**ALGORTİMA ÖDEVİ**

**1)Algoritma nedir? Algoritmanın gerekliliğini ve avantajlarını açıklayınız?**

Algoritma Nedir?

Algoritma, belirli bir problemi çözmek, belirli bir görevi yerine getirmek veya belirli bir sonuca ulaşmak için belirli adımları içeren, düzenli ve mantıklı bir sıra ile ifade edilen talimatların bir dizisidir. Algoritmalar, karmaşık problemleri daha küçük ve yönetilebilir alt problemlere böler, böylece sorunlar daha kolay ve etkili bir şekilde çözülebilir.

Algoritmalar birçok farklı alan ve uygulamada kullanılır. Örnek olarak, bilgisayar bilimleri, matematik, mühendislik, yapay zeka, veri analizi, otomasyon ve daha birçok alanda algoritmalar yaygın olarak kullanılır.

Algoritmanın Gerekliliği

Problem Çözme: Algoritmalar, karmaşık problemleri basit adımlara böler, bu da problemleri daha kolay ve hızlı bir şekilde çözmeyi mümkün kılar.

Verimlilik: İyi tasarlanmış algoritmalar, kaynakların (zaman, bellek, işlemci gücü) verimli bir şekilde kullanılmasını sağlar. Bu, işlemleri hızlandırabilir ve maliyetleri azaltabilir.

Tekrarlanabilirlik: Algoritmalar, aynı problemi veya görevi tekrar tekrar çözmek için kullanılabilir. Bu, güvenilir sonuçlar elde etmeyi kolaylaştırır.

Hata Azaltma: Algoritmalar, belirli bir sıra ve mantıkla çalıştıkları için insan hatasını minimize edebilir. Ayrıca hataları bulmak ve düzeltmek daha kolaydır.

Zaman ve Enerji Tasarrufu: İyi tasarlanmış algoritmalar, görevleri daha hızlı tamamlamayı sağlayabilir. Bu, iş süreçlerini hızlandırabilir ve enerji tasarrufu sağlayabilir.

Algoritmanın Avantajları

Etkin Problemlerin Çözümü: Algoritmalar, karmaşık sorunları daha basit ve anlaşılabilir adımlara böldüğü için etkin bir şekilde problemleri çözmeyi sağlar.

Daha Verimli Süreçler: İyi tasarlanmış algoritmalar, kaynakların verimli kullanılmasına yardımcı olur, bu da süreçleri hızlandırır ve kaynak israfını önler.

Hata Azaltma: Algoritmalar, önceden belirlenmiş adımların takip edilmesiyle hata yapma olasılığını azaltır.

Tekrarlanabilirlik: Algoritmalar, aynı işlemi tekrar tekrar uygulamak için kullanılabilir ve her seferinde aynı sonuçları üretir.

Otomasyon: Algoritmalar, rutin işleri otomatize etmek ve insan gücünü daha stratejik görevlere yönlendirmek için kullanılabilir.

Büyük Veri İşleme: Algoritmalar, büyük veri setlerini analiz etmek ve anlamlı bilgiler çıkarmak için kullanılır.

Yapay Zeka ve Makine Öğrenme: Bu alanlarda algoritmalar, karmaşık desenleri tanıma, tahminler yapma ve öğrenme yeteneklerini geliştirme işlevini yerine getirir.

Algoritmaların bu avantajları, birçok alanda kullanılmalarını ve problem çözme süreçlerini daha etkili hale getirmelerini sağlar. Bu nedenle algoritmalar, bilgisayar bilimleri, mühendislik, işletmeler, bilimsel araştırmalar ve daha pek çok alanda vazgeçilmez bir rol oynar.

**2)Program yazılmadan önce algoritması mı hazırlanmalı yoksa akış diyagramı mı çizilmelidir?**

Bir program yazmadan önce çalıştırmadan önce veya diyagram diyagramını çizmeden, yazılım geliştirme sisteminin önemli bir parçası ve hangi yöntemi kullanarak projenizin özelliklerine ve kişisel tercihlerinize bağlıdır. Her iki yöntem de faydalı olabilir, ancak projenizin karmaşıklığına ve özelliklerine göre tercih edilebilecek yöntemler değişebilir.

Algoritma Hazırlama:

Algoritma, bir programın çalışmasının adımlarını metin tabanlı bir şekilde gösterir. Bu, programın ne devam etmesi, adım adım anlaşılmasına yardımcı olur.

Algoritma yazmak, genel insan anlamanıza ve tasarım kararlarınızı vermenize yardımcı olabilir.

Daha küçük paketler veya daha basit problemler için parçaları ayırmak yeterli olabilir.

Algoritma, daha fazla teknik detay içerebilir ve kod yazmaya geçiş öncesinde bir üretim tasarımının geliştirilmesine yardımcı olur.

Akış Diyagramı Çizme:

Akış diyagramları, bir program mantığı ve dağıtılan grafiksel bir şekilde gösterilir. İşlem sırasını ve karar noktalarını görsel olarak temsil eder.

Büyük ve karmaşık yapılar için akış diyagramları daha anlaşılır ve yönetilebilir olabilir. Projenin daha net bir şekilde görmeye yardımcı olması mümkündür.

Özellikle işbirliği gerektiren büyük ekiplerle birlikte akış diyagramları, herkesin genel anlamda anlamasına yardımcı olabilir.

Projenizin büyüklüğü, karmaşıklığı ve ekibinizin yapısı gibi faktörler, hangi yaklaşım seçeneğini belirlemeyi belirler şemaları yazmak, hem de boru hattı diyagramını çizmek, daha iyi anlaşılmasına ve daha iyi tasarlanmasına yardımcı olur. İdeal olarak, bu iki yöntemi bir arada kullanabilirsiniz. Algoritma hazırlandıktan sonra akış diyagramı çizmek veya akış diyagramı üzerinde çalışanlarınızı göz önünde bulundurmak, programın çalıştırılması sırasında daha fazla netlik sağlar.

**3)Algoritma hazırlanırken dikkat edilmesi gereken hususları açıklayınız.**

1. Netlik ve Basitlik: Algoritma, mümkün olduğunca net ve basit olmalıdır. Karmaşıklık, hem yazma hem de anlama açısından zorluk yaratabilir.

2. Adım Adım İşlemler: Algoritma, her adımın net ve açık bir şekilde tanımlandığı ve her adımın işlevinin açıkça belirtildiği bir işlem sırasını izlemelidir.

3. Başlangıç ve Bitiş: Algoritmanın başlangıcı ve bitişi belirtilmelidir. Hangi koşullarda algoritma başlar ve hangi koşullarda sona erer?

4. Kontrol Yapıları: Kontrol yapıları, algoritmanın kararlarını ve tekrarlamalarını yönetir. İf-else ifadeleri, döngüler ve diğer kontrol yapıları, algoritmanın akışını kontrol eder.

5. Veri Girişi ve Çıkışı: Algoritmanın verileri nasıl alacağı ve sonuçları nasıl üreteceği net bir şekilde belirtilmelidir.

6. Mantıksal Doğruluk: Algoritmanın mantıklı olması gerekir, yani her adımın hedefinizi ve sorunuzu doğru bir şekilde ele alması gerekmektedir.

7. Modülerlik: Algoritmayı parçalara bölmek ve her parçayı ayrı ayrı ele almak daha büyük ve karmaşık algoritmaların daha kolay anlaşılmasını sağlar.

8. Döngülerin ve Yinelemelerin Sınırları: Eğer bir algoritma içinde döngüler veya yinelemeler kullanılıyorsa, bunların kaç kez çalışacağı veya hangi koşullar altında sona ereceği belirtilmelidir.

9. Performans ve Verimlilik: Algoritmanın işlemci ve hafıza kullanımı açısından verimli olması önemlidir. Fazla karmaşık veya gereksiz işlemlerden kaçınılmalıdır.

10. Hata Kontrolleri: Algoritma, hatalı girişlere veya beklenmeyen durumlara nasıl tepki vereceğini içermelidir. Bu, hata kontrol mekanizmalarının eklenmesini gerektirebilir.

11. Güvenlik: Eğer algoritma hassas verileri işliyorsa veya güvenlik endişeleri varsa, güvenlik önlemleri alınmalıdır.

12. Belgelendirme: Algoritmanın ne yaptığını ve nasıl kullanılması gerektiğini anlatan belgeler oluşturulmalıdır.

13. Test ve Hata Düzeltme: Algoritma oluşturulduktan sonra, test edilmesi ve herhangi bir hata veya eksikliğin giderilmesi önemlidir.

Algoritma, bir programı geliştirme sürecinin önemli bir başlangıç adımıdır ve programın doğru çalışmasını sağlamak için titizlikle oluşturulmalıdır. İyi bir algoritma, kodlama ve hata düzeltme süreçlerini daha verimli hale getirebilir.

**4)Değişken nedir? Programlarda neden değişkenlere ihtiyaç duyulmaktadır?**

Bilgisayar bilimleri ve programlama alanlarında, değişken (variable) bir değeri saklamak ve bu değeri daha sonra kullanmak için kullanılan bir isimdir. Değişkenler, programlama dillerinde veri depolamak için kullanılan temel yapı taşlarından biridir.

Eğer değişkenler olmasaydı programlama yaparken tüm olasılıkları tek tek yazmak zorunda kalırdık. Değişkenler sayesinde hem zaman kazanabiliriz hem de hata olasılığı daha düşük seviyelerde olur.

**5)Sayaçlar, nerelerde ve niçin kullanılmaktadır?**

Sayaçlar algoritmadaki olayları takip etmekte ve düzenli çalışmasında özellikle tekrar eden işlemlerin kontrolünde kullanılır bu yüzden programlamada kullanımı oldukça yaygındır.

Sayac = Sayac + 1 Bu işlemde Sayac değişkenine 1 eklenmekte ve oluşan sonuç yine kendisine yani Sayac değişkenine aktarılmaktadır.

**6)Aşağıdaki algoritmanın sonucu nedir?**

1. Başla

2. T=0

3. S=0

4. Eğer S>10 ise git 8

5. T=T+2\*S

6. S=S+2

7.Git 4

8.Yaz T

9.Dur

Sonuç = T = 0 dır.

**7)Aşağıdaki algoritmanın sonucunu hesaplayınız?**

1. Başla

2. F=1

3. S=20

4. Eğer S<1 ise git 9

5. S=S-3

6. F=F+S

7.F+2

8.Git 4

9.Yaz F

10.Dur

Cevap = F = 20 dır.

**8)Girilen üç sayıdan en büyüğünü bulan programın algoritmasını hazırlayınız.**

1. Başla

2. Sayıları a, b ve c olarak tanımla.

3. EnBüyük değişkenini a olarak başlat.

4. Eğer b > EnBüyük ise EnBüyük = b

5. Eğer c > EnBüyük ise EnBüyük = c

6. EnBüyük'ü ekrana yazdır.

7. Dur

**9)Girilen üç sayıyı küçükten büyüğe doğru sıralayan programın algoritmasını hazırlayınız.**

1. Başla

2. Sayıları A, B ve C olarak al

3. EnKucuk = A // Başlangıçta A'yı en küçük sayı olarak kabul ediyoruz

4. Orta = B

5. EnBuyuk = C

6. Eğer B < EnKucuk ise

7. Orta = EnKucuk

8. EnKucuk = B // Eğer B, şu ana kadar en küçük sayıdan daha küçükse, B'yi en küçük sayı olarak güncelle

9. Eğer C < EnKucuk ise

10. Orta = EnKucuk

11. EnKucuk = C // Eğer C, şu ana kadar en küçük sayıdan daha küçükse, C'yi en küçük sayı olarak güncelle

12. Eğer C > EnBuyuk ise

13. Orta = EnBuyuk

14. EnBuyuk = C // Eğer C, şu ana kadar en büyük sayıdan daha büyükse, C'yi en büyük sayı olarak güncelle

15. Eğer B > EnBuyuk ise

16. Orta = EnBuyuk

17. EnBuyuk = B // Eğer B, şu ana kadar en büyük sayıdan daha büyükse, B'yi en büyük sayı olarak güncelle

18. Sonuç = "Sıralama: " + EnKucuk + ", " + Orta + ", " + EnBuyuk

19. Yaz Sonuç

20. Dur

**10)1-99 arasındaki tek ve çift sayıların toplamları ile çarpımlarını ayrı ayrı hesaplayan programın algoritmasını hazırlayınız.**

1. Toplam değişkenlerini başlatın: tekToplam = 0 ve çiftToplam = 0 olarak başlatın.

2. Çarpım değişkenlerini başlatın: tekÇarpım = 1 ve çiftÇarpım = 1 olarak başlatın.

3. 1'den 99'a kadar olan sayılar için bir döngü oluşturun.

4. Her sayıyı kontrol edin:

a. Eğer sayı tekse:

- Tek toplamı güncelleyin: tekToplam += sayı

- Tek çarpımı güncelleyin: tekÇarpım \*= sayı

b. Eğer sayı çiftse:

- Çift toplamı güncelleyin: çiftToplam += sayı

- Çift çarpımı güncelleyin: çiftÇarpım \*= sayı

5. Döngü bittikten sonra, tekToplam, çiftToplam, tekÇarpım ve çiftÇarpım değişkenlerini yazdırın.

İşte bu algoritmanın Python programı:

python

tekToplam = 0

çiftToplam = 0

tekÇarpım = 1

çiftÇarpım = 1

for sayı in range(1, 100):

if sayı % 2 == 0: # Çift sayı

çiftToplam += sayı

çiftÇarpım \*= sayı

else: # Tek sayı

tekToplam += sayı

tekÇarpım \*= sayı

print("Tek sayıların toplamı:", tekToplam)

print("Çift sayıların toplamı:", çiftToplam)

print("Tek sayıların çarpımı:", tekÇarpım)

print("Çift sayıların çarpımı:", çiftÇarpım)

Bu program, 1'den 99'a kadar olan tek ve çift sayıların toplamını ve çarpımını hesaplayacaktır.