Yazılım Geliştirmeye Giriş

Infotech Academy
MSCD-Yazılım Uzmanlığı Eğitimi
Engin Niyazi Ergül

Yazılım(Program) Nedir?

Bir bilgisayarın, telefonun, tabletin ya da benzer bir elektronik cihazın çalışmasını sağlamak, belirli görevleri yerine getirmek veya belirli bir işlevi gerçekleştirmek üzere tasarlanmış talimatları içeren bir program/uygulama olarak tanımlanabilir. Yazılım, bir cihazın donanımını kontrol etmek, verileri işlemek, kullanıcı arayüzleri oluşturmak, ağları yönetmek, oyunlar oluşturmak gibi çeşitli görevleri yerine getirmek amacıyla kullanılır.

Genel olarak iki kategoriye ayrılır:

1) Sistem Yazılımı: Windows, macOS, Linux vb.
2) Uygulama Yazılımı: Office, ERP yazılımları, VS Code vb.

Bu yazılımları hazırlayan, geliştiren kişiye yazılımcı/yazılım geliştirici gibi isimler verilir.

Sektörde İhtiyaç?

Yazılım alanındaki iş imkanlarını genellikle özel sektör, kamu sektörü veya serbest çalışan (freelancer) olarak üç ana kategoriye ayırabiliriz:

Özel Sektör:

Şirketler ve Kuruluşlar: Birçok şirket, yazılım geliştiricilere, test mühendislerine, veri bilimcilerine ve diğer yazılım uzmanlarına ihtiyaç duyar. Büyük teknoloji şirketleri, finans kuruluşları, perakende şirketleri, sağlık sektörü ve diğer sektörlerde yazılım pozisyonları mevcuttur.

Start-Up Şirketleri: Yeni kurulan şirketler, genellikle hızlı büyüme ve yenilikçilik amacıyla yazılım uzmanlarına ihtiyaç duyar. Start-up ortamları, çok yönlü ve girişimci bir yaklaşımı teşvik eder.

Sektörde İhtiyaç?

Kamu Sektörü:

Devlet Kurumları ve Belediyeler: Kamu sektörü, yazılım uzmanlarına geniş bir yelpazede iş imkanı sunar. Örneğin, vergi sistemleri, sağlık bilgi sistemleri, kamu güvenliği yazılımları gibi projelerde çalışabilirler.

Eğitim Kurumları: Okullar, üniversiteler ve eğitim kurumları, öğrenci yönetimi sistemleri, e-öğrenme platformları ve diğer yazılımlar için yazılım uzmanlarına ihtiyaç duyar.

Sektörde İhtiyaç?

Freelancer (Serbest Çalışan):

Bağımsız Geliştirici (Freelance Developer): Birçok yazılım uzmanı, kendi işlerini bulmak ve müşterilere hizmet vermek amacıyla serbest çalışan olarak tercih eder. Freelance yazılımcılar genellikle proje bazlı çalışırlar ve çeşitli müşterilere hizmet verirler.

Freelance Analist veya Danışman: Analiz, veri bilimi veya proje yönetimi gibi alanlarda uzmanlık sağlayan freelancerlar da mevcuttur. Müşterilere stratejik danışmanlık ya da proje bazlı hizmetler sunabilirler.

Freelance Test Mühendisi: Yazılım testi konusunda uzmanlaşan freelancerlar, çeşitli projelerde test süreçlerini yönetebilir ve hata ayıklama işlemlerine katkıda bulunabilirler.

Sektörde İhtiyaç: SONUÇ

İş arayanlar veya serbest çalışmak isteyen yazılım uzmanları genellikle çeşitli iş bulma platformlarını (Upwork, Freelancer, Toptal gibi), kariyer sitelerini(yenibiris.com, kariyer.net vb) sosyal medya gruplarını (özellikle linkedin) ve şirketlerin kendi web sitelerini kullanabilirler.

Ayrıca, ağ kurma ve kişisel bağlantılar da freelancerların iş bulma sürecinde önemli olabilir.

Her bir çalışma şekli kendi avantajlarına sahip olduğu gibi, bireyin kariyer hedeflerine ve tercihlerine bağlı olarak seçilebilir.

Ne tür uygulamalar geliştirilebilir?

1) **Web Uygulamaları:**
   Bir tarayıcı üzerinden çalıştırılabilen uygulamalar. (Parasut, Facebook, Instagram vb.)
2) **Mobil Uygulamalar:**
   Mobil cihazlar üzerinden çalıştırılabilen uygulamalar. (Örnek vermeye gerek yok sanırım :))
3) **Masaüstü Uygulamalar:**
   Bilgisayarların kendi sistemleri üzerinde çalıştırılabilen uygulamalar. (Office, Visual Studio vb.)

Nedir bu “ALGORİTMA”?

- Algoritma, belirli bir problemi çözmek veya belirli bir görevi gerçekleştirmek için adım adım talimatları içeren bir bilgisayar programlama kavramıdır.

- Bir algoritma, başlangıç noktasından hedefe ulaşana kadar izlenmesi gereken bir dizi adımdan oluşur.

- Temel olarak, algoritma bir işlemi gerçekleştirmek için tasarlanmış bir plan veya yönergeler kümesidir.

Algoritma’ya neden ihtiyaç duyarız?

1- Problemi Çözme Yeteneği:

Algoritmalar, karmaşık problemleri daha küçük ve yönetilebilir adımlara bölen ve bu adımları çözen sistemlerdir. Bir yazılım geliştirici, bir sorunu çözebilmek için etkili bir algoritmaya ihtiyaç duyar.

Algoritma’ya neden ihtiyaç duyarız?

2 - Performans ve Verimlilik:

İyi tasarlanmış algoritmalar, bir görevi gerçekleştirmek için daha verimli ve hızlı bir şekilde çalışabilir. Yazılım geliştiricileri, işlemlerin zaman karmaşıklığını ve kaynak kullanımını optimize etmek için algoritmaları dikkatlice seçer ve tasarlar.

Algoritma’ya neden ihtiyaç duyarız?

3 - Mantıklı ve Yapılandırılmış Programlama:

Algoritmalar, programlama dillerindeki temel yapı taşlarını oluşturur. Doğru bir algoritma kullanmak, yazılımın mantıklı ve yapılandırılmış bir şekilde çalışmasını sağlar. Bu, kodun daha anlaşılır, bakımı daha kolay ve genellikle hatasız olmasını sağlar.

Algoritma’ya neden ihtiyaç duyarız?

4 - Bellek ve Kaynak Yönetimi:

Algoritmalar, bellek kullanımını ve genel kaynak yönetimini etkileyebilir. Doğru algoritmalar kullanıldığında, yazılım daha az bellek tüketir ve kaynakları daha etkili bir şekilde yönetir.

Algoritma’ya neden ihtiyaç duyarız?

5 - Ölçeklenebilirlik:

İyi tasarlanmış algoritmalar, bir uygulamanın ölçeklenebilir olmasını sağlar. Özellikle büyük veri setleri veya yüksek trafikli uygulamalar için etkili algoritmalar kullanmak, performansın ve tepki sürelerinin korunmasına yardımcı olabilir.

Algoritma’ya neden ihtiyaç duyarız?

6 - Veri Yapıları ile Entegrasyon:

Algoritmalar, veri yapıları ile sıkça entegre edilir. Veri yapıları, bilgiyi düzenlemek ve depolamak için kullanılır, ancak algoritmalar bu veriler üzerinde çalışır. Doğru algoritma-veri yapısı kombinasyonu, belirli işlemleri daha etkili hale getirebilir.

Algoritma’ya neden ihtiyaç duyarız?

7 - Proje Yönetimi ve Planlama:

Algoritma, projenin planlanması ve yönetilmesi aşamasında da kritik bir rol oynar. İyi bir algoritma, geliştirme sürecini daha öngörülebilir ve yönetilebilir hale getirebilir.

Algoritma’nın Özellikleri?

Giriş: Her algoritmanın bir başlangıç noktası vardır.

Çıkış: Her algoritmanın bir bitiş noktası vardır.

Kesinlik: Her adımda yapılacak iş ve adımın amacı açık olmalıdır. İşler şansa bırakılmaz.

Basitlik: Problemin çözümünü, mümkün olan en az adım ile en kısa sürede gerçekleştirmelidir.

Sonluluk: Algoritma adımları sonlu sayıda ve bir noktada çıkış değerini üretmiş olarak sona ermelidir.

Psuedo Code da neyin nesi?

Pseudocode bir algoritmayı veya programın mantığını anlatmak için kullanılan bir dil veya teknik olarak kabul edilebilir. Pseudocode, bir programlama dilinin spesifik sözdizimine bağlı olmaksızın, genel anlamda kodun mantığını ifade etmek için kullanılır. Bu, programcının algoritmanın tasarımına odaklanmasına ve daha sonra bir programlama dilinde uygulama yapmasına olanak tanır.

Pseudocode, yazılım geliştirme sürecinin tasarım aşamasında veya bir algoritmanın planlanması sırasında kullanılabilir. Mantıksal adımları açıklamak ve bir algoritmanın işleyişini anlamak için kullanılan açık bir dil sağlar.

Pseudocode, farklı programlama dillerinde çalışan bir ekibin bir araya gelerek ortak bir anlayışa sahip olmasına da yardımcı olabilir.

Karşıdan karşıya geçme problemi pcode örneği

01 BAŞLA

02 Yolun soluna bak

03 Yolun sağına bak

04 Yolun soluna bak

05 Araç geliyorsa 02 adımına git

06 Yolu hızlı adımlarla geç

07 BİTİR

Çay demleme problemi pcode örneği

01 BAŞLA

02 Demliğe çay koy

03 Çaydanlığa su doldur

04 Çaydanlığı ocağa koy ve altını yak

05 Su kaynayana kadar bekle

06 Çayı demle ve çaydanlığa su ilave et

07 Tekrar ocağa koy

08 Su kaynayana ve çay dem alana kadar bekle

09 Servis yap

10 BİTİR

Kullanıcıdan iki sayıyı alıp toplayan Psuedocode örneği:

```
Algoritma: IkiSayiyiTopla

  Başla
    Sayi1 = Console'danOku()
    Sayi2 = Console'danOku()
    Toplam = Sayi1 + Sayi2
    Console'aYaz("Toplam: " + Toplam)
  Dur
```

Kullanıcının girdiği 3 sayının ortalamasını bulan pcode örneği:

```
Algoritma: UcSayiOrtalamasi

  Başla
    Sayi1 = KullanicidanAl()
    Sayi2 = KullanicidanAl()
    Sayi3 = KullanicidanAl()

    Ortalama = (Sayi1 + Sayi2 + Sayi3) / 3

    EkranaYaz("Üç sayının ortalaması: " + Ortalama)
  Dur
```

Girilen sayının çift sayı olup olmadığını bulan pcode örneği

```
Algoritma: CiftSayiKontrolu

  Başla
    Sayi = KullanicidanAl()

    Eğer Sayi % 2 == 0 ise
      EkranaYaz("Girilen sayı çifttir.")
    Değilse
      EkranaYaz("Girilen sayı tekttir.")
    SonEğer
  Dur
```

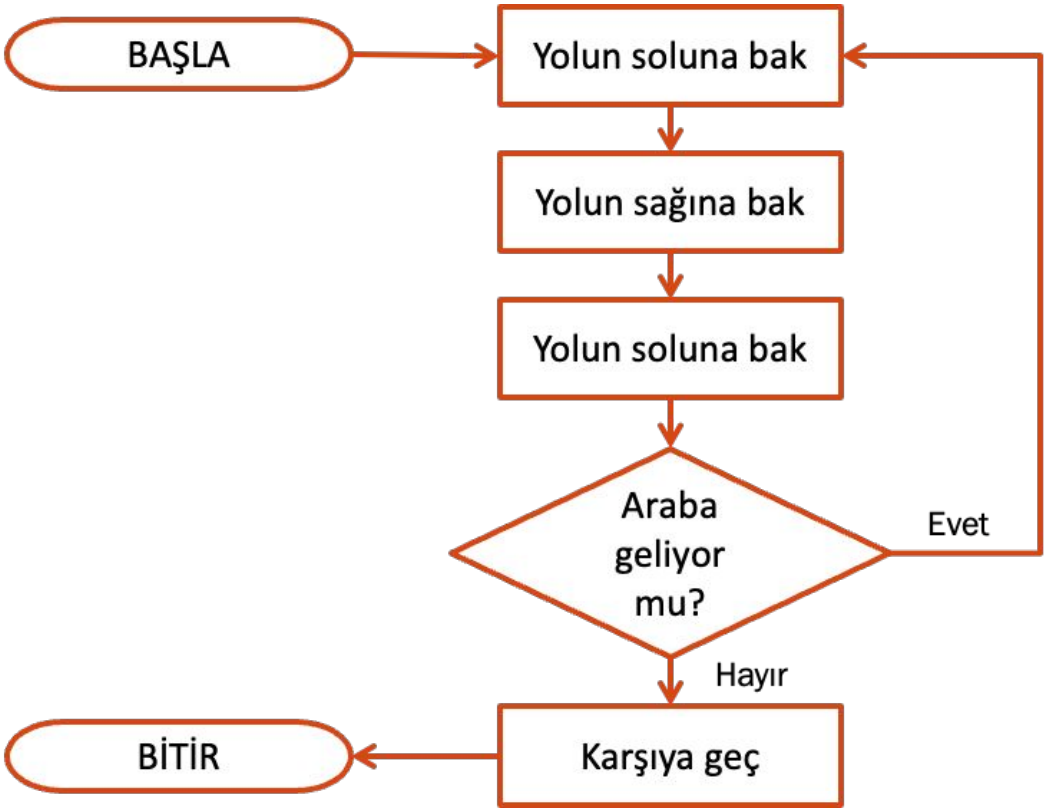
Girilen iki sayıdan büyük olanı bulan pcode örneği

```
Algoritma: BuyukSayiBul

  Başla
    Sayi1 = KullanicidanAl()
    Sayi2 = KullanicidanAl()

    Eğer Sayi1 > Sayi2 ise
      EkranaYaz("Büyük sayı: " + Sayi1)
    Değilse Eğer Sayi2 > Sayi1 ise
      EkranaYaz("Büyük sayı: " + Sayi2)
    Değilse
      EkranaYaz("Sayılar eşittir.")
    SonEğer
  Dur
```

Peki ya akış diyagramı???

Karşıdan karşıya geçme akış diyagramı

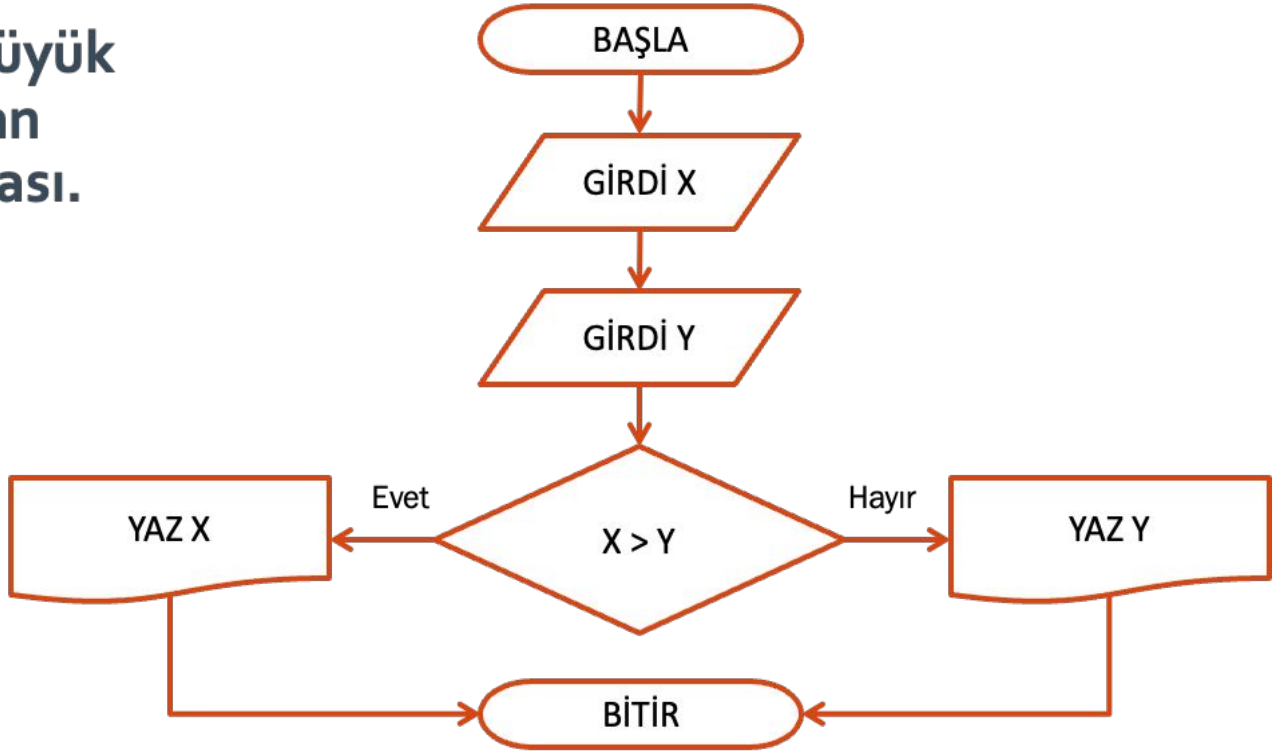
Yaya olarak karşıdan karşıya nasıl geçmeliyiz?



Çay demleme akış diyagramı

Büyük olanı bulma akış diyagramı

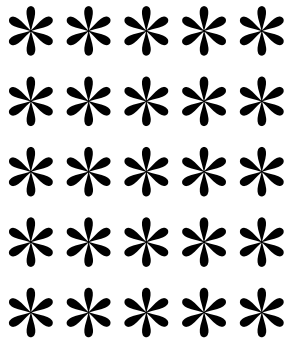
Girilen iki sayıdan büyük olanı ekrana yazdıran programın akış şeması.



SIRA SİZDE :)

Ekrana “*” sembolü ile 5’e 5 kare çizdiren programın algoritmasını psuedocode ve akış diyagramı olarak hazırlayınız.

HEDEFİNİZ:



ÇÖZÜM: PsuedoCode

A0 Başla

A1 satir = 0

A2 satir >= 5 ise A9'a git

A3 sutun = 0

A4 sutun >= 5 ise A7'ye git

A5 Ekrana * yaz

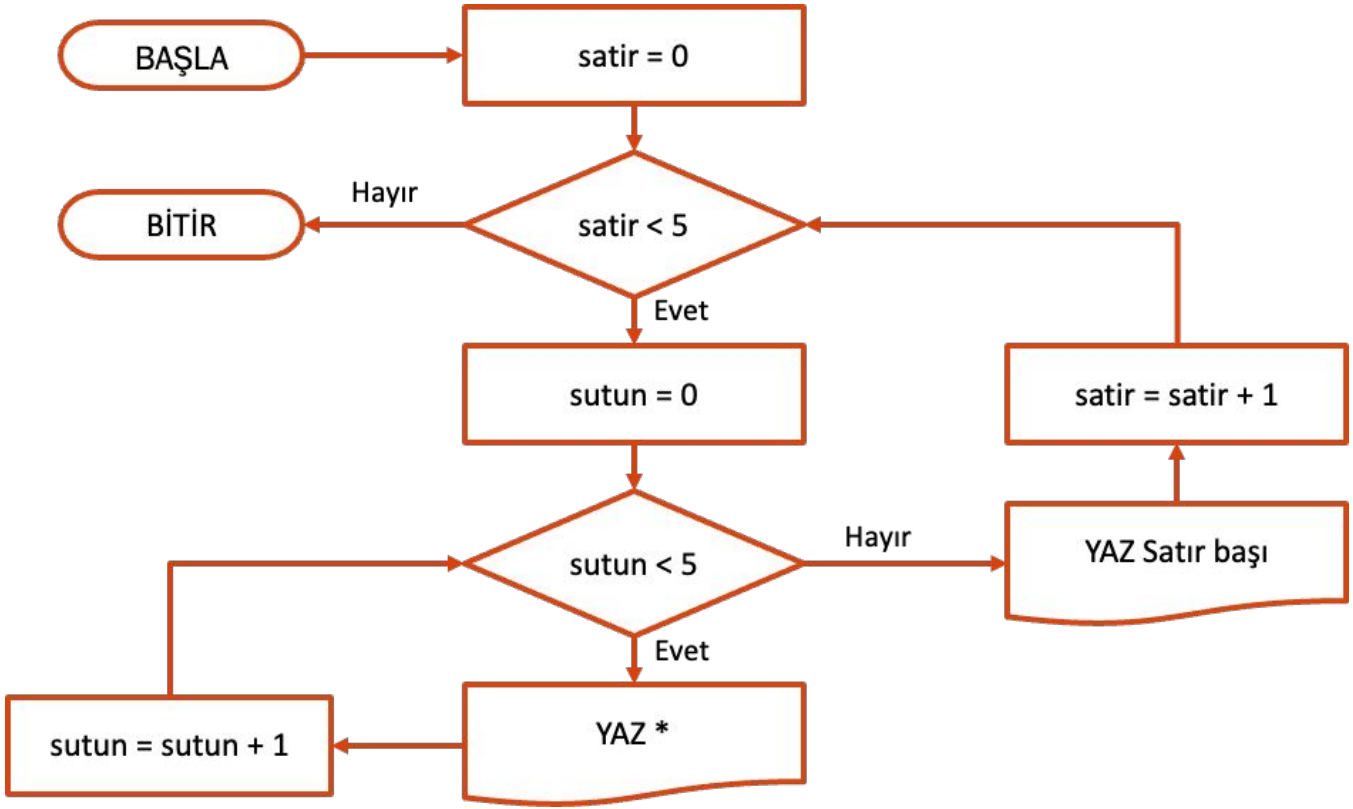
A6 sutun'u bir arttır A4'e git

A7 Bir satır alta geç

A8 satir'i bir arttır A2'ye git

A9 Dur

ÇÖZÜM: Akış Diyagramı



Geliştirme Ortamı: Kullanacağımız Araçlar

1) **Visual Studio 2022 Community Edition**
   Not: Visual Studio 2022 Community Edition ile birlikte, dotnet de kurulu gelir.
2) **Visual Studio Code**
   Not: Eğer sistemde daha önce yoksa mutlaka dotnet ayrıca kurulmalıdır.
3) **MS Sql Server, MS Sql Server Management Studio**
4) **Postman**
5) **Git, Github.com, Github Desktop**