifade>

| <tanımlayıcı> [ <dizi-boyutu> ]

| <tanımlayıcı> [ <dizi-boyutu> ] = <dizi-başlatma>

<fonksiyon-deklarasyonu>::= <tanımlayıcı> ( <parametre-listesi> ) başla <blok> bitir

<parametre-listesi> ::= <parametre> | <parametre> , <parametre-listesi>

<parametre> ::= <tanımlayıcı>

<blok> ::= { <ifade-listesi> }

<ifade-listesi> ::= <ifade> | <ifade> <ifade-listesi>

<ifade> ::= <atama>

| <değişken-deklarasyonu>

| <atama>

| <eğer-ifadesi>

| <sırasında-döngüsü>

| <için-döngüsü>

| <fonksiyon-çağrısı>

| <dönüş-ifadesi>

| <giriş-ifadesi>

| <çıktı-ifadesi>

<atama> ::= <tanımlayıcı> = <ifade>

<eğer-ifadesi> ::= eğer ( <ifade> ) <blok> bitir

| eğer ( <ifade> ) <blok> aksi\_takdirde <blok> bitir

<sırasında-döngüsü> ::= sırasında ( <ifade> ) <blok> bitir

<için-döngüsü> ::= i = <sayı> için i = <sayı> olana kadar i <operatör> <sayı> <blok> bitir

<fonksiyon-çağrısı> ::= <tanımlayıcı> ( <argüman-listesi> )

<argüman-listesi> ::= <ifade> | <ifade> , <argüman-listesi>

<dönüş-ifadesi> ::= döndür <ifade>

<giriş-ifadesi> ::= girdi <tanımlayıcı>

<çıktı-ifadesi> ::= çıktı <ifade>

<ifade> ::= <mantıksal-ifade>

<mantıksal-ifade> ::= <karşılaştırma-ifadesi> | <karşılaştırma-ifadesi> <mantıksal-operatör> <mantıksal-ifade>

<karşılaştırma-ifadesi> ::= <aritmetik-ifade> | <aritmetik-ifade> <karşılaştırma-operatör> <karşılaştırma-ifadesi>

<aritmetik-ifade> ::= <terim> | <terim> <aritmetik-operatör> <aritmetik-ifade>

<terim> ::= <faktör> | <faktör> <çarpma-operatörü> <terim>

<faktör> ::= <tanımlayıcı> | <veri-türü> | ( <ifade> ) | <fonksiyon-çağrısı> | <dizi-erişimi>

<veri-türü> ::= <tamsayı> | <ondalık> | <dize> | <mantıksal>

<dizi-erişimi> ::= <tanımlayıcı> [ <ifade> ]

<dizi-boyutu> ::= <tamsayı>

<toplama> ::= <terim> + <terim>

<çıkarma> ::= <terim> - <terim>

<çarpma> ::= <faktör> \* <faktör>

<bölme> ::= <faktör> / <faktör>

<terim> ::= <faktör> | <faktör> <toplama-operatörü> <terim>

<faktör> ::= <ifade> | <ifade> <çarpma-operatörü> <faktör>

<ifade> ::= <tanımlayıcı> | <sabit> | ( <ifade> )

<dizi-başlatma> ::= [ <ifade-listesi> ]

<ifade-listesi> ::= <ifade> | <ifade> , <ifade-listesi>

<tip> ::= tamsayı | ondalık | dize | mantıksal | <tanımlayıcı>

<tanımlayıcı> ::= <harf> <tanımlayıcı-harfleri>

<harf> ::= a | b | c | ... | z | A | B | C | ... | Z | \_

<tanımlayıcı-harfleri> ::= <harf> | <rakam> | <tanımlayıcı-harfleri>

<rakam> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<tamsayı> ::= <rakam>+

<ondalık> ::= <rakam>+ . <rakam>+

<dize> ::= " <dize-harfleri> "

<dize-harfleri> ::= <karakter> | <karakter> <dize-harfleri>

<karakter> ::= <harf> | <rakam> | <özel-karakter>

<özel-karakter> ::=

<mantıksal> ::= doğru | yanlış

<kütüphane-ithali> ::= import <kütüphane-adı>

<kütüphane-adı> ::= <tanımlayıcı>

EGEDERT++ hakkında tartışmak ve sorun çözmek için başka EGEDERT++ kullanıcıları ile iletişime geçmek için [www.egedertplusplus.com/forum](http://www.egedertplusplus.com/forum) sitemizi ziyaret etmeyi unutmayın.

Geri bildirimleriniz bizim için önemli. EGEDERT++ içerisinde herhangi bir hata ile karşılaşmanız durumunda geri bildirim yapmanız EGEDERT++’ı geliştirmemize yardımcı olur.

EGEDERT++ kullanım kılavuzumuzun burada sonuna gelmiş bulunuyoruz. Bu kısa yolculukta bize eşlik ettiğiniz için teşekkür ederiz. Umarız EGEDERT++ dünyaya yön verecek olan programlarınız için iyi bir platform olur.