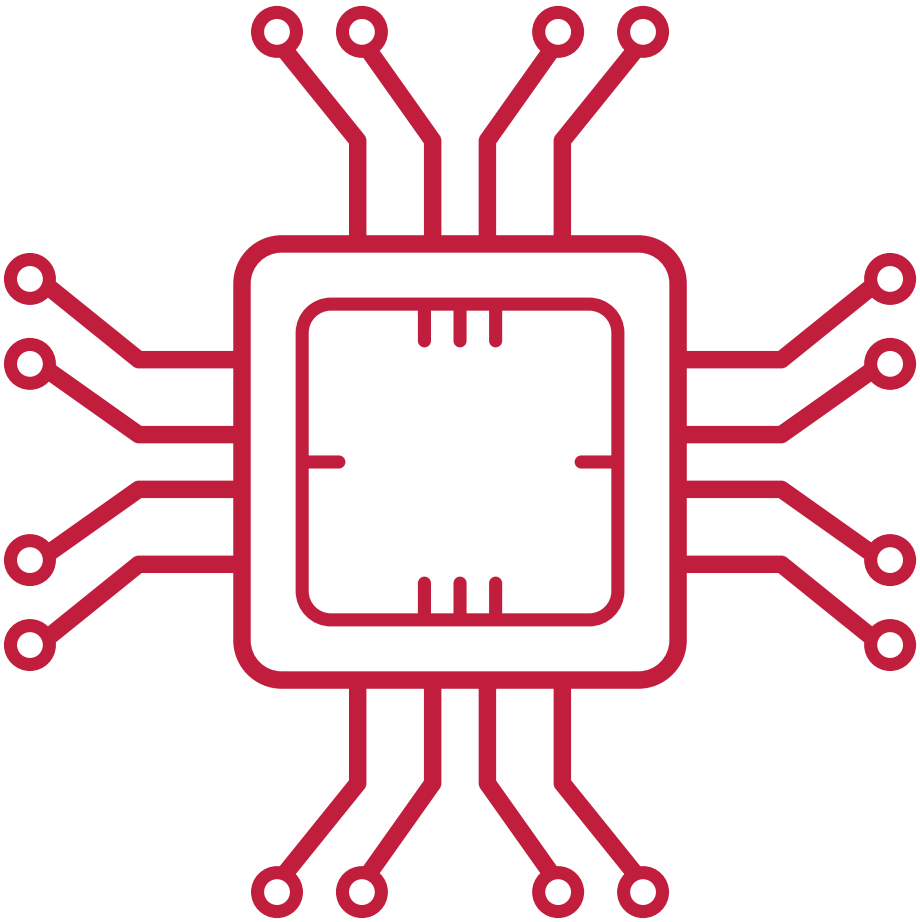
#MİLLİ TEKNOLOJİ HAMLESİ

DENEYAP TÜRKİYE

YAZILIM TEKNOLOJİLERİ

LİSE



TÜBİTAK

TÜRKİYE TEKNOLOJİ TAKIMI

TÜBİTAK Popüler Bilim Kitapları

YAZILIM TEKNOLOJILERI
LİSE

Dr. Öğr. Üyesi Caner ÖZCAN
Dr. Öğr. Üyesi Rafet DURGUT
Arş. Gör. Dr. Sevil ORHAN ÖZEN
Bilişim Teknolojileri Öğrt. Sercan ÖZEN

ISBN 978-605-312-443-6
Yayıncı Sertifika No: 47703

Genel Yayın Yönetmeni: Fatma BAŞAR
Mali Koordinatör: Adem POLAT
Telif İşleri Sorumlusu: Öznur KILIÇKAYA
Sayfa Düzeni: Duran AKCA

TÜBİTAK
Bilim ve Toplum Başkanlığı
Popüler Bilim Kitapları Genel Yayın Yönetmenliği
Tunus Caddesi No: 80 Kavaklıdere 06680 Ankara
Tel: (312) 298 95 21
e-posta: kitap@tubitak.gov.tr
esatis.tubitak.gov.tr

DENEYAP

Teknoloji Atölyeleri

YAZILIM TEKNOLOJİLERİ

LİSE

Dr. Öğr. Üyesi Caner ÖZCAN
Dr. Öğr. Üyesi Rafet DURGUT
Arş. Gör. Dr. Sevil ORHAN ÖZEN
Bilişim Teknolojileri Öğrt. Sercan ÖZEN

TÜBİTAK
POPÜLER BİLİM KİTAPLARI

İçindekiler

Sunuş

Eğitim dünyası günlük hayatta teknolojisiz ortamda yer almayan, okul hayatlarında da öğrenme için teknolojiden faydalanan pek çok çevrimiçi ürün ve teknolojileri kullanmanın yanı sıra, onları üretme becerisini de gösteren çok yönlü bir nesil ile karşı karşıyadır. Teknoloji çağında doğan dijital yerli olarak adlandırdığımız bu nesil hayatın her alanında hızla değişim yaşanan bir ortamda öğrenmeye adapte olmaya çalışmaktadır. Özellikle gençlerimizin adaptasyon sürecinde hazırlanan öğretim programlarının şekli ve uygulayıcıların kullandıkları öğrenme öğretme yöntemleri kilit noktadır.

Teknolojiyle birlikte değişen ve gelişen ihtiyaç ve pazar koşullarında gençlerimizi tüketim alışkanlıkları ile baş başa bırakmamak için 81 İlde 100 Deneyap Teknoloji Atölyesi kurulmasına yönelik olarak, T.C. Sanayi ve Teknoloji Bakanlığı, T.C. Gençlik ve Spor Bakanlığı, TÜBİTAK ve Türkiye Teknoloji Takımı Vakfı arasında önemli bir işbirliği tesis edilmiştir. Bu dört önemli kurumun katkılarıyla Deneyap Teknoloji Atölyeleri 2019 yılında hayata geçirilmiştir.

Ülkemizin kalkınması için teknoloji üretme yetkinliği yüksek genç bireyler yetiştirmeyi hedef alan Deneyap Teknoloji Atölyelerinin eğitim modeli kapsamında 36 ay süre ile ücretsiz olarak Tasarım ve Üretim, Robotik ve Kodlama, Elektronik Programlama ve Nesnelerin İnterneti, Yazılım Teknolojileri, İleri Robotik, Nanoteknoloji ve Malzeme Bilimi, Siber Güvenlik, Yapay Zeka, Mobil Uygulama, Enerji Teknolojileri, Havacılık ve Uzay Teknolojileri derslerine ilişkin eğitimler verilmektedir. Yazılım Teknolojileri, öğrencilere temel programlama mantığı, problem çözümüne yönelik algoritma tasarımı ve tasarlanan algoritmaları C++ programlama dilini kullanarak kodlama becerisi edinmelerini sağlayan bir eğitim içeriği sunmaktadır. Bu içerik doğrultusunda öğrencilerin gerçek hayat problemlerini algoritmaya dönüştürerek akış diyagramları oluşturabilmesi, C++ programlama dilini kullanarak akış diyagramının koda dönüştürülmesi, C++ fonksiyonların kullanımını sağlayabilmesi, nesneye yönelik programlama ile kodlama gerçekleştirebilmesi, dosyalama kavramını kullanarak proje tasarlayabilir hale gelmesi hedeflenmektedir. Eğitim öğrencilerin girişimcilik, yaratıcı düşünme, eleştirel düşünme, karmaşık problemleri çözme, etkili iletişim ve takım çalışması gibi becerileri kazanmalarına yönelik tasarlanmış olup, toplamda 12 hafta sürmektedir. Eğitim sonunda görev temelli bir yarışma düzenlenmektedir.

Okuyuculara sunulan bu kitap Deneyap Teknoloji Atölyeleri kapsamında lise öğrencilerine eğitim verecek eğitmenler için hazırlanan Yazılım Teknolojileri Dersi Öğretim Programını içermektedir. Ancak kitabın tamamı lise seviyesinde Yazılım Teknolojileri eğitimi vermek isteyen tüm kurum, kuruluş ya da eğitmenlerin kullanımına hizmet etmek için hazırlanmıştır.

Sizlere ve ülkemiz öğrencilerine faydalı olması dileğiyle!

Yazılım Teknolojileri Dersi Öğretim Planı Uygulama Kılavuzu

Bu kılavuzda yazılım teknolojileri dersinin öğretimi hakkında aşağıdaki başlıklara yer verilmektedir:

- Yazılım Teknolojileri dersinin hangi öğretim tasarım zemininde oluşturulduğu,
- Belirlenen öğretim tasarımının aşamalarının ne olduğu ve bu öğretim tasarımının aşamalarında eğitmenin süreçte nasıl rol alacağı,
- Eğitmenin yazılım teknolojileri dersinin öğretim süreçlerinde kullanabileceği öğretim yöntem ve teknikleri,
- Eğitmenin yazılım teknolojileri dersinin süreçlerinde kullanabileceği değerlendirme teknikleri,
- Eğitmenin yazılım teknolojilerinde kullanacağı programların tanıtımı,
- Eğitmenin yazılım teknolojilerinde kullanacağı teknolojik araçların tanıtımı.

Yazılım Teknolojileri Dersi Bilgi Paketi

Dersin Amacı

Bu dersin amacı, öğrencilere temel programlama mantığının öğretilmesi, problem çözümüne yönelik algoritmaların tasarlanması, tasarlanan algoritmaların C++ programlama dili kullanılarak kodlanmasıdır. Bu amaç doğrultusunda hedeflerimiz,

- Programlama temellerinin öğretilmesi,
- Verilen probleme yönelik uygun algoritma tasarımlarının geliştirilmesi,
- Akış diyagramından kodlamaya geçiş yapılması,
- C++ programlama dili kullanılarak problem çözümü,
- Nesne yönelimli programlama mantığının geliştirilmesi,
- C++ programlama dili ile proje geliştirilebilmesidir.

Dersin Çıktıları

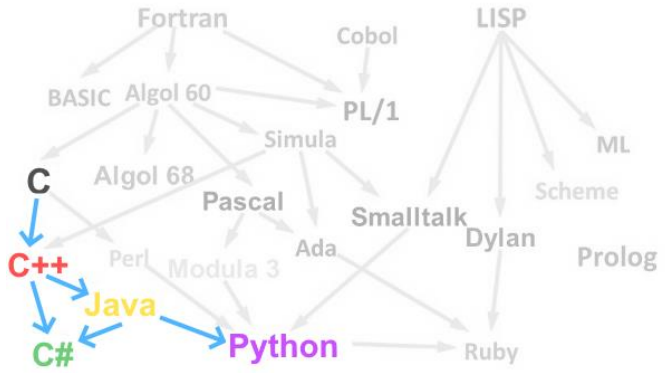
Bu dersi alan öğrenciler;

- Gerçek hayat problemlerini algoritmaya dönüştürebilir ve akış diyagramları oluşturabilir.
- C++ programlama dilini kullanarak akış diyagramını koda dönüştürebilir.
- C++ veri tipleri, programlama komutları ve fonksiyonların kullanımını sağlayabilir.

- Nesne yönelimli programlama ile kodlama gerçekleştirebilir.
- Dosyalama kavramlarını kullanarak proje tasarlayabilir.

Neden C++ Programlama Dili

C++, 1979 yılında Bjarne Stroustrup tarafından Bell Labs'da geliştirilen nesne yönelimli ve yüksek seviyeli, genel maksatlı programlama dilidir. C++ dili kullanılarak sistem yazılımları, özel yazılımlar, uygulamalar, sürücü yazılımları, kullanıcı taraflı yazılımlar ve gömülü firmware yazılımlar üretilmektedir. C++ dilinin orta seviyeli bir dil olmasından dolayı diğer yüksek seviyeli programlama dillerinden gerekli optimizasyon yapıldığında daha performanslı olduğu söylenebilir. C++ dilini öğrenir, mantığını anlarsak bu dilden etkilenerek oluşmuş programlama dillerini de temel seviyede öğrenmemiz kolay olur. Birçok üniversitede programlamaya giriş dersi olarak C++ eğitimi verilmektedir. Günümüzün en başarılı programcılarının çoğu C veya C ++ ile kod yazmayı öğrenmeye başlamıştır. Resim 1'de verilen programlama dillerine ait aile ağacı grafiğinde özellikle C#, Java ve Python dillerinin atasının C++ olduğunu görebilirsiniz.



Resim 1. *Programlama dillerine ait aile ağacı*

C++ programlama dilinin, farklı birçok uygulamada tercih edilen bir dil olmasını sağlayan iki temel özellik hız ve donanıma yakınlıktır. C++ programlama dili nesnelerin kullanımını sağlayan popüler bir dildir. Programcı için işleri kolaylaştıran yerleşik işlevlerle dolu bir kütüphane sunar. Java ve Python programlamlama dillerinden farklı olarak yorumlayıcı tabanlı değil derleyici tabanlı bir dildir ve bu nedenle bu dillerden nispeten daha hızlıdır. Basit içeriği ile yeni bir programlama dili öğrenmek isteyen programcılara hitap eder. Resim 2'de C++ programlamanın üstün özelliklerini görebilirsiniz.

Resim 2. *C++ programlama dilinin üstün özellikleri*

C++ önemli ve güçlü bir kullanım alanına sahiptir. Bu alanlardan bazılarına örnek verecek olursak;

- Gömülü Sistemler (Robotik Programlama) ve Elektronik Kartlar
- Masaüstü ve Hesaplama Uygulamaları
- Web Tarayıcı Oluşturma, Oyun Programlama, Derleyici Geliştirme
- Yeni Programlama Dili ve Yeni İşletim Sistemi Geliştirme

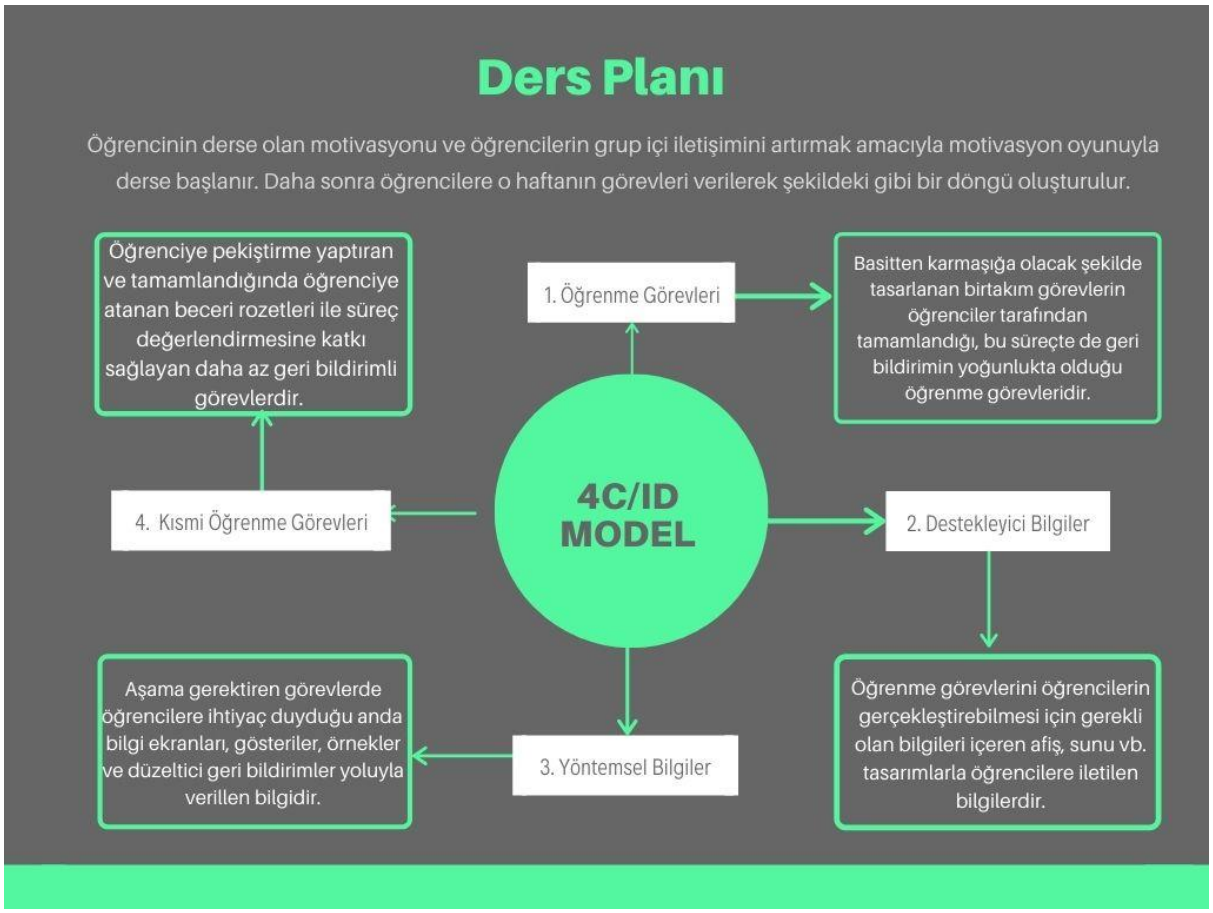
YouTube, Google, Amazon, Twitter, Facebook gibi uygulamaların yapımında C++ programlama dili de kullanılmıştır. Dünya çapında popülerliğini korumaktadır. Yüksek performanslı uygulamalar, oyunlar ve karmaşık araçlar yazmak için ve yazılan uygulamanın direkt olarak donanım ile haberleşmesini için C++ dili kullanmak gereklidir.

Ders Haftalık Planlaması

- Hafta 1: Programlamaya Giriş, Programlama Terimleri, Matematiksel İşlemler, Karşılaştırma İşlemleri, Mantıksal İşlemler, Sayı Sistemleri
- Hafta 2: Algoritma Tasarımı, Operatörler ve Kavramlar, Örnek Akış Şemaları, Kodlamaya Geçiş, Program Yazmanın Adımları (Hafta İçi Uygulama: C++ Programlama Dilinin Özellikleri, IDE (Derleyici) Kurulumu ve İlk Programlama)
- Hafta 3: C++ Tanımlamalar ve Operatörler, Değişken Tanımlama, Veri Tipleri, Sabitlerin Tanımlanması, Veri Tipi Dönüşümleri, Kaçış Dizgeleri, Operatörlerin Kullanımı, Bit İşlemler
- Hafta 4: Karar Verme Komutları (if-else, switch), If ifadesi, Switch İfadesi, Mantıksal Operatörler, Operatör Öncelikleri
- Hafta 5: Döngü Komutları (for, while, do-while), İç İçe Döngüler, Rastgele Sayılar
- Hafta 6: Diziler, Karakter Katarları, Örnek Problem Çözümleri
- Hafta 7: Fonksiyonlar, Fonksiyon Sözdizimi, Fonksiyon Çağırma
- Hafta 8: Nesne Yönelimli Programlama, Sınıf Tanımlama, Metod Tanımlama, Nesne Değişkenleri Oluşturma, Yapıcı Yöntemler
- Hafta 9: Nesne Yönelimli Programlama, Nesne Değişkenleri Oluşturma, Yapıcı ve Yıkıcı Metotlar
- Hafta 10: Nesne Yönelimli Programlama, Veri Soyutlama, Kapsülleme, Kalıtım, Polimorfizm, Aşırı Yükleme, Geçersiz Kılma
- Hafta 11: Kütüphane Kullanımı ve Dosyalama İşlemleri
- Hafta 12: Proje Yarışması

Dersin İşleniş Biçimi

Dersin işleniş biçimi Resim 3'te aşağıdaki gibi özetlenmiştir.



Resim 3. *Ders işleniş planı*

Yazılım teknolojileri dersinin öğretim tasarımı Dört Bileşenli Öğretim Tasarımı (4C/ID) modelinden temel alınarak planlanmıştır. Bu tasarıma göre;

Öncelikle her hafta için hazırlanmış olan ders materyal bölümündeki (bu dokümanın ilerleyen bölümlerinde yer almakta) görevler sırasıyla oluşturulan öğrenci gruplarına dağıtılır. Her görev için oluşturulacak öğrenci gruplarının sayısı hazırlanan haftalık ders planında belirtilmiştir.

Eğitmenin verdiği öğrenme görevinin yapılabilmesi için gerekli destekleyici bilgiler (teorik bilgiler) öğrencilere afiş, sunum vb. tasarımlar yoluyla eğitmen rehberinde belirtildiği gibi verilir. Birden fazla aşama gerektiren öğrenme görevlerinde öğrenciler bazı aşamaları geçemediği için görevi tamamlayamayabilir. Bu durumda eğitmen devreye girerek öğrenciye anlık ve hızlı geri bildirimlerle o görevi bitirmesi için işlemsel bilgi (yöntemsel bilgi) vermesi beklenir.

Kısmi öğrenme görevleri ise o haftanın konusunu öğrencinin pratikler yaparak otomatikleşme sağlaması için verilen görevlerdir. Bu aşamada yöntemsel bilgiler gittikçe azaltılır. Eğer öğrenci artık yöntemsel bilgiye gerek duymadan görevleri yerine getirebiliyor ise konuyu özümsediği anlaşılır.

Yazılım Teknolojileri Dersinde Kullanılacak Öğretim Tasarım Modeli

Yazılım teknolojileri dersi kapsamında belirlenen öğrenme hedeflerine erişmek için kullanılacak öğretim tasarımı modeli önem arz etmektedir. Öğretim tasarımı belirlenen hedeflere nasıl ulaşılacağını (hangi öğretim stratejileri ve araçlarının kullanıldığı) ve öğrencinin hedeflere ulaşıp ulaşmadığının kontrolü açısından (değerlendirmenin nasıl yapılabileceği) eğitimcilere yol göstermektedir (Mager, 1984). Öyle ki öğretim tasarımı, eğitim-öğretim ortamlarında uygulanacak her türlü faaliyetin belli bir plana göre uygulanmasıdır (İşman, 2015). Öğretim tasarımının genel amacı, öğrenmeyi daha verimli ve daha kolay hâle getirmektir.

Yazılım teknolojileri dersinin öğretim tasarımı **Dört Bileşenli Öğretim Tasarımı** (4C/ID) modelinden esinlenerek planlanmıştır. Bu modelin seçim nedeni 4C/ID modelinin, karmaşık öğrenme veya mesleki yeterliliklerin öğretilmesi için eğitim programlarının dört bileşen hâlinde tanımlanmasıdır. Burada karmaşık öğrenmeden kasıt, zor anlamında değil; öğrenme bağlamında edinilmiş yeni bilgi, beceri ve tutumların entegrasyonunu ve koordinasyonunu gerektiren bir takım bütüncül görev grubunun başarıyla uygulanmasıdır (Costa ve Miranda, 2019). Bu model, gerçek yaşam durumlarına yönelik öğrenme görevlerinin öğrenmeyi geliştiren unsur olduğunu düşünmektedir (van Merriënboer ve Kester, 2014).

Geleneksel olarak, bu tür derslerin tasarımı “teori” den başlar. Eğitmenler, bu teoriyi öğrencilere “iletmek” için bir dizi ders tasarlar. Buna ek olarak, tüm görevin tek ve küçük yönleriyle pratik yapmak için ödevler tasarlanır. Eğitmenler, bu teoriyi öğrencilere "iletmek" için bir dizi ders ve pekiştirme amaçlı ödevler tasarlar. Örneğin, bu ödevler döngüleri programlama dilinde kullanma veya veri tabanında bilgi oluşturma, okuma, güncelleme ve silme üzerine odaklanabilir. Eğitmenler daha sonra tüm konular ele alınana kadar küçük alıştırmalar ile dersi sürdürür. Bu şekilde geleneksel tasarım modellerinin yaklaşımını takip eder. Teorik bilginin sunulması ile başlayıp belirli uygulama öğelerine bağlanmasını izler. 4C/ID modeli bu yaklaşımı devirir. Profesyonel görevleri belirleyerek başlar, bunları öğrenme görevlerine dönüştürür ve ancak daha sonra hangi “teorinin” öğrencilerin bu öğrenme görevlerini tamamlamalarına yardımcı olması gerektiğini araştırır (Frere Jean vd. 2019).

Yazılım geliştirmek tipik bir karmaşık beceridir. Çünkü programlama dilleri, veri tabanları, geliştirme ortamları vb. hakkında kapsamlı bilgi gerektirir. Ayrıca geliştirilen yazılımı çalıştırmak, temiz ve doğru kodlar yazmak veya bir kullanıcı arayüzü tasarlamak gibi birden fazla beceri gerektirir. Buradan hareketle yazılım teknolojileri dersinde, karmaşık beceriler ve mesleki yeterliliklerin eğitimi üzerine odaklanan Dört Bileşenli Öğretim Tasarımı (4C/ID) modelinden faydalanılabileceğine karar verilmiştir (van Merriënboer, 2016). Bu kapsamda modelin Deneyap Teknoloji Atölyeleri Yazılım Teknolojileri dersinde öğrenme görevleri ile ilgili basamağı bire bir uygulamaya dökülmemiştir. Çünkü bu derste C++ programlama dili verilmektedir. C++ programlama dilinin öğretildiği bu derste, dilin zorluğu ve dikkat gerektirmesi hedef kitle bakımından göz önüne alınmalıdır. Bu nedenle modele ilişkin öğrenme görevleri, belli sayıda parçalı hâlde verilen içeriklerin birleşiminden oluşmaktadır. İçerikte öğrenme görevleri konulara göre kendi içinde karmaşık becerilere hitap eden haftalık etkinlikler şeklindedir. Bu dilin öğretiminde öğrencileri doğrudan karmaşık bütüncül görevlere dahil etmek onların motivasyonunu düşürmeye neden olabilir.

Yazılım Teknolojileri dersi kapsamında 4C/ID modelinde açıklanan karmaşık beceriler, çok sayıda farklı beceriyi aynı anda icra etme olarak düşünülebilir. Öğrencilerin sahip olabileceği

"Biz bunu neden öğreniyoruz?" sorusuna cevap oluşturabilmek ve onların öğrendiklerini bir yazılımcı olarak anlamakta güçlük yaşamalarına engel olmak için modelin basamakları kullanılırken, öğrenme görevleri temel ders konularına ayrılarak haftalık bütünlük içinde verilmeye çalışılmıştır. Örneğin öğrenciler Yazılım Teknolojileri dersinde bir derste problem çözme aşamalarını, diğer derste algoritma, bir başka derste akış şemaları bilgisini öğrenme görevleri şeklinde ayrı ayrı görmektedir. Bu şekilde sunulan öğrenme görevleri birkaç haftalık konu birikiminin ardından küçük bir proje ile sonlanır. Bu noktada bir yazılımcının mesleki hayatında karşılaşabilecekleri durumlara göre, bilgi ve becerilerini birleştirdiği bütüncül uygulamalar söz konusu olacaktır. Örneğin beş hafta boyunca verilen öğrenme görevleri, altıncı hafta tüm öğrenme görevlerini kapsayan karmaşık ancak daha bütüncül ve günlük hayatla ilişkili bir proje görevinde birleşmektedir. Bu anlamda projede kullanılan modelin aşamaları aşağıda açıklanmaktadır.

Dört Bileşenli Öğretim Tasarımı (4C / ID) Bileşenleri ve Modelde Eğitmenin Rolü

4C-ID modeline göre, karmaşık becerileri eğitmek için iyi tasarlanmış ortamlar birbiriyle ilişkili dört bileşenden oluşur:

(1) Öğrenme Görevleri

Öğrenme görevleri gerçek yaşam görevlerine dayanmalı, öğrencinin bu görevleri mantıksal akıl yürütme ve problem çözme becerilerinin bütünleştirmesini gerektirmelidir. Öğrenme görevleri projeler, görevler, vakalar, sorunlar veya diğer türden görevler olabilir. Öğrenme görevleri, öğretim modelinin ve öğrenci öğrenmesinin merkezinde yer alır. Öğrenciler, bir profesyonelin karşılaşabileceği en basit aşamaları içeren görevlerle işe başlar ve daha sonra bir öğrencinin üstesinden gelebilmesi gereken karmaşıklık düzeyindeki görevleri içeren görevlerle bitirir (van Merriënboer, Kirschner ve Kester, 2003). Bu görevler üzerinde çalışırken, eğitmen ve öğretim materyalleri öğrencilerin öğrenme görevlerini yerine getirmelerine yardımcı olmak için gerekli desteği ve rehberliği sağlar (Frere Jean vd. 2019).

Yazılım Teknolojileri dersinde öğrenme görevleri proje ya da problemler şeklinde öğrenciye parça parça iletilmektedir. Öğretim planının bel kemiğini oluşturur ve pratikte karşılaşılan gerçek yaşam durumlarına dayanır. Bu nedenle görevin günlük hayatla ilişkilendirilmesi önemli bir noktadır. Basitten karmaşığa sıralanan görev sınıfları mevcuttur. Bu görevler üzerinde çalışırken, eğitmen ve eğitim materyalleri öğrencilerin görevlerini tamamlamalarına yardımcı olmak için gerekli desteği ve rehberliği sağlar. Haftalık içerikte sunulan öğrenme görevleri gruplama, sıralama, analiz-sentez, tahmin etme gibi tekniklerle öğrencilerin eleştirel düşünmelerini sağlayacak düzeyde kendi içinde karmaşıklaştırılmış parçalı görevlerdir. Bu görev sınıflarının konulara göre gruplaşmasıyla, daha bütüncül ve karmaşık yapıda proje görevlerini oluşturur. Örneğin akış şemaları, karar verme, döngü komutları ve diziler ile ilgili basit ve haftalık öğrenme görevleri proje görevinde gruplaşır. Bu durumda öğrenciler proje görevi öncesi temel yapıyı da edinmiş olarak, tüm bu becerileri kapsayan daha bütüncül ve karmaşık bir görevle karşılaşmış olacaktır. Öğrenciler bu görevler üzerinde çalışırken, eğitmen ve eğitim materyalleri onların görevleri tamamlamasına yardımcı olmak için gerekli desteği ve rehberliği sağlar.

(2) Destekleyici bilgiler

Destekleyici bilgi, öğrencilere verilen görev veya problemlerin nasıl çözüleceğine yönelik yaklaşımları tanımlar. Destekleyici bilgi genellikle "teori" olarak adlandırılır ve öğrenme görevlerini tamamlamak için gerekli olan zihinsel modeller ve bilişsel stratejiler geliştirmek için bilgi içerir. Destekleyici bilgi, görevin problem çözme, akıl yürütme ve karar verme ile ilgilenen yinelenmeyen yönlerini hedefler. Dersler, çalıştaylar veya çalışma materyalleri şeklinde sunulabilir. Öğrencilerin öğrenme görevlerini yerine, önce ya da çalışırken öğrenim görmeleri için kullanılabilir (Frere Jean vd. 2019). Bu aşamada öğrenciye görevdeki performansına göre bilişsel geri bildirim verilir. Örnek olarak, eğitmen öğrenciye çözümünü bir akranının çözümüyle karşılaştırmasını önerebilir (Costa ve Miranda, 2019). Öğrenme görevinin tamamlanması aşamasında öğrencilere destek olan bilgiler kolay erişilebilir niteliktedir. Bu derste destekleyici bilgi için çeşitli bilgi kartları, tartışma panoları, tekrar ve pekiştirme amacıyla hazırlanan konu ya da kavramları özetleme teknikleri (afiş, kavram haritası, sunu vb.) kullanılmaktadır. Ayrıca eğitmenler öğrencilere görev tamamlama esnasında ihtiyaca dayalı rehberlik etmektedir.

(3) Yöntemsel bilgi

Öğrenme görevinin rutin kısımlarının nasıl çözüleceğini öğrenciye anlatır ve öğrenciye ihtiyacı olduğu anda anlık olarak sunulur. Öğrenci zaman içinde deneyip kazandıkça yöntemsel bilgi ortadan kalkar. Bu derste eğitmenler öğrencilere görev tamamlama esnasında ihtiyaca dayalı rehberlik etmektedir.

(4) Kısmi görev uygulaması

Öğrencilerin görev içinde bir rutin oluşturmak için gerekli becerileri otomatikleşinceye kadar tamamladıkları alıştırma görevleridir. Bu bileşen yalnızca öğrenme görevleri yeterli tekrarı öğrenciye sunmadığı zaman kullanılır. Ancak bu derste kısmi öğrenme görevleri süreçte değerlendirmeyi sağlamak amacıyla da kullanılmıştır. Her kısmi görevin doğru tamamlanmasının ardından öğrenciye görevi tanımlayan bir beceri rozeti verilir. Bu anlamda beceri rozetleri öğrencilere süreçte öğrenilen becerileri pratikleştirme imkânı veren yeni görevler sunmak için kullanılmaktadır.

Kısmi görevlerle bağlantılı bu rozetler ile öğrenciler öğrenme seviyelerine uygun şekilde gruplandırılarak eğitim sonunda projelerde birlikte çalışmaktadır. Bu projeler öğrencilerin edindikleri becerilerin tamamına hitap eden bütüncül görevlerdir. Bu şekilde hem akran öğretimine destek verilmekte hem de 4C/ID öğretim modelinin doğasına uygun bir bütünlük sağlanmaktadır. Ayrıca öğrencilerin kendi öğrenmesini yansıtma işlemi gerçekleştirmesi beklenir.

Eğitmenin rolü:

4C/ID modelini uyarlandığı Yazılım Teknolojileri dersinde her haftanın içeriği 20-25 dk. arasında değişen öğrenme görevlerinden oluşmaktadır. Bu görevler grup çalışmaları ile bilgi paylaşımı, eğitsel ve motivasyon oyunları ve bilgisayar üzerinde yapılacak öğrenme görevleri şeklindedir. Öğrenme görevleri başında ve aralarında 10 dk. süre ile motivasyon oyunları oynatılmaktadır. Motivasyon oyunlarının her hafta hangi aşamada kaç dakika süre içinde uygulanacağı her haftanın genel özetinde yer alan ders akışı bölümünde belirlenmiştir. Eğitmen motivasyon oyunu saatinde kendi tercih ettiği bir oyunu öğrencileriyle uygulayabilir. Ders

başında ve aralarında oynanacak bu motivasyon oyunlarının öğrencileri monotonluktan çıkarmak, grup etkinlikleri için kendi aralarında kaynaşma sağlamak, dikkatlerini toplamak, motivasyonu artırmak vb. amaçlarla öğrenme görevlerinde öğrenci katılımını destekleyeceği düşünülmüştür.

Her öğrenme görevinin kendine özgü öğrenme materyalleri bulunmaktadır. Bu materyaller, konu anlatımı ve öğrencilerin öğrenme görevlerini tamamlamalarını kolaylaştıracak görsel kartlar, tartışma kartları, çalışma kâğıtları, sunumlar ve oyun uygulamalarını içermektedir. Öğrenme görevlerine ilişkin tüm önemli bilgi ve materyaller eğitmenler için hazırlanan eğitmen rehberi bölümünde toplanmıştır. Eğitmenin hafta başında işlenecek öğrenme görevlerini inceleyerek derse hazırlık yapması ve gerekli materyalleri hazırlayarak öğrenme ortamını düzenlemesi beklenmektedir.

Kitap içeriği hafta hafta bölümlere ayrılmış ve uygulama yönergesi şeklinde hazırlanmıştır. Her haftanın başında ders akışını açıklayan genel özet bölümü yer almaktadır. Bu bölümde:

- Kazanımlar: Haftalık bir dersin 4 saat boyunca öğrencilere aktarılması gereken kazanımlarını içerir.
- Amaç: Haftaya özgü konuların temel amacını içerir.
- Önerilen Ders Akışı: 4 saatlik süre boyunca izlenecek öğrenme adımlarını içerir.

Genel özet bölümünün hemen altında önerilen ders akışı, sırasıyla öğrenme görevinin adına göre açıklanmaktadır. Her görevin adı süre, kazanımlar, eğitim materyalleri, uygulama ve eğitmenin yararlanabileceği konu içeriği olmak üzere alt başlıklardan oluşmaktadır. Eğitmen haftanın konusunu dersten önce eğitmen rehberine uygun şekilde aktarmak için bu adımları çok iyi incelemeli ve hazırlık yapmalıdır.

Model kapsamında öğrenciler öğrenme görevlerinde birtakım uygulamalar gerçekleştirirken eğitmen öğrencilere ihtiyaç duyduğunda ipucu, eleştirel sorular yönelterek geri bildirimlerde bulunmalıdır. Burada eğitmen görev için yeterli süre var ise, görevin yanıtlarını öğrencilere doğrudan vermek yerine onları biraz daha düşünmeye teşvik edecek geri bildirimler iletmelidir. Örneğin bunlar aşağıdaki gibi olabilir:

- *Bu konuya bir de şu açıdan bakmayı deneyin!*
- *Grup arkadaşlarınızla bunu biraz daha tartışmanızı istiyorum. Yeterli süreyi aldığınızda birlikte keşfedebileceğinize eminim.*
- *Cevabının doğru olduğunu söyleyemem ama yaklaştığını söyleyebilirim. Burada .... üzerine biraz daha odaklanmanı istiyorum.*

Eğitmen geri bildirimleri yerine öğrencilerin doğru yanıtları keşfedebilmeleri için akran geri bildirimlerinden de yararlanılabilir. Bunun için derste bilinçli olarak doğru yanıtlara ulaşan öğrencilerin iyi gözlemlenmesi önemlidir. Bu öğrencileri diğerleri ile eşleştirerek birbirlerine görevi açıklamaları istenebilir. Bu açıklamalar yapılırken eğitmen yakından süreci izlemeli ve yanlış öğrenmelerin önüne geçmeli, öğrenme boşluklarını doldurmak için tekrar anlık geri bildirimlerde bulunmalıdır. Bu anlamda eğitmen sınıfta süreci yöneten, izleyen ve öğrencilerle birlikte öğrenmeyi kontrol eden bir göreve sahiptir. Kısaca sürece rehberlik eder. Unutulmamalıdır ki eğitmenin öğrenmeye rehberlik etmesi, bilgiyi doğrudan aktarmasından çok daha zor olacaktır.

Eğitmenin Yazılım Teknolojileri Dersinin Öğretim Süreçlerinde Kullanabileceği Öğretim Metotları ve Teknikleri

Eleştirel Düşünme Teknikleri

Öğrenme görevlerinin öğrenciye sunumunda eleştirel düşünme tekniklerinden yararlanılmaktadır. Bu tekniklerden bazıları örnekleriyle şu şekildedir:

- Sıralama: Algoritma ya da kod satırlarını düzene koyma ya da sıraya dizme.
- Gruplama: Ortak özelliklere sahip nesneleri ya da dizileri gruplama.
- Talimat Verme: İki sayıyı toplama ile ilgili kod yazarı akranına sözde kod yöntemiyle talimat verme.
- Tahmin Etme ve Çıkarımda Bulunma: Ekran çıktısı ya da kodun çözüm getirdiği problemi tahmin etme.
- Analiz ve Sentez: Diziler içinden eleman çıkarma, iki yazıyı birleştirme.

SCAMPER Tekniği

Öğrenme görevlerinin tasarımında kullanılan bir diğer teknik ise SCAMPER yönlendirilmiş beyin fırtınası tekniğidir (Çilci, 2019; İslim, 2011; Yağcı, 2012; Yiğitalp, 2014). Yaratıcı düşünme tekniği (Özyaprak, 2016) olarak da bilinen SCAMPER tekniği ile bir nesne ya da fikri farklı açılardan düşünmeyi sağlayacak sorular yöneltilir. Buna göre her bir harf, temelde farklı soru kalıplarına işaret etmektedir (Özyaprak, 2016). SCAMPER akrostişini oluşturan İngilizce kelimeler ve kullanım örnekleri şu şekildedir:

- S: Substitude (Yer değiştirme): Buradaki dizilerin yerlerine başka ne gelebilir?
- C: Combine (Birleştirme): Hangi kodlar birleştirilirse, problemi çözebiliriz? Ekran çıktısında hangi iki sayıyı bir araya getirip toplanmıştır?
- A: Adapt (Uyarlama): Ortamın sıcaklık değeri 10 derece azalsaydı, kodun ekran çıktısında tahmin edilen hava durumu nasıl olurdu?
- M: Modify, Minify, Magnify (Değiştirme, küçültme, büyütme): Bu kodu daha hızlı çalışır hâle getirmek için nasıl bir değişiklik yapabilirim? Nesne değişkeninin değerlerini daha büyük verirsem, ekran çıktısı nasıl olur?
- P: Put to other uses (Başka amaçlarla kullanma): “Bu değişkeni başka hangi amaçla kullanabilirim?” ya da “bir nesneyi polimorfizm yoluyla başka hangi amaçlarla kullanabilirim?”.
- E: Eliminate (Yok etme, çıkarma): “Bir sınıf tanımlama içerisinden method çıkarsanız, ekran çıktısı nasıl olur?” ya da “diziden eleman çıkarsanız, ne değişir?”
- R: Reverse, Rearrange (Tersine çevirme ya da yeniden düzenleme): Bir dizideki elemanların dizilimini tersine çevirirseniz, ekran çıktısı nasıl değişir?

İşbirlikli Öğrenme

Öğrenciler öğrenme görevlerini tamamlarken küçük gruplar hâlinde çalışmaktadır. Burada bahsi geçen işbirlikli öğrenme sıradan bir grup çalışması değildir. Bunun nedenleri “Her Küçük Grup Çalışması İşbirlikli Öğrenme Değildir” alt başlığı altında ayrıntılı olarak açıklanmaktadır. Grup çalışmalarının işbirlikli öğrenme yapan öğrencilerin hem kendilerini hem de arkadaşlarını kapasitelerinin sonuna kadar geliştirmeye çalışmalarıdır. Bu, tek tek her öğrencinin öğretilenleri tam olarak öğrenmesinden farklı bir durumdur. Grup çalışması sırasında

öğrenciler tek başlarına edinemeyecekleri, ancak başka biriyle etkileşerek kazanacakları öğrenme yaşantılarını deneyimler. Örneğin soru sorma, açıklama yapma, eleştirme, örnek verme gibi.

İşbirlikli yöntemin kullanıldığı bu grup çalışmaları bazen istasyon tekniği, bazen de ayrılıp birleşme (jigsaw) tekniği ile tamamlanır. İstasyon tekniği bütün sınıfın her aşamada (her istasyonda) çalışarak bir önceki grubun yaptıklarına katkı sağladığı, bir basamak ileri götürmeyi, yarım kalan işi tamamlamayı öğreten bir yöntemdir. İstasyonlar öğrencilerin eş zamanlı olarak çeşitli öğrenme aktivitelerini gerçekleştirebilecekleri merkezlerdir. Diğer bir işbirlikli teknik olan Jigsaw ile öğrenciler 5-7 kişilik takım oluştururlar. Akademik materyal (ünite) ya da konu gruplardaki öğrenci sayısınca bölümlere (konuya) ayrılır. Her takıma aynı ünite (konu) verilir ve takımlardaki üyelerden ünite parçalarından (konulardan) birini seçmeleri istenir. Her üye kendi konusunu okur. Daha sonra farklı takımlarda aynı konuyu alan üyeler, gruplarından ayrılarak uzmanlık gruplarında bir araya gelirler; konu üzerinde tartışırlar. Sonra kendi takımlarıyla geri birleşerek, takım arkadaşlarını, kendi konularıyla ilgili olarak bilgilendirirler.

Eğitmenin Yazılım Teknolojileri Dersinin Süreçlerinde Kullanabileceği Değerlendirme Teknikleri

Oyun Temelli Düşünme, günlük veya dönemsel yaşam deneyimlerini, yarışma, keşif, senaryolaştırma veya işbirliğine dayalı eylemlere dönüştürmeye dayalı felsefe-düşünme biçimidir. Bu derste de birtakım oyunlaştırma öğelerinden yararlanılmaktadır. Bunlar her ders sonunda öğrenciler tarafından tamamlanan kısmi öğrenme görevlerinden alınacak beceri rozetlerinin kullanımı ile sağlanacaktır. Kısmi öğrenme görevleri, öğrencilerin görev içinde bir rutin oluşturmak için gerekli becerileri otomatikleşinceye kadar tamamladıkları alıştırma görevleridir. Her haftanın son saatinde bu öğrenme görevleri öğrenciye yeterli tekrarı sunmak ve süreç becerilerini değerlendirmek amacıyla süreli olarak kullanılır. Süre kullanılmasının temelinde oyunlaştırma öğeleri ile motivasyonun sağlanması hedeflenmektedir. Ayrıca öğrenciler bu süre içinde kısmi öğrenme görevlerinden kendi seçimiyle istedikleri sayıda görevi tamamlama esnekliğine sahiptir. Diğer bir ifadeyle, öğrenciler haftanın sonunda 5 kısmi görev varsa bunlardan 3'ünü istediği sıralamada yapmak isteyebilir. Eğitmenlerin bu noktada onları tüm kısmi görevleri tamamlama konusunda teşvik etmesi beklenir. Tamamlanan her görevin ardından öğrenci bir beceri rozeti kazanmaktadır. Bu anlamda beceri rozetleri öğrencilere süreçte öğrenilen becerileri pratikleştirme imkânı veren yeni görevler sunmak için kullanılmaktadır. Tüm haftalarda kısmi görevlerin içeriğiyle uyumlu ve göreve tanımlı dört farklı rozet bulunmaktadır. Bunlar:

1. **Analizci Rozet:** Verilen problem için üretilen çözümlerin uygunluğunu kontrol eder ve varsa mantık hataların giderilmesini sağlar.

2. **Kodlayıcı Rozet**: Probleme uygun çözümlerin uygulamaya geçebilmesi için kodlanmasını sağlar.

3. **Tasarlayıcı Rozet**: Verilen probleme uygun çözümün nasıl olabileceği ile ilgili ön hazırlıkları yaparak gerekli algoritma ve akış diyagramlarının hazırlanmasını sağlar.

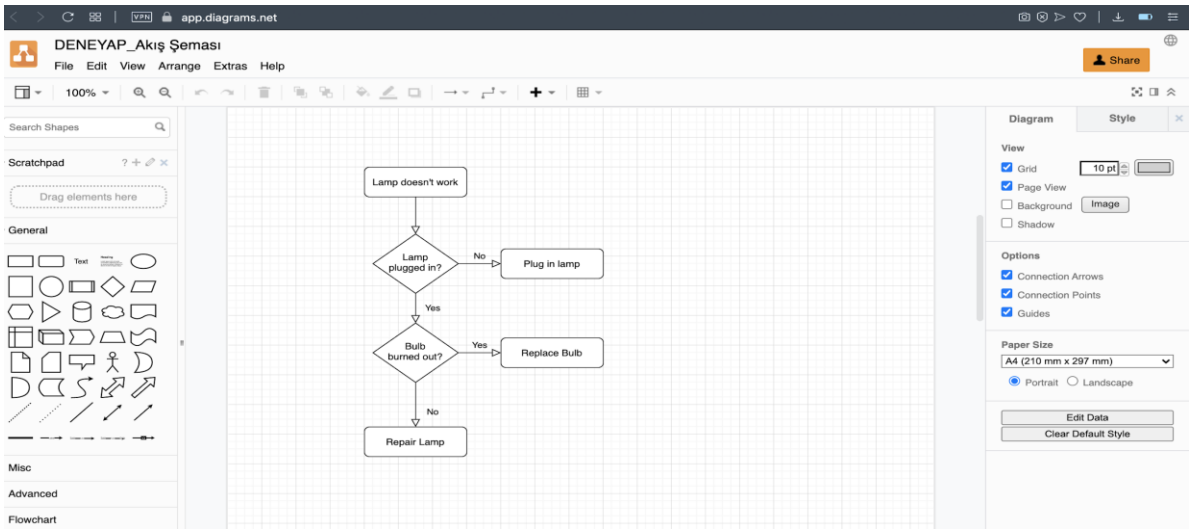
4. **Denetleyici Rozet**: Verilen problem için üretilen kodlamaların uygunluğunu kontrol eder ve varsa derleyici hatalarının giderilmesini sağlar.

Beceri rozetleri verilen kısmi öğrenme görevleri süreçte değerlendirmeyi sağlamak amacıyla kullanılmıştır. Bu şekilde öğrencilerin etkinliklere katılımını artırmak hedeflenmektedir. Ayrıca öğrenciler tüm hafta boyunca birden fazla analizci ya da kodlayıcı rozeti kazanabilir. Örneğin dönem sonunda edindikleri analizci rozet sayıları öğrencinin portfolyosunda ya da özgeçmişinde yer alacaktır. Bu noktada öğrenciler dönem boyunca edindikleri rozet tür ve sayılarına göre dönem sonunda yer alacakları proje görevlerinde ekip arkadaşlarını bulacaktır. Bu projeler öğrencilerin edindikleri becerilerin tamamına hitap eden bütüncül görevlerdir. Buradaki amaç ise oluşturulacak ekiplerin niteliğini artırmak adına birtakım yeterliliklere gelmiş öğrencilerin bir araya gelmesini sağlamak, ekip üyelerinin birbirlerini beceri anlamında tamamlamalarına imkân vermektir. Bu şekilde hem akran öğretimine destek verilmekte hem de 4C/ID öğretim modelinin doğasına uygun bir bütünlük sağlanmaktadır.

Eğitmenin Yazılım Teknolojilerinde Kullanacağı Programların Tanıtımı

Code::Blocks: Öğrencilerin, Yazılım Teknolojileri dersi kapsamında öğrenecekleri C++ programlama dilindeki programlarını yazmalarını ve derlemelerini kolay bir şekilde gerçekleştirebilmesi için Entegre Geliştirme Ortamı (Integrated Development Environment-IDE) ile derleyici özelliği olan ve her platform tarafından desteklenen Code::Blocks geliştirme ortamı tercih edilmiştir. Code::Blocks, geliştiriciler için işlevsel araçlarla tamamen yapılandırılabilir ve genişletilebilir açık kaynaklı ve ücretsiz bir IDE çözümüdür. Açık kaynaklı tasarımı sayesinde işlevlerinin büyük kısmı eklentilerle genişletilebilmektedir. Var olan gelişmiş araçlar sayesinde çok başarılı bir hata yakalama çerçevesi sağlamaktadır. Yazılımın derleyici eklentisi, yazılımcıların çok sayıda görevi birleştirmesini kolaylaştıran çalışma alanları sunmaktadır. Yazılım, platformlar arası bir çözümdür ve Mac, Windows ve Linux dahil olmak üzere farklı işletim sistemlerinde çalışır. C++ ile yazılmıştır ve çalışması için kısıtlayıcı kütüphaneler veya çevrilmiş bir dil gerektirmez. Code::Blocks kurulumu ders içeriğinde öğrencilere verilecek materyaller arasında mevcuttur.

Diyagram Çizim Uygulaması: Dersin önemli teknoloji ortamlarından biri ücretsiz çevrimiçi diyagram çizim uygulaması olan draw.io'dur. Bu uygulama Moodle sistemi içinde entegre şekilde kullanılmakta olup, öğrenciler tarafından ayrı bir kullanıcı kaydı ya da kurulum gerektirmemektedir. Öğrenciler uygulamaya farklı bir link üzerinden erişim sağlamaksızın, Moodle üzerinden aktif şekilde kullanabilmektedir. Bu uygulama ile öğrenciler Yazılım Teknolojileri dersindeki projeler için algoritmaların akış şemalarını bu dijital ortamda kolaylıkla çizebilecektir. Resim 4'te Draw.io uygulamasının arayüzü verilmektedir.



Resim 4. *Draw.io uygulamasının arayüzü*

Öğrenci çizimleri Google Drive, Dropbox, Onedrive, Github gibi dijital ortamlarla mevcut cihaz ya da bilgisayarda kaydedilebilir. Bu şekilde eğitmenin öğrenci çizimlerini takip etmesi daha kolay hâle gelecektir. Yazılım Teknolojileri dersinde öğrenci çizimlerinin kayıt ortamları için çoğunlukla Github kullanılmaktadır.

Eğitmenin Yazılım Teknolojilerinde Kullanacağı Diğer Teknolojik Araçların Tanıtımı

Öğrenme Yönetim Sistemi (ÖYS): Derste aktif şekilde kullanılacak olan temel ortamlardan biri öğrencilerin ilgili derse kullanıcı kaydı yapacakları bir Öğrenme Yönetim Sistemidir. Deneyap Teknoloji Atölyeleri tarafından Moodle açık kaynaklı popüler bir öğrenim yönetim sistemi kullanılacaktır. Yazılım Teknolojileri dersleri öğrenme görevleri şeklinde öğrenciler tarafından tamamlanan ya da üretilen ürünlerle ilerlemektedir. Bu nedenle öğrenme görevleri ÖYS ortamında modüler bir yapıda hafta hafta sunulmaktadır.

Yazılım Teknolojileri dersi için Moodle aşağıdaki amaçlara hizmet etmektedir:

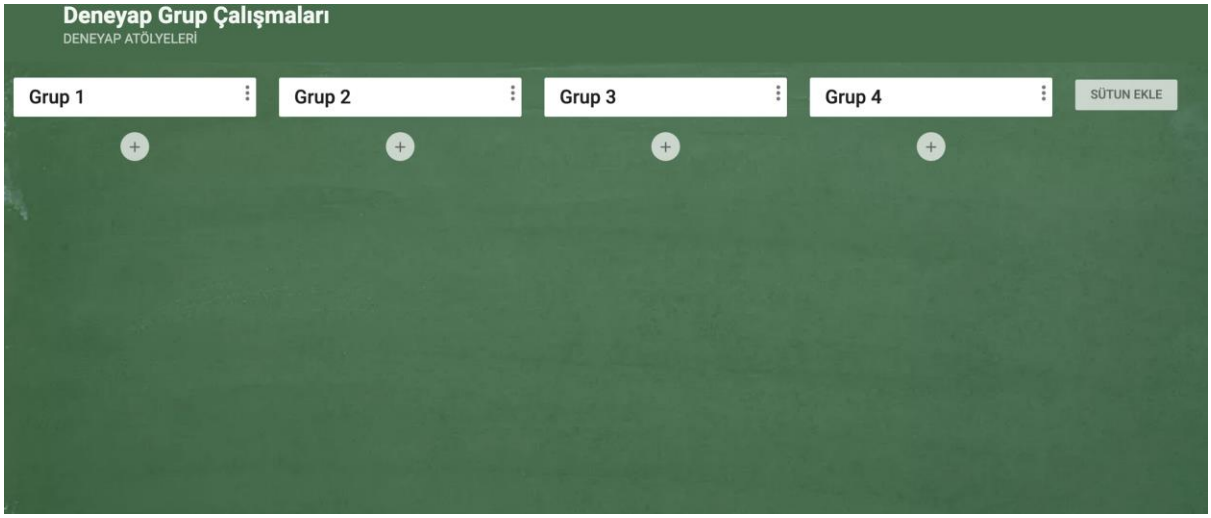
- Dersle ilgili genel duyuruların iletilmesi
- Öğrencilere sunulacak öğrenme materyallerinin paylaşılması
- Öğrenme görevi sırasında kullanılacak diğer teknolojik araçların iletilmesi
- Kısmi öğrenme görevleri gibi bireysel görevlilerin teslimi
- Grup olarak oluşturulacak ürün ya da tamamlanan görevlerin teslimi
- Eğitmenin öğrenci ürünlerini takip etmesi ve geri bildirim sağlaması
- Öğrencilerin kazandığı beceri rozetlerinin iletilmesi, kaydı ve raporlanması
- Öğrenci e-portfolyosunun raporlanması
- Öğrenci-eğitmen, öğrenci-öğrenci ve eğitmen-eğitmen etkileşiminin atölye dışında da sürdürülmesi
- Gerektiğinde BigBlueButton aracılığıyla senkron (canlı) ders ortamının oluşturulması (Oluşturulan canlı ders çalışma odaları destekli olup grup çalışmalarına imkân vermektedir.)
- Eğitmen eğitimleri sürecinde gerekli kaynakların, sunumların ve dokümanların eğitimcilerle paylaşılabilmesi
- Eğitimcilerin fikir alışverişi yapabilmesi için tartışma ortamlarının kurulabilmesi

GitHub: Derste aktif şekilde kullanılacak olan arşivleme profolyo oluşturma ortamlarından biri Github'tır. GitHub açık kaynaklı projeler tarafından tercih edilen ve yazılım geliştirme projeleri için kullanılan web tabanlı popüler bir depolama servisidir. GitHub ile dünya çapında herkes tarafından görüntülenebilen bir projenize, farklı ekip üyeleri ekleyerek takım çalışmaları düzenlenebilir. Ayrıca GitHub üzerinde paylaşılan kodlar ile kişisel gelişim sağlanabilir. Bu anlamda öğrencilerin GitHub ortamını kullanma deneyimini Yazılım Teknolojileri dersinde edinmesi önemli bir beceridir.

Yazılım Teknolojileri dersi için GitHub aşağıdaki amaçlara hizmet etmektedir:

- Her öğrencinin GitHub kullanıcı hesabı alması
- GitHub Branches ile küçük gruplar hâlinde küçük projeler geliştirilmesi
- GitHub Repository ile öğrencilerin kendi depolarını oluşturarak portfolio hazırlanması
- Öğrencilerin takıldıkları noktalarda yardımlaşması için GitHub görev yönetimi (Issues) ile düzenlemeler önerilmesi, yapılacaklar listesinin oluşturulması ve görev atamalarının yapılması
- Tartışma başlıkları açılması

Dijital Pano: Derste aktif şekilde kullanılacak olan temel ortamlardan bir diğeri dijital pano uygulamalarından kullanımı kolay ve popüler olan Padlet'tir. Bu uygulama Türkçe destekli pek çok özelliği ile 3 panoya kadar ücretsiz olan bir ortamdır. Bu uygulama boş bir duvarı içeriklerle doldurma imkânı veren dijital bir panodur. Öğrenciler giriş yapmaksızın padlet ortamında ücretsiz şekilde görsel, video ya da yazı ekleyebileceği bir panoyu birlikte doldurabilmektedir. Ayrıca padlet ortamının raf stili grup çalışmalarını desteklemektedir. Raf stili ile dijital pano üzerinde sütun şeklinde her grubun kendine ait bir alanı bulunur. Gruplar kendine ait sütun altında yer alan + işaretine tıklayarak ürünlerini paylaşabilir. Yazılım Teknolojileri dersinde bu stilden oldukça fazla yararlanılmaktadır. Resim 5'te örnek bir padlet raf stilinin görselini inceleyebilirsiniz.



Resim 5. *Padlet ortamı raf stili örneği*

Yazılım Teknolojileri dersinde padlet ortamı aşağıdaki amaçlara hizmet etmektedir:

- Gruplar ya da öğrenciler için ortak çalışma alanı kurma
- Tartışma ya da beyin fırtınası oluşturma
- Grup ürünlerini paylaşma
- Akranların yaptıklarını inceleme
- Atölye içi ve dışı işbirlikli çalışmayı destekleme
- Oluşan dijital panoyu çıktı hâlinde öğrencilerle paylaşma

Yazılım Teknolojileri dersinde padlet kullanımının avantajları aşağıda verilmektedir.

- Ders aktivitelerinde grup ya da bireysel öğrenci paylaşımlarının kaydını tutma
- Tüm öğrencilere birbirlerinin paylaşımlarını anlık olarak inceleme fırsatı sunma
- Akran öğretimini destekleme
- Ders kaynak ya da materyallerini dijitalleştirme, eğitim maliyetini düşürme
- Öğrencinin multimedya kaynaklarına ders sürecinde erişimini artırma
- Canlı ders yürütürken grup çalışmalarının ürünlerini dijital pano aracılığıyla takip etme

Kaynakça

Costa, J. M., & Miranda, G. L. (2019). Using Alice Software with 4C-ID Model: Effects in Programming Knowledge and Logical Reasoning. *Informatics in Education,* 18(1), 1-15.

Çilci, N., & Aydın, İ. (2019). *Scamper (Yönlendirilmiş Beyin Fırtınası) Tekniğinin 5 ve 6. Sınıf Öğrencilerinin Yaratıcı Yazıları Üzerindeki Etkisi* (Master's thesis). Ordu Üniversitesi, Ordu.

Demirel, Ö. (2005). *Eğitimde Program Geliştirme*. Ankara: Pegem A Yayınları.

Fox, J. (2004). *Rotate, differentiate, and motivate: "how a blend of learning stations and multiple intelligences theory can boost motivation and enhance learning in the middle school classroom* (Unpublished master's thesis). USA, Virginia: College of William & Mary.

Frerejean, J., van Merriënboer, J. J., Kirschner, P. A., Roex, A., Aertgeerts, B., & Marcellis, M. (2019). Designing instruction for complex learning: 4C/ID in higher education. *European Journal of Education, 54*(4), 513-524.

İslim, Ö. F. (2011). Scamper (Yönlendirilmiş Beyin Fırtınası Tekniği). *Uluslararası Bilgisayar ve Öğretim Teknolojileri Sempozyumu,* 22-24.

İşman, A. (2015). *Öğretim Teknolojileri ve Materyal Tasarımı.* Ankara: Pegem Akademi.

Mager, R.F. (1984). *Measuring Instructional Results*. Belmont, CA: David S. Lake Publishers.

Özyaprak, M. (2016). Yaratıcı Düşünme Eğitimi: SCAMPER Örneği. *Journal of Gifted Education and Creativity,* 3(1), 67-81.

Senemoğlu, N. (2007). *Gelişim öğrenme ve öğretim. (Düzenlenmiş Yeni Baskı).* Ankara: Gönül Yayıncılık.

Van Merriënboer, J. J., & Kester, L. (2014). *The four-component instructional design model: Multimedia principles in environments for complex learning*. Maastricht University.

Van Merriënboer, J. (2016). *How people learn.* The Wiley handbook of learning technology, 15-34.

Van Merriënboer, J. J., Kirschner, P. A., & Kester, L. (2003). Taking the load off a learner's mind: Instructional design for complex learning. *Educational psychologist, 38*(1), 5-13.

Yağcı, E. (2012). Yönlendirilmiş beyin fırtınası Tekniği: Scamper konusunda veli görüşleri üzerine bir çalışma. Hacettepe Üniversitesi Eğitim Fakültesi Dergisi, 2012(43), 485-494.

Yiğitalp, N. (2014). *Yönlendirilmiş beyin fırtınası (Scamper) tekniğine dayalı eğitimin beş yaş çocuklarının problem çözme becerilerine etkisinin incelenmesi* (Yüksek Lisans Tezi). Hacettepe Üniversitesi, Ankara.

Hafta 1. Programlamaya Giriş

Kazanımlar

- K1. Temel programlama terimlerini açıklar.
- K2. Bilgisayarın temel birimlerini açıklar.
- K3. Programlama türlerini ayırt eder.
- K4. Veri saklama birimlerini ayırt eder.
- K5. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.
- K6. Değişkenlere farklı türlerde değer ataması yapar.
- K7. İkili, sekizli, onlu ve onaltılı sayı sistemlerini kullanır.

Amaç

Bu haftanın amacı, programlamaya dünyasına giriş yapılarak öğrencilerin temel düzeyde bildikleri matematiksel, karşılaştırma ve mantıksal işleçleri (operatörleri) kullandırmak ve sayı sistemlerinin keşfedilmesidir.

Önerilen Ders Akışı

A. Giriş: Çizgi-Nokta Oyunu (15 dk.)
B. Öğrenme Görevleri
   B1. Nasıl Anlatırsın? (20 dk.)

   B2. Bilgisayarın Çalışması Neye Benzer? (20 dk.)

   - ❑ Ders Arası (5 dk.)
   - ❑ Motivasyon Oyunu (10 dk.)

   B3. Ben Farklıyım! (20 dk.)

   B4. Bilgisayar Verileri Nasıl Saklar? (20 dk.)

   - ❑ Ders Arası (5 dk.)

   B5. Beni Bul ve Değiştir (20 dk.)

   B6. Farklı Atama Türlerini Tanıyalım (20 dk.)

   - ❑ Ders Arası (5 dk.)

   B7. Sayı Sistemlerini Keşfedelim (20 dk.)

   - ❑ Motivasyon Oyunu (10 dk.)

C. Kısmi Öğrenme Görevleri (50 dk.)

A. Giriş: Çizgi-Nokta Oyunu

Süre: 15 dk.

Uygulama: Eğitmen derse girişte öğrenme sırasında grupları belirlemek için Çizgi-Nokta oyununu uygular. Bu oyunda öğrencilerden isimlerinin baş harfini alfabetik sıraya dizerek çizgi şeklinde durmaları istenir. Öğrenciler çizgiyi oluştururken hiç konuşmadan beden ya da işaret dili kullanacaktır. Bunun için eğitmen "Birbirinizin ismini konuşmadan sessizce tahmin edip çizgiyi oluşturalım" diyerek, kendisi de çizgiye katılır. Çizgi tamamlandığında herkes ismini söyler ve kontrol yapılır. Çizgiden sonra eğitmen "Şimdi de lütfen sizinle aynı mevsimi seven bir arkadaşınızı bulup ikili gruplar oluşturun. Yine bu işlemi de sessizce yapın. İşaret kullanabilirsiniz. Bu şekilde arkadaşınızla bir nokta oluşturmuş olacaksınız." diyerek nokta olmalarını ister. Oyun nokta aşamasında bitirilir. Nokta olanlar dersin devamında grup çalışmalarında ikili grupları oluşturacaktır.

Eğitmene Öneriler: Eğitmen birbirini daha önceden tanıyan bir öğrenci grubu ile çalışıyor ise, isimler ile ilgili çizgi ve nokta talimatlarını değiştirebilir. Örnek talimatlar: Doğdukları aylara göre çizgi olmak ve aynı göz rengine sahip olan biriyle nokta olmak vb. Süreye bağlı olarak sadece bir kez ya da daha fazla çizgi ve nokta aşamaları gerçekleştirebilir. Oyun nokta oldukları aşamada bitirilmelidir.

B. Öğrenme Görevleri

B1. Nasıl Anlatırsın?

Sure: 20 dk.

Kazanımlar: K1. Temel programlama terimlerini açıklar.

Materyaller: Bilgisayar ya da akıllı tahta

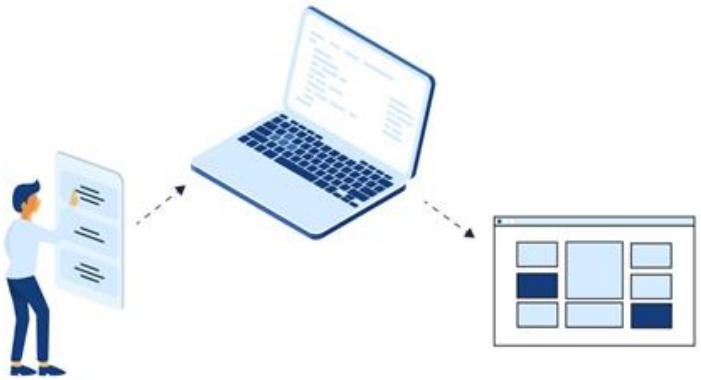
Uygulama: Öğrenciler dörderli gruplara ayrılır. Grupların belirlenmesi için öğrenciler arasındaki benzerliklerden yararlanılır. Örneğin aynı ayda doğanlar, adının baş harfi aynı olanlar vb. özellikler kullanılabilir. Her bir öğrenciye birer parça kâğıt verilir. Her öğrenci kâğıdına onun önemli bir özelliğini ya da o anki ruh hâlini yansıtan bir sıfat ile ismini yazar. Örneğin "Uykulu Özlem" ya da "Çalışkan Ahmet" gibi. Sınıfta oluşan gruplara "programlama" ve "programlama dili" kavramı verilir. Gruptaki her öğrenciden aldıkları kavram ile ilgili akıllarına gelen ilk kelimeyi kâğıtlarına yazmaları istenir. Daha sonra kâğıdı grup içindeki arkadaşları ile değiş tokuş eder ve akıllarına gelen ikinci bir kelime yazarlar. Bu şekilde öğrencinin kendi isminin olduğu kâğıt ona gelene kadar grup içinde yazma devam eder ve her bir gruptaki öğrenci sayısı kadar kavramlarla ilgili özellikler yer alacaktır. Bu işlemden sonra grup içindeki herkes kâğıdında tekrar eden ya da ortak olan kelimeleri grup kâğıdına yazar. Kâğıtların dönme süresi için eğitmen bir uyaran kullanabilir. Değiş ya da çan sesi gibi. Etkinlik bittikten sonra, Programlama ve kavramlarına ilişkin özellikler için tahta ikiye ayrılır. Öğrencilerden grup kâğıtları toplanır ve tahtada bu kelimeler sesli bir şekilde okunarak öğrencilerle tartışılır. Önemli noktalar tahtaya yazılarak, programlama ve programlama dili

kavramları beyin fırtınası yoluyla özetlenir. Sonuç olarak öğrencilerin aşağıdaki içeriğe ulaşmaları sağlanır.

Konu içeriği:

Programlama	Programlama Dili
Algoritma, Akış diyagramı	Bilgisayar ile konuşma
Yazılım, Kodlama	Bilgisayarla iletişim dili
Talimat verme	Kod dizisi
Bilgisayar kontrolü	Sözdizimi
Bilgisayar görevi	C, C++, Python, Java

Bu bölümdeki amacımız, programlamayla ilgili temel kavramların öğrencilere aktarılmasını sağlamaktır. Programlamanın tanımı, "çeşitli görevleri ya da işlemleri gerçekleştirmek için bilgisayara talimat verme" olarak verilebilir. Bilgisayarlar güçlü ve hızlı sistemler olmalarına rağmen insanlar gibi akıllı varlıklar olmadıkları için ne yapacaklarını bildiren bazı talimatlara ihtiyaç duyarlar. İşte programlama bu talimatları yazma işlemidir. Bunu yapmak için de bir programlama dili kullanılır. Programlama dilleri, belirli bir sözdizimsel biçimde yazılır ve bilgisayarın anlayabileceği okunabilir bir formata sahiptir. Hazırlanan programlar daha sonra bilgisayar tarafından yürütülerek istenilen görev ya da işlemlerin gerçekleştirilmesi sağlanır.



Resim 6. *Programlama süreci*

Tıpkı biz insanların birkaç dili (İngilizce, Almanca, Çince vb.) anlayabilmesi gibi, bilgisayarlarda da durum böyledir. Bilgisayarlar, programlama dili olarak adlandırılan belirli bir sözdizimsel biçimde yazılan talimatları anlar. Görevler ise "iki sayıyı toplama", "sayının karesini alma" gibi basit görevler olabileceği gibi bir dizi çoklu talimat içeren karmaşık görevler de olabilir. Özet olarak, programlama bilgisayarlara belirli bir görevi yerine getirmelerini söylemenin bir yoludur.

Yukarıda belirtildiği üzere bilgisayarlar, programlama dili olarak adlandırılan belirli bir sözdizimsel biçimde yazılan talimatları anlar. Programcının istediği bir görevi bilgisayar tarafından anlaşılması ve yürütülmesi için ifade etmesini sağlar. Popüler programlama dillerinden bazıları C, C++, Python, Java'dır. Bu ders kapsamında C++ programlama dili kullanılarak çalışmalar gerçekleştirilecektir. Bu dile ait bilgiler ilerleyen bölümlerde detaylı olarak verilecektir.

B2. Bilgisayarın Çalışması Neye Benzer?

Süre: 20 dk.

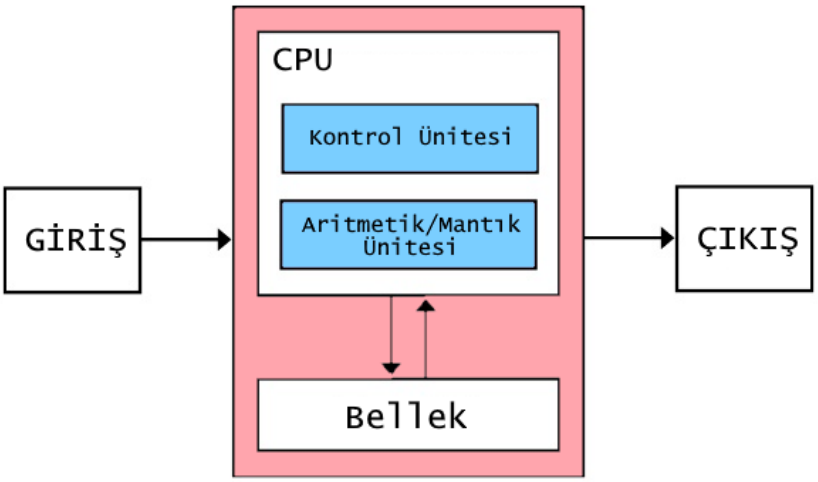
Kazanımlar: K2. Bilgisayarın temel birimlerini açıklar.

Materyaller: L1. B2. 1. Eşleştirme Kartları

Hazırlık: Eşleştirme kartlarından ders öncesi dört adet çıktı alınır. Dört grup için bu kartlar kesikli çizgilerden kesilerek karıştırılır.

Uygulama: Bu etkinlikte bilgisayarın temel bileşenlerini insan organları ile eşleştirme oyunu uygulanacaktır. Öğrencilerle beşerli gruplar hâlinde çalışılır. Eğitmen, öğrencilerden bilgisayarın temel birimlerinin olduğu kartları, insan organlarının bulunduğu kartlar ile eşleştirmelerini ister. Bu kartlar gruplara karışık şekilde dağıtılır. Eşleştirilen kartların aralarındaki ilişkiyi öğrencilerin kendi arasında tartışmalarını ister. Eğitmen son olarak sınıfta grupların kartlarının kontrolünü yapar ve beyin fırtınası ile konuyu aşağıdaki bilgilere erişecek şekilde özetler.

Konu içeriği: Bu bölümde öğrencilere temel bilgisayar mimarisinin öğretilmesi amaçlanmaktadır. Bilgisayarların birkaç bileşenden oluştuğu bilinmektedir. Bu bileşenlerden olan klavye, fare veya monitörü tanımlamak çok kolaydır. Bunların dışında CPU (merkezî işlem birimi) ve bellek gibi daha kompleks bileşenler vardır. CPU, bilgisayarın beyni olarak tüm kararların alındığı, veri işleyen ve yazılım komutlarını gerçekleştiren bölümdür. Genellikle RAM (rastgele erişimli bellek) olarak adlandırılan bellek ise verileri ve talimatları saklamak için kullanılan ve herhangi bir zamanda okunabilen veya değiştirilebilen bir tür veri deposudur.



Resim 7. *Bilgisayar Mimarisi*

Kontrol ünitesi CPU, içindeki veri akışını kontrol eder. Aritmetik/mantık ünitesi, veriler üzerinde aritmetik ve mantık işlemleri gerçekleştirir. Giriş bir veri yolu üzerinden CPU'ya gelirken, çıkış CPU'dan veri yolu üzerinden çıkar. Veriler, klavye girişi, dosya içeriği, web sunucusu istekleri veya bilgisayar tarafından kullanılan herhangi bir bilgi parçası gibi birçok şeyi ifade edebilir. Talimatlar ise bu iki sayıyı ekleyin, bu verileri buraya taşıyın, daha sonra bu talimata atlayın gibi CPU'ya bir sonraki adımda ne yapacağını belirten özel bir veri türüdür.

B3. Ben Farklıyım!

Süre: 20 dk.

Kazanımlar: K3. Programlama türlerini ayırt eder.

Materyaller: L1. B3.1. Bilgi kartları

Hazırlık: Eşleştirme kartlarından ders öncesi 10 adet çıktı alınır. Dört grup için bu kartlar kesikli çizgilerden kesilerek karıştırılır.

Uygulama: Eğitmen, öğrencileri dörderli gruplara ayırır. Her gruba programlama türlerinin şematik görseli olan bilgi kartlarından ikişer tane verilir. Öğrenciler grup içinde bu bilgi kartları arasındaki benzerlik ve farklılıkları tartışacaktır. Örneğin yapısal ve modüler programlama bilgi kartını alan grup, bu iki programlama türü arasındaki ortak ve farklı noktaları grup kâğıdına yazar. Bu şekilde her grup kendi içerisinde 10 dk. tartışır ve grup kâğıtlarını hazırlar. Son olarak grup kâğıtları eğitmen eşliğinde tüm sınıfça tartışılır. Eğitmen grup kâğıtları üzerinden konuyu aşağıdaki içeriğe benzer şekilde özetler.

Konu içeriği: Programlama, talimatları bir makineye veya bilgisayara göndermek için tasarlanmış bir gösterim sunar. Programlama temel olarak bir makinenin performansını kontrol etmek veya algoritmaları ifade etmek için kullanılır. Programlamada genel olarak üç farklı tür mevcuttur.

1. Yapısal Programlama

Yapısal programlama, programın tek bir yapı olarak oluşturulduğu ve genellikle programlamaya yeni başlayanların küçük ve basit kodlar yazarak başladığı programlama yaklaşımıdır. Talimatlar seri ve yapılandırılmış bir şekilde yürütülecektir. Yapısal programlama tekniğinde kod içerisinde var olan alt programlar tekrarlı olarak çağrılabilir. Böylece gereksiz kod tekrarı ortadan kalkmış olur. Yapısal bir program kolay anlaşılır ve düzenlenebilir bir programdır.

2. Modüler Programlama

Modüler programlama, programın ayrı ayrı modüller içinde gruplandırıldığı bir yazılım tasarım tekniğidir. Programın daha küçük parçalara ayrılarak büyük yazılım programlarının ve sistemlerinin oluşturulmasını sağlar. Her modül ana program içinde tanımlı değişkenleri kullanabilir ve kendi verisine de sahiptir.

3. Nesne Yönelimli Programlama

Nesne yönelimli programlama, nesnelere ve nesneler üzerindeki işlemlere dayanan programlama yapısıdır. Programcının kendi sınıfını ve prosedürlerini oluşturup bunun üzerinde işlemler yapar. Programlar, nesnelerin birbirlerine mesaj göndererek etkileşime geçmeleri üzerine kurgulanmıştır. Çok sayıdaki nesne önceden programcıya hazır bir şekilde sunulur. Nesne yönelimli programlama, gerçek dünya varlıklarını programlamada uygulamayı amaçlar.

B4. Bilgisayar Verileri Nasıl Saklar?

Süre: 20 dk.

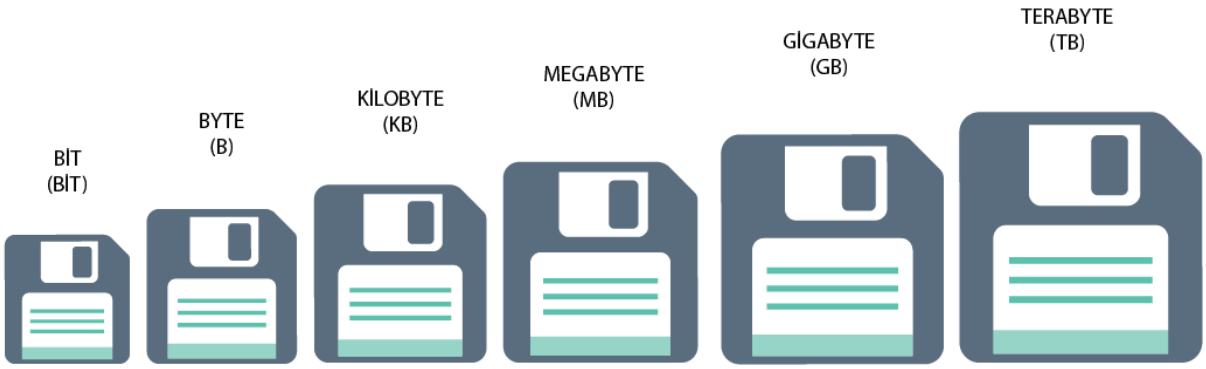
Kazanımlar: K4. Veri saklama birimlerini anlar.

Materyaller: L1. B4. 1. Sıralama kartları

L1. B4. 2. Çalışma yaprağı

Hazırlık: Sıralama Kartları, ön yüz ile arka yüzü iki tarafı eşleşecek şekilde arkalı önlü 4 adet çıkartılmalıdır. Çalışma yaprağı ise ÖYS üzerinden erişime açılır.

Uygulama: Eğitmen, öğrencilere bir soru yöneltilir. "Nasıl olur da sayılarla çalışan bilgisayar benim resimlerimi depolayabilir?" Eğitmen bu soruyu öğrencilerine sorarak aralarında tartışma yapmalarını sağlar. Beş dakikalık bir tartışmanın ardından öğrenciler 5'erli kişiler hâlinde 4 gruba ayrılarak onlara eğitmen tarafından sunulan karışık sıralama kartları verilir.

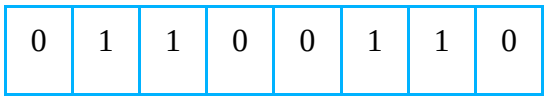


Resim 8. *Sıralama çizgisi*

Veri saklama birimleri ile ilgili kartların üzerine her bir hafıza birimi (bit, byte, mb, gb vb.) için hem görsel hem de açıklama olacak şekilde arkalı önlü kartlar oluşturulur. Öğrenciler 10'arlı gruplara ayrılarak, her bir grup üyesinin oluşturulacak kartlardan birini seçmesi istenir. Öğrenciler seçtikleri kartı grup arkadaşlarına göstermeyecektir. Eğitmen iki grup için farklı noktalarda iki sıralama çizgisi oluşturur. Sıralama çizgisi için tahta ya da grup masaları kullanılabilir. Öğrenciler ellerindeki kartların sıralama çizgisinde hangi konuma geleceği konusunda tahmin yürüterek yerleştirecektir. Çizgi üzerine yerleştirilen tüm kartlar kapalı konumdadır. Burada amaç grubun elindeki kartların küçükten büyüğe doğru sıralanmış bir çizgi şeklinde olmasıdır. Eğitmen öğrencilerden hazır olduklarında kartları açmalarını ve sıralamayı kontrol etmelerini ister. Eğitmen bu şekilde sıralama çizgisindeki yanlışların nasıl düzeltileceğini öğrencilere keşfettirmeye çalışır. Sıralama çizgisinin ardından eğitmen öğrencilere veri saklama çalışma kâğıdını dağıtır ya da ÖYS üzerinden çalışma kâğıdını öğrencilere ödev olarak gönderir. Etkinlik sonunda öğrencilerden bu çalışma kâğıdını inceleyip bireysel olarak doldurmaları istenir. Doğru yanıtlar sınıfta beyin fırtınası aracılığıyla tartışılır ve konu içeriği aşağıdaki gibi özetlenir.

Konu içeriği:

Bilgisayar sistemlerindeki bütün bilgiler ikili sistemde “0” ve “1” ile temsil edilerek saklanır. İkili sistemdeki her bir basamağa bit denir. Bu durumlar, elektrik akımının ortamda var olup olmamasına karşılık gelir. Eğer elektrik akımı varsa, anlık durum 1 değerini alır, yoksa 0 değerini alır. Bit, nicelik ifade edebilmek için yeterli bir birim olmadığı için en yaygın dijital veri depolama birimi olarak 8 bitten oluşan bayt kullanılır. Yani ikili sayı sistemi özünde sadece iki seçenek sunabildiği için, çalışma mekaniklerini bir sonraki aşamaya taşıyabilmek için dizgileri kullanırız. Örneğin sadece bir bit kullanmak yerine, sekiz tane biti bir araya getirip 1 byte oluşturabiliriz.



8 bit = 1 bayt

Bilgisayar sisteminde her bir karakter 8 bit’ten oluşur. Örneğin: A karakteri bilgisayar sisteminde “00100001” ikili sayısı ile ifade edilir. İşte bu sayının her basamağına 1 bit denir. Depolama bayt cinsinden ifade edildiği için tüm büyük birimlerin kısaltılmış adları kullanılır. Bilgisayar verileri metrik sistemde olduğu gibi bayt (B), kilobayt (KB), megabayt (MB), gigabayt (GB), terabayt (TB), petabayt (PB), exabyte (EB), zettabayt (ZB) ve yottabyte (YB) olarak ifade edilir ve 1 KB 1024 bayttır.

Tablo 1. *Birimler*

Birim	Kısaltma	Kapasite
Bit	b	1 ya da 0
Bayt	B	8 bits
Kilobayt	KB	1024 bayt
Megabayt	MB	1024 kilobayt
Gigabayt	GB	1024 megabayt
Terabayt	TB	1024 gigabayt
Petabayt	PB	1024 terabayt
Eksabayt	EB	1024 petabayt
Zettabayt	ZB	1024 eksabayt
Yottabayt	YB	1024 zettabayt

Tıpkı alfabeyi, konuşulan dilleri ve sayıları anlamak gibi dijital depolama birimlerinde de bilgili olmak önemlidir. Yeni dijital dünya bu yeni dijital depolama sistemlerini öğrenmeyi önemli hâle getirdi. Bu tür bir dil ve ölçümün anlaşılmasının faydaları sürekli artmaktadır. Bu dil sisteminde ne kadar akıcı olursak, dijital dünyayı o kadar hızlı anlarız. Zaman içinde veri

hacmi ile daha fazla depolama ihtiyacı arttıkça, kaçınılmaz olarak gerekli yeni kelimeleri de geliştireceğiz.

Örnekler:	
20 MB	= 20480 KB
8192 MB	= 8 GB
17 GB	= 17408 MB
9 MB	= 9216 KB
7168 EB	= 7 ZB

B5. Beni Bul ve Değiştir

Süre: 20 dk.

Kazanımlar: K5. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.

Materyaller: L1. B5. 1. Temel işlem tablosu

L1. B5. 2. İşleç bilgi kartları

L1. B5. 3. Örnek olay sunum kartı

Hazırlık: İşleç bilgi kartları ve örnek olay sunum kartı ÖYS üzerinden erişime açılırken, Temel işlem tablosundan 4 adet çıktı alınır.

Uygulama: Bu öğrenme görevi iki aşamadan oluşmaktadır. İlk olarak öğrenciler beşerli gruplara ayrılır. Eğitmen işleç kavramı hakkında kısa bir giriş yapar.

"Programlama dillerinde kendi başlarına kullanımlarında herhangi bir anlam ifade etmeyen fakat program akışına katkıda bulunan sembol veya sembol topluluklarına işleç denir."

Bu kısa bilgiden sonra her gruba temel işlem tablosu ve işleç bilgi kartları karışık şekilde verilir. Grupların ilk görevi işleç bilgi kartlarındaki işlemleri gruplayarak, temel işlem tablosunda doğru yere yazmaktır. İkinci görevde ise, öğrencilere örnek olay sunum kartını açmaları istenir ve onlardan aşağıdaki görevi yapıp grup kâğıdına yazmaları istenir.

- *Örnek olaydaki koşul ifadelerini temel işlem tablosundaki semboller ile değiştirmeyi deneyin. Önceki sonuçlar ile yeni sonuçları karşılaştırın.*

Burada öğrenciler örnek olayı incelerken düzenledikleri temel işlem tablosunda yer alan bilgileri kullanmalıdır. Görevi tamamlayan grup, bir diğeri ile yer değiştirerek grup kâğıdında yapılanları inceler. Görevin tamamlanmasının ardından öğrencilerin grup kâğıtlarında yaptıkları benzer ve farklı işlemler, beyin fırtınası ile sınıfta tartışılır. Sonuç olarak öğrencilerin aşağıdaki bilgilere erişmesi sağlanır.

Konu içeriği:

Aritmetik İşlemler

Matematiksel hesaplamalar bilgisayar programlarında yaygın olarak kullanılmaktadır. Programlama dilinde bir işleç, bir değer veya değişken üzerinde çalışan bir semboldür. İki sayıyı birbirine eklemek (3+5) gibi basit bir hesaplama yapabilen ya da $P(x) = 2x^3 - 3x^2 + 6x + 4$ gibi karmaşık bir denklemi çözebilen bir program yazabiliriz. Programlama dilinde bu iki ifade aritmetik ifadeler ve bu ifadelerde kullanılan artı (+), eksi (-) gibi semboller de aritmetik işleçler olarak adlandırılır. Bu ifadelerdeki 3, 5 ve x gibi değerler de işlenen olarak adlandırılır. İfadelerin matematiksel yazılımlarının uygun bir şekilde bilgisayar kodlanmasının gerçekleştirilmesi gerekmektedir. Aşağıdaki tablodaki örnekleri inceleyiniz.

Not: Sözde kod bilgisayar kodlamasında '^' sembolü üst alma olarak kullanılmaktadır.

Tablo 2. *Matematiksel ifadeler ve bilgisayar kodlamaları*

Matematiksel Yazılım	Bilgisayar Kodlanması
$x - 2y + 4xy$	x-2*y+4*x*y
$2x^2 - 3xy + y^2 + x^2y^3$	2*(x^2)-3*x*y+(y^2)+(x^2)*(y^3)
$x^2 + \frac{x}{5} + 3x^2y - \frac{x}{y}$	x*x+x/5+3*(x^2)*y-x/y
$\frac{2x^3 + 4y^2}{xy}$	(2*(x^3)+4*(y^2)) / (x*y)
$\sqrt{x+y} + \frac{3xy}{xy^2 - y}$	(x+y)^(1/2)+(3*x*y) / (x*(y^2)-y)

Aşağıdaki tabloda, bilgisayar programlamasında kullanılan önemli aritmetik işleçler listelenmiştir. Tabloda x ve y bir değişken olarak kabul edilmiştir.

***Tablo 3**. Aritmetik işleçler*

İşleç	Açıklama	Kullanım
+	İki işleneni birbirine ekler.	x + y
-	İkinci işleneni birinciden çıkarır.	x - y
*	İki işleneni çarpar.	x * y
/	İkinci işlenenle birinciyi böler.	x / y
%	Tamsayı bölümünden kalanı verir.	x % y
++	Sayıyı bir arttırma	x++ veya ++x
--	Sayıyı bir azaltma	x-- veya --x

Not: Bölme işlemlerinde elde edilen bölüm kısmı kesirli sayı olarak elde edilebilir. Fakat yukarıdaki tabloda verilen bölme işleminde "/" işleci ile bölme işlemi sonucunda bölümün tam kısmı alınmaktadır.

++ veya -- işleçleri değişken değerlerini 1 arttırmak ya da 1 azaltmak için kullanılır. "x++" ifadesinde x değişkeni var olan değeri ile işleme girer ve işlem tamamlandıktan sonra x değişkeninin değeri bir arttırılır. "++x" ifadesinde ise önce değişkenin değeri bir arttırılır ve daha sonra işlem yapılır. Aynı şekilde -- operatörü içinde benzer kullanım geçerlidir. Aşağıdaki örneği inceleyiniz.

```
a = 5
b = ++a          // a değeri 6 olur, b değeri 6 olur

c = 2
d = c++;         // c değeri 3 olur, d değeri 2 olur

a = 5
b = --a          // a değeri 4 olur, b değeri 4 olur

c = 2
d = c--;         // c değeri 1 olur, d değeri 2 olur
```

Karşılaştırma İşlemleri

Bu işleçlerin amacı iki değişken ya da değişken grubunu belirtilen şarta göre karşılaştırmaktır. Bu karşılaştırmalar aynı türdeki değişkenler arasında olmalıdır. Bu işleçleri özellikle ilerleyen haftalarda göreceğimiz koşul yapıları ve döngülerde kullanılır. Yapılan karşılaştırmalar doğruysa "1" yanlışsa "0" sonucunu döndürür.

Tablo 4. *Karşılaştırma işleçleri*

işleç	Açıklama	Kullanım
<	Küçüktür	x < y
>	Büyüktür	x > y
<=	Küçük eşittir	x <= y
>=	Büyük eşittir	x >= y
==	Eşittir	x == y
!=	Eşit değildir	x != y

Mantıksal İşlemler

Mantıksal işleçler programlama dillerinde çok önemli bir yere sahiptir ve belirli koşullara göre karar vermemize yardımcı olurlar. İki koşulun sonucunu birleştirmek istediğimizde mantıksal VE ve VEYA istediğimiz sonucu üretmemize yardımcı olur. Bütün şartların sağlanması için koşullar arasına VE, herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA ve koşulu sağlamayanlar isteniyorsa DEĞİL işleçi kullanılmalıdır.

Tablo 5. *Mantıksal işleçler*

işleç	Açıklama	Kullanım
&&	Mantıksal VE	x && y
||	Mantıksal VEYA	x || y
!	Mantıksal DEĞİL	!x

Mantıksal işleçlerin sonucu şu şekilde belirlenmektedir. Eğer x&&y kullanılıyor ise her iki değişkenin değeri "1" olursa sonuç "1" olur, aksi hâlde sonuç "0" olur. Eğer x||y kullanılıyor ise her iki değişkenin değeri "0" olursa sonuç "0" olur, aksi hâlde sonuç "1" olur. Eğer !x kullanılıyor ise değişkenin değeri "1" ise sonuç "0", "0" ise sonuç "1" olacaktır. Mantıksal VE için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 6. *Mantıksal VE kullanımı*

İşlenen 1	İşleç	İşlenen 2	Sonuç
0	&&	Sıfırdan farklı	0
Sıfırdan farklı	&&	0	0
0	&&	0	0
Sıfırdan farklı	&&	Sıfırdan farklı	1

Mantıksal VEYA için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 7. *Mantıksal VEYA kullanımı*

İşlenen 1	İşleç	İşlenen 2	Sonuç
0	||	Sıfırdan farklı	1
Sıfırdan farklı	||	0	1
0	||	0	0
Sıfırdan farklı	||	Sıfırdan farklı	1

Mantıksal DEĞİL için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 8. *Mantıksal DEĞİL kullanımı*

İşleç	İşlenen 2	Sonuç
!	Sıfırdan farklı	0
!	0	1

B6. Farklı Atama Türlerini Tanıyalım

Süre: 20 dk.

Kazanımlar: K6. Değişkenlere farklı türlerde değer ataması yapar.

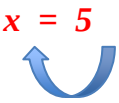
Materyaller: L1. B6.1. Yer Değiştirme Görev Kartları

Hazırlık: İşleç bilgi kartları ve örnek olay sunum kartı ÖYS üzerinden erişime açılırken, Temel işlem tablosundan 4 adet çıktı alınır.

Hazırlık: Materyalin çıktısı arkalı önlü gelecek şekilde 5 adet alınır ve kartlar her grup için ayrı ayrı kesilir.

Uygulama: Eğitmen atama işlemleri için değişken ve değer atama ile ilgili aşağıdaki gibi kısa bir giriş yapar.

Bir değişkene değer atamak için kullanılır. Atama operatörünün sol taraftaki işleneni değişkendir ve atama operatörünün sağ taraftaki işleneni bir değerdir.



x değişkenine 5 değeri atanıyor.

Öğrenciler dörderli beş gruba ayrılır. Eğitmen yukarıdaki örnek üzerinden gruplara farklı tür atama işleçlerini birlikte keşfedeceklerini belirtir ve onlara yer değiştirme görev kartları verir. Eğitmenin yukarıda verdiği değişken atama örneği üzerinden bu görev kartlarını uygulamaları ve aradaki değişimleri tartışmaları istenir. Eğitmen grup çalışmalarının tamamlanmasının ardından her gruptan bir sözcünün görev kartlarından birini diğer gruplara anlatmasını ister. Bu şekilde tüm gruplara bir kartı açıklama görevi verilir. Gruplar açıklama yaparken eğitmen de farklı tür atama işleçlerini aşağıdaki gibi tablo hâlinde tahtada özetler.

Konu İçeriği:

Farklı tür atama operatörleri aşağıda verilmektedir.

Tablo 9. *Atama operatörleri*

İşleç	Açıklama	Kullanım
=	Sağdaki değeri soldaki değişkene atamak için kullanılır.	x = y z=10
+=	Bu işleç, '+' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere ekler ve ardından sonucu soldaki değişkene atar.	x+=y (x=x+y)
-=	Bu işleç, '-' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerden çıkarır ve ardından sonucu soldaki değişkene atar.	x-=y (x=x-y)
*=	Bu işleç '*' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerle çarpar ve ardından sonucu soldaki değişkene atar.	x*=y (x=x*y)
/=	Bu işleç '/' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere böler ve ardından sonucu soldaki değişkene atar.	x/=y (x=x/y)
%=	Bu işleç '%' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere göre modunu alır ve ardından sonucu soldaki değişkene atar.	x%=y (x=x%y)

B7. Sayı Sistemlerini Keşfedelim

Süre: 20 dk.

Kazanımlar: K7. İkili, sekizli, onlu ve onaltılı sayı sistemlerini kullanır.

Materyaller: L1. B7. 1. Sayı sistemleri interaktif sunum

L1. B7. 2. Veri dönüştürme örnek olay kartları

Hazırlık: Materyallerden öğrencilere ÖYS üzerinden paylaşılır. İkinci materyal ise 5 grup için çıktı alınarak grup içinde her bir öğrencinin paylaşması için kesilir.

Uygulama: Eğitmen öğrencilerden ikişerli gruplar hâlinde interaktif sunum materyalini incelemelerini ister. Öğrencilerden bu materyalde karşılaştıkları basamak, basamak değeri, basamak ağırlığı ve sayı tabanı kavramlarını birbirleriyle tartışmaları beklenir. Devamında eğitmen sayı sistemlerinde basamak, basamak değeri, basamak ağırlığı ve sayı tabanının bilinmesinin önemini aşağıdaki şekilde kısa bir giriş ile vurgular.

Bilgisayarlar sadece sayıları anlayabildiğinden bazı harfleri ve kelimeleri yazdığımızda bunları otomatik olarak sayıya çevirmektedir. Sayıları temsil etme ve bunlarla çalışma tekniğine ***sayı sistemi*** *denir. Sayı sistemlerinde bir doğal sayıyı oluşturan her bir rakam bir* ***basamak*** *olarak adlandırılır. Rakamların bulundukları yerdeki değerine* ***basamak değeri****, her basamağın sahip olacağı üstel ifadeye* ***basamak ağırlığı*** *ve bu doğal sayının tanımlandığı sayı sistemine de* ***sayı tabanı*** denir.

Bizler günlük yaşantımızda onlu sayı sistemini kullanırken, bilgisayar sistemleri ikili sayı sistemini kullanır. Onlu sistemde taban 10, ikili sistemde ise taban 2'dir. İkili sayı sisteminin yanı sıra, sekizli ve onaltılı sayı sistemleri de sıklıkla kullanılmaktadır. Onlu sayı sistemlerinde her bir rakam ondalık basamak ya da sadece basamak olarak adlandırılırken, ikili sayı sistemlerinde ikili basamak ya da sadece bit olarak adlandırılır. Sayı sembolleri 0 ile (taban–1) arasında değer almaktadır.

n sayı tabanı ve a, b, c, d, e rakamları n'den küçük olmak üzere,

$A = (abcde)_n$ *sayısının basamakları*

e : n^0 *lar basamağı,*

d : n^1 *ler basamağı,*

c : n^2 *ler basamağı,*

b : n^3 *ler basamağı,*

a : n^4 *ler basamağıdır.*

$A = (abcde)_n = a.n^4 + b.n^3 + c.n^2 + d.n^1 + e.n^0$

şeklinde yazılmasına A sayısının n tabanına göre çözümlenmesi denir.

Bu kavramlardan sonra öğrenciler dörderli gruplar hâlinde çalışacaktır. Eğitmen gruplara veri dönüştürme örnek olay kartlarını verir ve grup bireylerinin her birinin bir kart seçmesini ister.

Kartlarını alan öğrenciler bunları inceler ve örnek olayı grup içindeki diğer arkadaşlarına anlatır. Daha sonra eğitmen bir örnek olay için aşağıdaki görevi tamamlamalarını ister.

> *Görev: Seçtiğiniz karttaki örneğe benzer bir başka örnek de siz yazın. Ancak örneklerdeki gibi basamak hesaplama kısmını grup arkadaşlarınıza bırakın. Bunun için yazdığın görevi yanındaki grup arkadaşın ile değiştirin.*

Konu İçeriği:

Onlu Sayı Sistemi

Onlu sayı sisteminde 0 ile 9 arasında yalnızca on rakam vardır. Bu sayı sisteminde her sayı 0, 1, 2, 3, 4, 5, 6, 7, 8 ve 9 ile temsil edilir. Onlu sayı sisteminin tabanı 10'dur, çünkü yalnızca 10 rakam kullanılır. Bu sistemde her bir basamak ağırlığı şu şekilde gösterilebilir:

10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

Örnek: Aşağıdaki onlu sayıların basamak analizini inceleyiniz.

$5429 = 5 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$

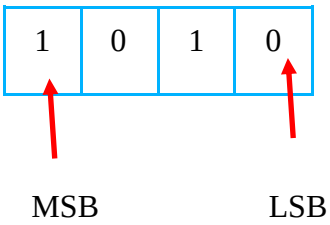
$387 = 3 \times 10^2 + 8 \times 10^1 + 7 \times 10^0$

İkili Sayı Sistemi

İkili sayı sisteminde yalnızca 0 ve 1 olmak üzere iki rakam bulunur. Her sayı, bu sayı sisteminde 0 ve 1 ile gösterilir. İkili sayı sisteminin tabanı 2'dir, çünkü sadece iki rakam kullanılır. Her ikili basamak ayrıca bit olarak ifade edilir. Bu sistemde her bir basamak ağırlığı şu şekilde gösterilebilir:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Herhangi bir ikili sayıda, en sağdaki basamağa en az anlamlı bit (LSB) ve en soldaki basamağa en anlamlı bit (MSB) denir.



Verilen bu sayının onlu eşdeğeri, basamak değeri ile her bir basamak ağırlığının çarpımının toplamıdır.

Örnek: Aşağıdaki ikili sayının basamak analizini inceleyiniz.

$(11001)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$= 16 + 8 + 0 + 0 + 1$

$= (25)_{10}$

Örnek: Aşağıdaki ikili sayının basamak analizini inceleyiniz.

$(0111010)_2 = 0\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 0 + 32 + 16 + 8 + 0 + 2 + 0$

$= (58)_{10}$

Sekizli Sayı Sistemi

Sekizli sayı sisteminde 0 ile 7 arasında sadece sekiz basamak vardır. Her sayı, bu sayı sisteminde 0, 1, 2, 3, 4, 5, 6 ve 7 ile temsil edilir. Sekizli sayı sisteminin tabanı 8'dir, çünkü sadece 8 basamaklıdır.

8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0

Herhangi bir sekizli sayının onlu eşdeğeri, basamak değeri ile her bir basamak ağırlığının çarpımının toplamıdır.

Örnek: Aşağıdaki sekizli sayının basamak analizini inceleyiniz.

$(247)_8 = 2\times8^2 + 4\times8^1 + 7\times8^0$

$= 128 + 32 + 7$

$= (167)_{10}$

Örnek: Aşağıdaki sekizli sayının basamak analizini inceleyiniz.

$(2315)_8 = 2\times8^3 + 3\times8^2 + 1\times8^1 + 5\times8^0$

$= 1024 + 192 + 8 + 5$

$= (1229)_{10}$

Onaltılı Sayı Sistemi

Onaltılı bir sayı sistemi, 0'dan 9'a ve A'dan F'ye kadar on altı alfasayısal değere sahiptir. Her sayı, bu sayı sisteminde 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E ve F ile temsil edilir. Onaltılı

sayı sisteminin tabanı 16'dır. Çünkü 16 alfasayısal değere sahiptir. Burada A:10, B:11, C:12, D:13, E:14 ve F:15 değerine sahiptir.

16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

Herhangi bir onaltılı sayının onlu eşdeğeri, basamak değeri ile her bir basamak ağırlığının çarpımının toplamıdır.

Örnek: Aşağıdaki onaltılı sayının basamak analizini inceleyiniz.

$(1A4F)_{16} = 1\times16^3 + 10\times16^2 + 4\times16^1 + 15\times16^0$

$= 4096 + 2560 + 64 + 15$

$= (6735)_{10}$

Örnek: Aşağıdaki onaltılı sayının basamak analizini inceleyiniz.

$(32EC)_{16} = 3\times16^3 + 2\times16^2 + 14\times16^1 + 12\times16^0$

$= 12288 + 512 + 224 + 12$

$= (13036)_{10}$

Aşağıdaki tabloda ikili, sekizli, onlu ve onaltılı sayı sistemleri arasındaki ilişki gösterilmektedir.

Tablo 10. *Sayı sistemleri dönüşümü*

Onaltılı	Onlu	Sekizli	İkili
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001

A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

Onlu Sayı Sisteminden İkili Sayı Sistemine Dönüştürme

İkili sayı sisteminden onlu sayı sistemine ve onlu sayı sisteminden ikili sayı sistemine dönüştürme işlemleri tüm bilgisayar sistemlerinin temelini oluşturduğundan önemli bir kavramdır. Yukarıdaki bölümde ikili sayı sisteminden onlu sayıya dönüştürmenin nasıl yapıldığını örneklerle gösterdik. Onlu sayı sisteminden ikili sayı sistemine dönüştürme için aşağıdaki adımlar kullanılır:

1. Sayıyı 2 ile bölünüz.
2. Sonraki yineleme için tamsayı bölümünü alın.
3. İkili sayının oluşturulması için kalanı alın.
4. Bölüm 0'a eşit olana kadar adımları tekrarlayın.

Örnek: $(47)_{10}$ sayısını ikili sayı sistemine dönüştürelim.

2 ile Bölme	Bölüm	Kalan	Bit Numarası
47/2	23	1	0
23/2	11	1	1
11/2	5	1	2
5/2	2	1	3
2/2	1	0	4
1/2	0	1	5

Sonuç olarak $(47)_{10} = (101111)_2$ olmaktadır.

Örnek: $(173)_{10}$ sayısını ikili sayı sistemine dönüştürelim.

2 ile Bölme	Bölüm	Kalan	Bit Numarası
173/2	86	1	0
86/2	43	0	1
43/2	21	1	2
21/2	10	1	3
10/2	5	0	4
5/2	2	1	5
2/2	1	0	6
1/2	0	1	7

Sonuç olarak $(173)_{10} = (10101101)_2$ olmaktadır.

C. Kısmi Öğrenme Görevleri

Süre: 50 dk.

Materyal: L1. C. 1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Kısmi Öğrenme Görevi Yanıtlar:

Tasarlayıcı: Otonom bir otomobil tasarlamak istiyoruz. Yoldaki çizgileri takip eden, hız sınırlarına dikkat eden, Otonom bir araç için giriş-çıkış, CPU, Bellek görevleri neler olmalıdır? Çizerek gösterelim.

Yanıt: Bu soru öğrencilerin hayal yeteneklerini geliştirmek için verilmiştir. Tek bir doğru cevabı yoktur. Giriş olarak: Kamera Çıkış olarak: Direksiyon, Gaz ve Fren pedalları düşünülebilir. CPU: görüntü işlemek için, bellek ise hız sınırlarının tutulması için kullanılabilir.

Tasarlayıcı: Aşağıdaki ifadeyi bilgisayar kodlanmasına çeviriniz.

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

Yanıt: x=-b + (((b^2 - 4*a*c)^(½))/ (2*a)

Analizci: Aşağıda verilen Fizik dersini geçme kriterlerini mantıksal koşul olarak tek satırda ifade ediniz.

Tüm deneyleri okulda (A) ya da evde (B) yapmak

Tüm ödevleri teslim etmek ©

Sınav ortalaması (D) 50'den büyük

Devamsızlığı (E) 20 günden az olmak

Yanıt: **(**A || B) && C && (D>50) &&(E<20)

Kodlayıcı: 8-bitlik siyah/beyaz görüntülerin her piksel (en küçük görüntü parçası) değeri 0-255 arasında bir değer alır. Genişliği 200 piksel ve yüksekliği 300 piksel olan bir görüntü için kaç KB hafıza gerekir?

Yanıt: 200 * 300*1 Bayt = 60000 Bayt = 58.6KBayt

Tasarlayıcı: Aşağıdaki tabloyu bilgisayarda sayısal olarak tutmak istersek; ikili ve onlu tabanda olması gereken değer kaçtır?

(O:0 ve X:1 olarak alınmalı.)

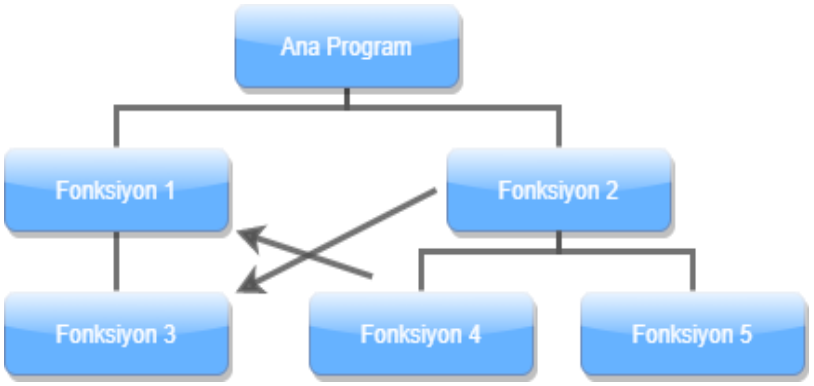
O	X	X	O	X	O	X	O

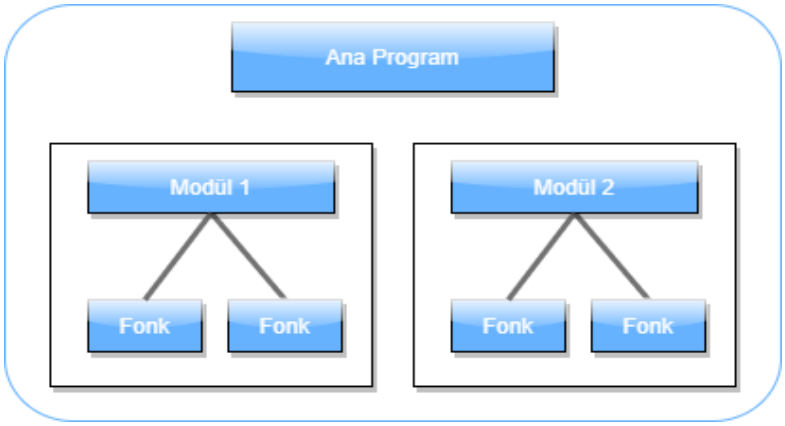
Yanıt: $(01101010)_2 = (106)_{10}$

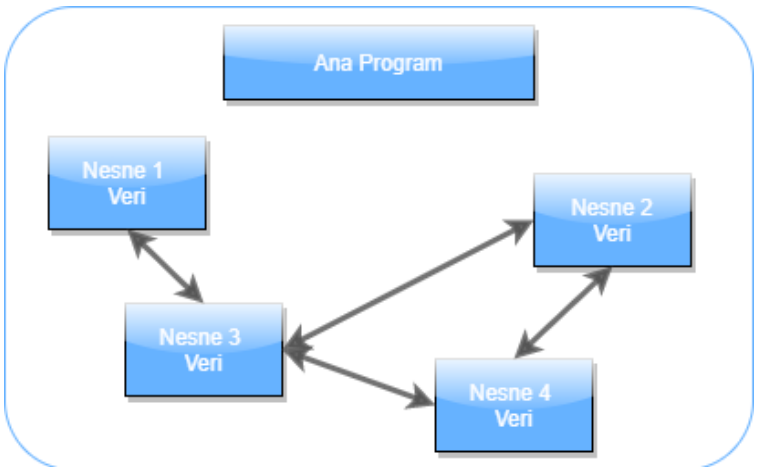
Hafta 1. Ders Materyalleri

L1. B2. 1. Eşleştirme Kartları

L1. B3.1. Bilgi Kartları







L1. B4. 1. Sıralama kartları (Ön Yüz)

BİT
(BİT)

1 BİT

BİT
(BİT)

2 BİT

BİT
(BİT)

5 BİT

MEGABYTE
(MB)

1 MB

GİGABYTE
(GB)

1 GB

TERABYTE
(TB)

1 TB

BİT
(BİT)

6 BİT

BYTE
(B)

1 BYTE

KİLOBAYT
(KB)

1 KB

BİT
(BİT)

8 BİT

L1. B4. 1. Sıralama Kartları (Arka Yüz)

Bilgisayar sistemlerindeki bütün bilgiler ikilik sistemde “0” ve “1” ile temsil edilerek saklanır. İkilik sistemdeki her bir basamağa bit denir.	Bilgisayarda depolanan bilgilerin “01” , “10”, “00”, “11” şeklinde bir araya gelmesiyle oluşur.	Bilgisayarda depolanan bilgilerin “01010” , “10001”, “00000”, “11111” şeklinde beş tane rakamın bir araya gelmesiyle oluşur.
1024 tane Kilobyte’ın bir araya gelmesiyle oluşan hafıza birimidir.	1024 tane Megabyte’ın bir araya gelmesiyle oluşan hafıza birimidir.	1024 tane Gigabyte’ın bir araya gelmesiyle oluşan hafıza birimidir.
Bilgisayarda depolanan bilgilerin “010101” , “100010”, “000000”, “111111” şeklinde altı tane rakamın bir araya gelmesiyle oluşur.	Bilgisayarda depolanan bilgilerin “01010101” , “10001001”, “00000000”, “11111111” şeklinde sekiz tane rakamın bir araya gelmesiyle oluşur.	1024 tane byte’ın bir araya gelmesiyle oluşan hafıza birimidir.
	Bilgisayarda depolanan bilgilerin “01010101” , “10001001”, “00000000”, “11111111” şeklinde sekiz tane rakamın bir araya gelmesiyle oluşur.	

L1. B4. 2. Çalışma yaprağı

Tablo-11, verilerimizi depolamak için kullanılan dosya biçimlerini ve bilgisayar depolama birimlerinden oluşturdukları kapasitelerini göstermektedir. Tablo-12, ilk tabloda gösterilen dosya biçimlerini depolamamıza yardımcı olacak donanımları ve sahip oldukları depolama alanlarını belirmektedir.

Bu çalışma kâğıdında sizden istenilen Tablo-2'de donanımların karşısında yer alan boşluklu kısımları doldurmanızdır. Tablo-1'den yararlanarak donanım ürünlerinin kaçar adet belge, resim, müzik, film ve oyun saklayabileceğini yazınız.

Tablo 11. *Dosyalarımın boyutu*

Dosya Biçimleri ve Boyutları				
				
Belge	Resim	Müzik	Film	Oyun
2 MB	4 MB	5 MB	1.2 GB	5 GB

Tablo 12. *Donanımın içine sığabilecek dosyalar*

Donanım	Her Birinden Kaç Tane Sığar				
	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun
Disket (1.4 MB)	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun
CD (700 MB)	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun
DVD (4.7 GB)	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun
Flash Bellek (32 GB)	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun
Hard-Disk (1 TB)	…… Belge	…… Resim	…… Müzik	…….. Film	…….. Oyun

L1. B5. 1. Temel İşlemler Çalışma Kâğıdı

Aritmetik İşlemler	Karşılaştırmalı İşlemler	Mantıksal İşlemler
Programlama dilinde iki ifade arasında toplama, çıkarma, bölme, yüzdesini alma gibi matematiksel işlemler için kullanılan sembollerdir.	*Programlama dilinde iki ifadenin büyüklük küçüklük ya da eşitlik açısından değerlendirilmesi için kullanılan sembollerdir.*	*Programlama dilinde farklı durumların birlikte değerlendirilmesinde kullanılan semboller mantıksal işleçler olarak adlandırılır.*

L1. B5. 2. İşleç Bilgi Kartları

Aritmetik İşlemler

Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. Arda ve Duru'nun yaşları arasında tanımlanabilecek aritmetik işlemleri inceleyiniz.

İşlem	Açıklama	Sonuç
a + d	Arda ve Duru'nun yaşlarını toplama	15+4=19
a – d	Arda'nın yaşından Duru'nun yaşını çıkarma	15-4=11
a * d	Arda ve Duru'nun yaşlarını çarpma	15*4=60
a / d	Arda'nın yaşını Duru'nun yaşı ile bölme	15/4=3
a % d	Arda'nın yaşını Duru'nun yaşına göre mod alma	15%4=3
a++	Arda'nın yaşını bir artırma	16
d--	Duru'nun yaşını bir azaltma	3

Karşılaştırma İşlemleri

Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. Arda ve Duru'nun yaşları arasında tanımlanabilecek karşılaştırma işlemleri inceleyiniz.

İşlem	Açıklama	Sonuç
a < d	Arda'nın yaşı, Duru'nun yaşından küçüktür.	0 (Yanlış)
a > d	Arda'nın yaşı, Duru'nun yaşından büyüktür.	1 (Doğru)
a <= d	Arda'nın yaşı, Duru'nun yaşına küçük eşittir.	0 (Yanlış)
a >= d	Arda'nın yaşı, Duru'nun yaşına büyük eşittir.	1 (Doğru)
a == d	Arda'nın yaşı, Duru'nun yaşına eşittir.	0 (Yanlış)
a != d	Arda'nın yaşı, Duru'nun yaşına eşit değildir.	1 (Doğru)

Mantıksal İşlemleri

Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. Arda ve Duru'nun yaşları arasında tanımlanabilecek mantıksal işlemleri inceleyiniz.

İşlem	Açıklama	Sonuç
((a==5) && (d>5))	Arda 5 yaşındadır VE Duru 5 yaşından büyüktür.	0 (Yanlış)
((a==5) || (d>5))	Arda 5 yaşındadır VEYA Duru 5 yaşından büyüktür.	1 (Doğru)
!(a==4)	Arda 4 yaşında değildir.	1 (Doğru)

L1. B5. 3. Örnek Olay Sunum Kartı

Bir okulda yaşı 15'in üzerinde olan ve 9. sınıfta okuyan öğrenciler için gezi planı yapılıyor. Okul müdürü sizden bu öğrencilerin isimlerini istedi. Bu durumda bu kriterlere uygun öğrencilerin ismini listelemek için aşağıdaki gibi bir ifade oluşturmalıyız.

Bu örnekte iki koşul bulunmaktadır. Bu iki koşulun da doğru (1) olması gerekmektedir. Bunun için,

Eğer Yaş > 15 **VE** Sınıf == 9 ise öğrenci ismi yaz

1. Koşul 2. Koşul

L1. B6. 1. Yer Değiştirme Görev Kartları (Ön Yüz)

*Görev kartları arkalı önlü olacak şekilde basılmalıdır. Görev kartlarındaki her satır bir görevi oluşturmaktadır.

L1. B6. 1. Yer Değiştirme Görev Kartları (Arka Yüz)

*Görev kartları arkalı önlü olacak şekilde basılmalıdır. Görev kartlarındaki her satır bir görevi oluşturmaktadır.

L1. B7. 1. Sayı sistemleri sunum

Powerpoint ortamında hazırlanan sunum animasyon şeklinde öğrencilere izletilmelidir. Sunuma eğitim paketinin hafta 1 materyaller klasörü içinden L1.B7.1 kodu ile erişebilirsiniz.

L1. B7. 2. Veri dönüştürme örnek olay kartları

Aşağıdaki ikili sayıların onlu sisteme dönüşümünü inceleyiniz.

$(11001)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$= 16 + 8 + 0 + 0 + 1$

$= (25)_{10}$

$(0111010)_2 = 0\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 0 + 32 + 16 + 8 + 0 + 2 + 0$

$= (58)_{10}$

Aşağıdaki sekizli sayının onlu sisteme dönüşümünü inceleyiniz.

$(247)_8 = 2\times8^2 + 4\times8^1 + 7\times8^0$

$= 128 + 32 + 7$

$= (167)_{10}$

$(2315)_8 = 2\times8^3 + 3\times8^2 + 1\times8^1 + 5\times8^0$

$= 1024 + 192 + 8 + 5$

$= (1229)_{10}$

Aşağıdaki onaltılı sayıların onlu sisteme dönüşümünü inceleyiniz.

$(1A4F)_{16} = 1\times16^3 + 10\times16^2 + 4\times16^1 + 15\times16^0$

$= 4096 + 2560 + 64 + 15$

$= (6735)_{10}$

$(32EC)_{16} = 3\times16^3 + 2\times16^2 + 14\times16^1 + 12\times16^0$

$= 12288 + 512 + 224 + 12$

$= (13036)_{10}$

$(47)_{10}$ sayısının ikili sayı sistemine dönüşümünü inceleyiniz.

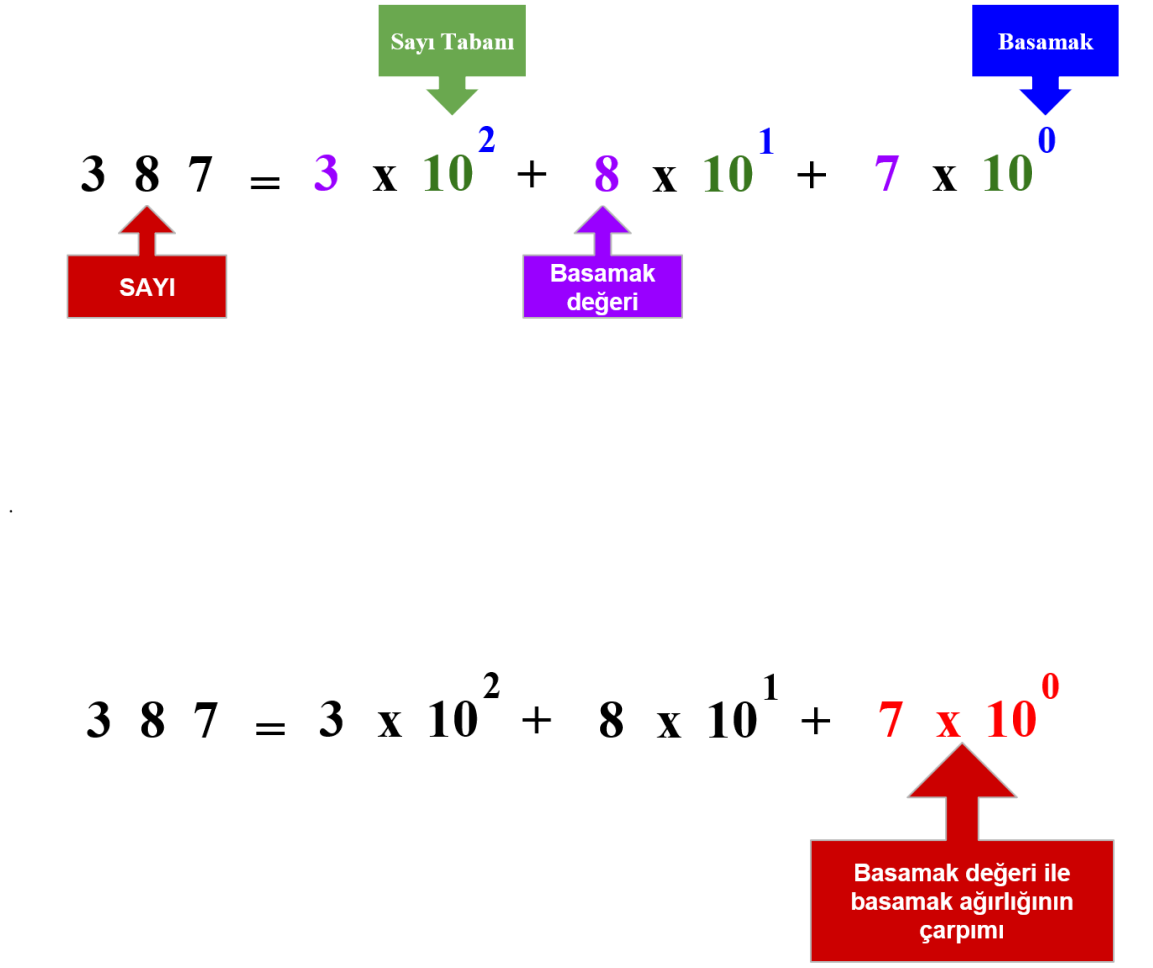
Bunun için aşağıdaki adımlar kullanılır:

1. Sayıyı 2 ile bölünüz.
2. Sonraki yineleme için tamsayı bölümünü alın.
3. İkili basamak için kalanı alın.
4. Bölüm 0'a eşit olana kadar adımları tekrarlayın.

2 ile Bölme	Bölüm	Kalan	Bit Numarası
47/2	23	1	0
23/2	11	1	1
11/2	5	1	2
5/2	2	1	3
2/2	1	0	4
1/2	0	1	5

Sonuç olarak $(47)_{10} = (101111)_2$ olmaktadır.

L1.B7.1. Basamak İşlemi



Resim 9. *Basamak işlemi sorusu*

L1. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 2. Algoritma Tasarımı

Kazanımlar

- K1. Algoritma temel kavramlarını analiz eder.
- K2. Bir problemin çözümüne uygun farklı algoritmalar tasarlar.
- K3. Bir problemin çözümüne uygun algoritmanın akış şemasını bilgisayar ortamında oluşturur.
- K4. Bir problemin çözümünde değişkenleri kullanır.
- K5. Verilen algoritmanın akış diyagramını sözde koda dönüştürür.
- K6. Verilen algoritmanın akış diyagramındaki değişkenleri ayırt eder.

Amaç

Haftanın amacı kâğıt üzerinde bir problemi çözmek için kod yazmada gerekli algoritmik düşünme yeteneğini geliştirmek ve problemin çözümünü bilgisayar kullanarak aktarmaktır.

Önerilen Ders Akışı

A. Giriş: Sona Kalan Kazanır! (10 dk.)
B. Öğrenme Görevleri
   B1. Algoritmayı Tanıyorum! (20 dk.)
   B2. Algoritmaları Eşleştir! (20 dk.)
   - ❏ Ders Arası (5 dk.)
   - ❏ Motivasyon Oyunu (10 dk.)

   B3. Çiftleri Topla! (25 dk.)
   B4. Otomatik Park Etme! (20 dk.)
   - ❏ Ders Arası (10 dk.)

   B5. Değişkenler Değerlidir! (20 dk.)
   B6. Algoritmayı Test Et! (25 dk.)
   - ❏ Ders Arası (5 dk.)
   - ❏ Motivasyon Oyunu (10 dk.)

C. Kısmi Görev Uygulamaları (60 dk.)

A. Giriş: Sona Kalan Kazanır!

Süre: 10 dk.

Uygulama: Eğitmen bütün öğrencilerin oldukları yerde ayağa kalkmalarını ister. Basit sorular soracağını ve bu sorular için cevabı "evet, o benim" diyenlerin yerlerine oturmaları gerektiğini, en son ayakta kalanın ise oyunu kazanacağını belirtir. Ayakta kalan son öğrenciye sürpriz olarak küçük bir hediye verilir. Bu buz kırıcı teknik, basitten karmaşığa sorular sorularak uygulanabilir. Örneğin: Kimin kız kardeşi var? / Kim eylül ayında doğdu? / Kimin gözleri siyah? / Kimin kolunda saat var?

Eğitmene Öneriler: Süreye bağlı olarak sorular çeşitlendirilip arttırılabilir.

B. Öğrenme Görevleri

B1. Algoritmayı Tanıyorum!

Süre: 20 dk.

Kazanımlar: K1. Algoritma temel kavramlarını analiz eder.

Materyaller: L2. B1. 2. Doğru algoritma hangisi?

Hazırlık: Ders materyali ÖYS üzerinden öğrenci erişimine açılır.

Uygulama: Eğitmen algoritmanın ne olduğu ile ilgili konuya dikkat çeker ve aşağıdaki gibi kısa bir giriş yapar.

> *Algoritma, bir problemi ya da sorunu çözmek için izlenecek mantıksal kural ve adımların listesidir. Diğer bir ifadeyle, bir problemi çözmek için takip edilecek sonlu sayıda adımdan oluşan bir çözüm yoludur. Mevcut var olan bilgilerden istenilenlere erişmemizi sağlar. Bir algoritmayı anlamanın en iyi yolu onu bir tarif (ilaç, yemek vb.) olarak düşünmektir. Günlük hayattaki yaşantı şeklimizde düzenli olarak birtakım işlemlerin sıra ile yapılması şeklindedir. Yani bir işi yapabilmek için bir takım alt işleri peş peşe gerçekleştiririz.*

Daha sonra öğrencilerden çizgi-nokta oyunundaki noktaların ikili grup hâlinde birleşmelerini ve ÖYS'de haftanın materyallerinden "Doğru Algoritma Hangisi" başlıklı olanı açmalarını ister. Daha sonra eğitmen öğrencilere materyal üzerinden aşağıdaki görevleri yapmalarını ister.

1. *Verilen kek algoritmaları arasındaki benzerlik ve farklılıkları düşünün. En iyi yazılan kek algoritmasını bulun.*
2. *Kek algoritması üzerinden doğru algoritma yazma özellikleri hakkında grup arkadaşınızla tartışınız.*

Bu görevleri yaparken öğrencilerden algoritmanın özellikleri üzerine düşünmeleri beklenir. Eğitmen görev sonunda sınıfta öğrencilere beyin fırtınası yaptırarak, onların algoritmanın özelliklerini aşağıdaki gibi keşfetmelerine imkân verir.

- *Bir başlangıç noktası olmalı*
- *Basit olmalı*
- *Algoritma içindeki ifadeler herkes tarafından aynı şekilde anlaşılabilir olmalı*
- *Yorum gerektirmemeli ve belirsiz ifadelere sahip olmamalı*
- *Her adımda tek bir iş yapılmalı*
- *Adımların gerçekleşme sırası doğru olmalı*
- *Mümkün olan en az adım ile en kısa sürede gerçekleşmeli*
- *Sonsuz döngüye girmemeli*
- *Bir bitiş noktası olmalı*

Eğitmene Öneriler: Görev sonunda eğitmen öğrencilere en iyi yazılmış kek algoritmasının diğerinden farklı ve benzer olan yanları üzerine sorular sorabilir. Bunlar algoritmanın özellikleri için ipuçları verecektir.

B2. Algoritmaları Eşleştir!

Süre: 20 dk.

Kazanımlar: K1. Algoritma temel kavramlarını analiz eder.

Materyaller: L2. B2. 1. Algoritmaları Eşleştirelim Çalışma Kâğıdı

Hazırlık: Nokta hâlinde oturan öğrencilerden yanlarındaki bir noktayla birleşmeleri ve dörtlü gruplar hâlinde oturmaları sağlanır. Ders materyali birer adet çıktısı alınır.

Uygulama: Eğitmen dörderli gruplar hâlinde oturan öğrencilere "Algoritmaları Eşleştirelim" materyali dağıtılır. Materyalde üç farklı problemin metin, sözde kod ve akış diyagramı şeklinde hazırlanmış algoritması vardır. Öğrenciler problemi ve yazım şekillerini eşleştirerek tahmin etmeleri istenir. Bu etkinlik ile öğrencilerden bir problemin algoritmasının üç farklı şekilde oluşturulabildiğini keşfetmeleri sağlanır.

Eğitmene Öneriler: Bu etkinlikte öğrencilerin edinmesi gereken içerik için aşağıdaki bilgiler incelenebilir.

a) Metin olarak yazım: Problemin çözüm adımları, düz metin olarak açık cümlelerle ifade edilir. Algoritmadaki her bir satıra numara verilir. "Başla" ile başlayıp "Bitir" ile bitirilir.

b) Sözde kod (pseudocode) yazım: Problemin çözüm adımları komut benzeri anlaşılır metinlerle ifade edilir. Sözde kod konuşma dilinde ve programlama mantığı altında, "eğer", "iken" gibi koşul kelimeleri ve '>', '=', '<' gibi ifadeler ile birlikte yazılır. Algoritma yazma ile kod yazma arasında kalan bir kısımdır ama kodlamaya daha yakındır. İyi yazılmış sözde koddan bir programlama diline kolaylıkla geçiş yapabilirsiniz.

c) Akış diyagramlarının çizilmesi: Problemin çözüm adımları, görsel olarak simge ya da sembollerle gösterilir. Akış şemalarının algoritmadan farkı, algoritma adımlarının

semboller şeklinde kutulara yazılması ve adımlar arasındaki ilişki ve yönün oklar ile gösterilmesidir.

B3. Çiftleri Topla!

Süre: 25 dk.

Kazanımlar: K2. Bir problemin çözümüne uygun farklı algoritmalar tasarlar.

K3. Bir problemin çözümüne uygun algoritmanın akış şemasını bilgisayar ortamında oluşturur.

Materyaller: L2. B3. 1. Çiftleri Toplama Algoritması

L2. B3. 2. Draw.io Program Kullanım Kılavuzu

L2. B3. 3. Akış Diyagramları Bilgi Afişi

L2. B3. 4. Algoritma Terimleri Görev Kartları

Hazırlık: Eğitmen nokta oluşturan öğrencilerle ikişerli gruplar oluşturur. Etkinlik materyallerinden bilgi afişi ve kullanım kılavuzu ÖYS'de öğrencilerin erişimine açılır. Birinci materyalden de 10 adet çıktı alınır. İki öğrenciye bir tane gelecek şekilde sınıfta dağıtılır. Görev kartları ise beş adet çıktı alınarak kesilir ve her gruba karışık şekilde dağıtılır.

Uygulama: Eğitmen nokta oluşturan öğrencilerle ikişerli gruplar oluşturur. Eğitmen her gruba sözde kodlar hâlinde ancak karışık sırada yazılmış, çiftleri toplama algoritmasının çıktısını verir. Gruplar sırasıyla aşağıdaki görevleri tamamlamalıdır.

1. *Sözde kod hâlinde verilen çiftleri toplama algoritmasını doğru şekilde sıralayın. Bu aşamada ÖYS'de erişimi açılan "Akış Diyagramları Bilgi Afişini" inceleyin. Nokta hâlindeki bir diğer grup ile sıralamanızı karşılaştırın.*
2. *Sıralanan algoritmanın akış şemasında kullanılacak diyagramları belirleyin ve "Draw.io" programını kullanarak algoritmayı çizin. Bu aşamada Draw.io program kullanım kılavuzundan yararlanın.*

Öğrenciler bu görevlerden sonra dörderli gruplar hâlinde çalışmaktadır. Bu aşamada öğrencilerin tanımlayıcı, değişken, sabit, sayaç, döngü, atama/aktarma ve ardışık toplama/çıkarma terimlerini öğrenmeleri beklenir. Eğitmen öğrencilerden çiftleri toplama algoritmasının akış şemasındaki boşlukları doldurmalarını ister. Doğru algoritmaya eriştikten sonra eğitmen görev kartlarından ilkini Tanımlayıcı görev kartını sınıfta uygular. Diğer görev kartlarını ise gruplara ikişer tane olacak şekilde rastgele dağıtır ve aşağıdaki talimatı verir. Etkinlik sonunda eğitmen aşağıdaki konu içeriğini yapılan görevler üzerinden özetler.

Çiftleri toplama algoritması üzerinde "Algoritma terimleri görev kartlarını" uygulayın.

Eğitmene Öneriler: Draw.io programında akış şemaları çizim özelliklerinden öğrencilere

kısaca bahsedilebilir. Program kılavuzunu ders öncesinde inceleyenler, incelemeyenler ile gruplanabilir.

Konu İçeriği:

1) Tanımlayıcı: Algoritmadaki değişkenleri, sabitleri, kayıt alanlarını ve özel bilgi tiplerini adlandırmak programcı tarafından gerçekleştirilen işlemlerdir. Tanımlayıcıların yerini tuttukları ifadelere çağrışım yapacak şekilde adlandırılmaları algoritmanın anlaşılması açısından önemlidir. Tanımlayıcı isimlerinde İngiliz alfabesindeki "A-Z" ve "a-z" arası harfler, "0-9" arası rakamlar, sembollerden alt çizgi "_" kullanılabilir. Bununla birlikte tanımlayıcı ismi, rakamla başlayamaz veya sadece rakamlardan oluşamaz.

2) Değişken: Algoritmanın her çalıştırılmasında, girdiğimiz değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı bellek alanlarıdır. Değişken isimlendirilmeleri, yukarıda sayılan tanımlayıcı kurallarına uygun biçimde yapılmalıdır. Örneğin bir ismin aktarıldığı değişken "ad" olarak, bir isim ve soy ismin aktarıldığı değişken "adSoyad", ev telefon numarasını aktarıldığı değişken "evTel", ev adresinin aktarıldığı değişken "evAdres" ve iş adresinin aktarıldığı değişken "isAdres" olarak tanımlanabilir. Burada özellikle İngiliz alfabesini kullandığımıza dikkat edelim.

3) Sabit: Uygulamanın çalıştığı süre boyunca, içeriği sabit olan değer ve ifadelerin saklanması için kullanılır. Algoritma boyunca kolay takip edilebilmesi için kullanılmaktadır. Değişkenler gibi isimlendirme kurallarına uygun olarak oluşturulmalıdırlar.

4) Atama/Aktarma: Bir değişkene değer atamak için gerçekleştirilen işlemdir. Algoritma adımlarında sağdaki değeri soldaki değişkene atamak için kullanılan bir işlemdir.

5) Sayaç: Algoritma tasarımlarında bazı işlemlerin belirli sayıda yaptırılması ve bu süreçte oluşan değerlerin sayılması gerekebilir. Bu amaçla kullanılan sayma işlemlerine sayaç denir.

6) Döngü: Algoritma tasarımlarında bir veya birden fazla işlem satırını, bir koşula bağlı olarak, belirli sayıda veya bir koşul sağlandığı sürece tekrarlayarak çalıştıran kalıplardır. Örneğin 1 ile 100 arasındaki çift sayıların toplamını hesaplayan programda T=2+4+6 ….+100 hesabını teker teker yapmak yerine 1 ile 100 arasında ikişer artan bir döngü kullanmak ve döngü değişkenini toplam hesabında kullanmak daha uygun olacaktır.

7) Ardışık Toplama/Çarpma: Algoritmalarda var olan değere yeni bir değerin eklenmesi ya da var olan değerin yeni bir değerlerle çarpılarak oluşan bu yeni değerin kullanılması işlemidir.

B4. Otomatik Park Etme!

Süre: 20 dk.

Kazanımlar: K2. Bir problemin çözümüne uygun farklı algoritmalar tasarlar.

Materyaller: L2. B4. 1. Otomatik Park Etme Algoritması

Hazırlık: Eğitmen iki nokta oluşturan öğrencileri birleştirerek dörderli gruplar oluşturur. Etkinlik materyalleri ÖYS'de öğrencilerin erişimine açılır.

Uygulama: Eğitmen öğrencilerden ÖYS'deki "Otomatik park etme algoritması" üzerinde aşağıdaki problemleri incelemelerini ve algoritmayı tekrar düşünmelerini ister.

1. Otoparkta boş yer yok ise, algoritma nasıl çalışacaktır? Grup içinde tartışın ve grup kâğıdına yazın.
2. Algoritmanın daha iyi çalışması için basamaklarda nasıl bir değişiklik yapılmalıdır? Değiştirilen algoritma önerinizi grup kâğıdına yazınız.

Eğitmen birinci görevin grup içinde tartışılmasının ardından, grupların saat yönünde yer değiştirmelerini ister. Yeni grup, bir önceki grubun grup kâğıdında tamamlanan görevi inceleyip kontrolünü gerçekleştirir. Ardından önceki grubun kâğıdından ikinci göreve devam edilir. İki görevin tamamlanmasının ardından sınıfta beyin fırtınası yapılarak algoritmalarda "Sonsuz döngü" problemi aşağıdaki gibi özetlenir.

> *Akıllı araç ilk sokağa geldiğinde yer olmadığı (sensörler yardımıyla) algılayacak ve sonraki sokağa kadar ilerleyecektir. İkinci sokakta P2 ve P3 alanlarının boş olması durumunda P2 alanına aracını park edecektir. Eğer otoparkta boş yer yok ise, 2. ve 3. adımlarda **sonsuz döngü** dediğimiz istenmeyen durumla karşılaşılacaktır. Bu durumu da göz önüne alarak, otoparkın sonuna gelindiğinde akışın sonlanmasını istiyorsak aşağıdaki şekilde yazabiliriz.*
>
> ***Algoritmayı Sonsuz Döngüden Kurtarma:***
>
> 1. *Başla.*
> 2. *Sonraki sokağa kadar ilerle.*
> 3. *Eğer otoparkın sonuna geldiysen 7. adıma git.*
> 4. *Eğer sokakta yer yoksa 2. adıma git.*
> 5. *Sokağa gir.*
> 6. *İlk uygun yere park et.*
> 7. *Bitir.*

Eğitmene Öneriler: Eğitmen öğrencilere algoritmadaki problemi keşfetmeleri için ipuçları kullanabilir. Örneğin; "Basamakları tek tek çalıştırmayı dene"; "İkinci basamağın tekrar çalıştırmayı düşünebilirsin"; "son basamağın nasıl çalıştığına dikkat et" vb. gibi ifadelerle problemin kaynağını doğrudan öğrenciye aktarmaktan kaçınılmalıdır.

B5. Değişkenler Değerlidir!

Süre: 20 dk.

Kazanımlar: K4. Bir problemin çözümünde değişkenleri kullanır.

Materyaller: L2. B5. 1. Örnek Kod Çalıştırma Tablosu

Hazırlık: Materyaller ÖYS üzerinden materyal olarak öğrencilere yayınlanır.

Uygulama: Eğitmen öğrencilere aşağıdaki örnek olayı verir. Öğrenciler ikili gruplar hâlinde çalışmaktadır.

Örnek Olay: Arkadaşınız aklından iki sayı tutmuştur ve sizden bu iki sayıdan büyük olanı bulmanızı istiyor. Bunun için bir algoritma yazmak istiyorsunuz.

Eğitmen ilgili algoritmanın sözde kodunu, değişken, değişken değerleri ve ekran çıktısını içeren aşağıdaki talimatları verir.

1. Örnek olaya ilişkin algoritma için değişkenleri ve değerlerini düşünün.
2. Bu değişkenleri kullanarak akış şemasını çizin ve sözde kodları oluşturup sıralayın
3. Önceki iki görev iki kişilik gruplarda yapıldı. Şimdi iki grup birleşerek dörderli yeni gruplar oluşturun ve elinizdeki kodları birlikte kontrol edip, kâğıt üzerinde adım adım çalıştırmayı deneyin.

Etkinlik sonunda eğitmen öğrencilerden Örnek Kod Çalıştırma Tablosu'nu incelemelerini ister. Eğitmen değişkenler ve değerleri üzerine konuyu özetler. Bunu yaparken eğitmen her bir adımın açıklaması için aşağıdaki tabloyu kullanılır.

Tablo 13. *Örnek kod çalıştırma tablosu*

Adım	Açıklama
Başla.	Programın Başlangıcı
Oku, (Sayi1,Sayi2)	Kullanıcıdan değerlerin alınması. Burada biz klavyeden alındığını düşünüyoruz. Eğer bir mobil cihaz üzerinde ya da web sitesi üzerinde çalışacaksa farklı teknikler kullanılabilir.
Eğer Sayi1>=Sayi2	Sayi1, Sayi2'den büyük eşit olup olmadığı kontrol ediliyor. Eğer bu koşul doğru ise bir alt satıra devam edeceğiz. Değilse aksi takdirde yazan satıra gideceğiz. Eğer aksi takdirde satırı yok ise bir alt adımdan devam edeceğiz.
3.1) Yaz, Sayi1	Ekrana Sayi1'in değerini yaz.
Aksi takdirde	Sayi1, Sayi2'den büyük değilse
4.1) Yaz, Sayi2	Ekrana Sayi2'nin değerini yaz
Bitir.	Programın Sonu

Eğitmene Öneriler: Eğitmen bu etkinlikte değişken ve değişken değerlerine dikkat çekmelidir. Bunun için aşağıdaki bilgileri de ekleyebilir.

Değişken zamanla değeri değişen veri saklayıcıdır. Örneğin Yaş bilgisi değişkendir. Yaşınızın kaç olduğu ise değerdir. Bilgisayar ortamında değişkenler hafızadan yer ayırmak için kullanılır. Daha sonra içerisine değerler yazarak kullanılır. İçerisindeki değerleri program çalıştığı süre boyunca unutmazlar ve siz değiştirmediğiniz sürece değişmezler. Algoritma ya da program yazarken genellikle daha kısa ve anlamlı isimler veririz. Değişken ve değerlere günlük yaşamdan örnekler verirsek;

Değişken	Değişken İsmi	Değer
Adınız	Ad	“Ahmet”, “Mehmet”, “Ali”,”Ayşe”
Yaşınız	Yas	“12”,”13”,”25”
Boyunuz	Boy	“150 cm””,”140 cm”
Telefon numaranız	Tel	“05555555555”
Sınıfınız	Sinif	“4”,”5”,”6”,”7”

B6. Algoritmayı Test Et!

Süre: 25 dk.

Kazanımlar: K5. Verilen algoritmanın akış diyagramını sözde koda dönüştürür.

K6. Verilen algoritmanın akış diyagramındaki değişkenleri ayırt eder.

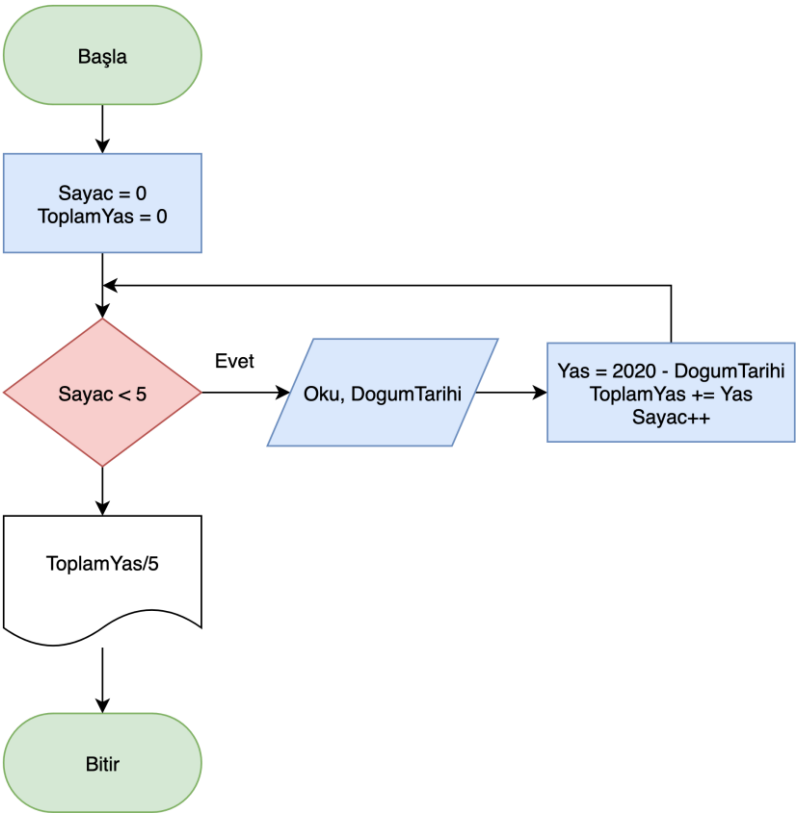
Materyaller: L2. B6. 1. Kod Çalıştırma Tablosu

Hazırlık: Draw.io programı öğrenci bilgisayarlarında kurulu olmalıdır. Öğrencilere kod çalıştırma tablosunun ÖYS üzerinden iletilir.

Uygulama: Eğitmen öğrencilere aşağıdaki görevi verir. Öğrenciler ikişerli gruplar hâlinde çalışır.

> *Görev 1: Sevdiğiniz 5 kişinin doğum tarihlerini girerek yaşlarının ortalamasını bulan bilgisayar programı için akış diyagramı ve sözde kodunu Draw.io programını kullanarak bilgisayarda çiziniz.*

Öğrencilerin çizmesi gereken akış şeması aşağıdaki gibi olmalıdır:



Görevi tamamladıktan sonra eğitmen ikinci göreve gelmeden önce iki grup birleşerek dörderli yeni gruplar oluşturur ve aşağıdaki talimatı verir.

> *Görev 2: Çizdiğiniz akış şemasındaki sözde kodları karşılaştırın ve adım adım çalıştırıp, test etmek için Kod Çalıştırma tablosunu doldurun.*

Hazırlanacak kod çalıştırma tablosu aşağıdaki gibi olmalıdır.

Tablo 14. *Kod çalıştırma tablosu*

Adım	Değişken değerleri	Çıktı
Başla.	Programın Başlangıcı	
Sayac=0, ToplamYas=0	Sayac=0 ToplamYas=0	
Eğer 0<5	evet	
3.1) Oku, DogumTarihi	DogumTarihi=2001	
3.2) Yas = 2020 - 2001	Yas = 19	
3.3) ToplamYas += Yas	ToplamYas = 19	
3.4) Sayac ++	Sayac=1	
Eğer 1 <5	evet	

4.1) Oku, DogumTarihi	DogumTarihi=2003	
4.2) Yas = 2020 - 2003	Yas = 17	
4.3) ToplamYas += Yas	ToplamYas = 36	
4.4) Sayac ++	Sayac=2	
Eğer 2<5	evet	
5.1) Oku, DogumTarihi	DogumTarihi=2005	
5.2) Yas = 2020 - 2005	Yas = 15	
5.3) ToplamYas += Yas	ToplamYas = 51	
5.4) Sayac ++	Sayac=3	
Eğer 3<5	evet	
6.1) Oku, DogumTarihi	DogumTarihi=2007	
6.2) Yas = 2020 - 2007	Yas = 13	
6.3) ToplamYas += Yas	ToplamYas = 64	
6.4) Sayac ++	Sayac=1	
Eğer 4<5	evet	
7.1) Oku, DogumTarihi	DogumTarihi=2004	
7.2) Yas = 2020 - 2004	Yas = 16	
7.3) ToplamYas += Yas	ToplamYas = 80	
7.4) Sayac ++	Sayac=5	
Eğer 5<5	hayır	
Yaz, ToplamYas/5	80/5	16
Bitir.	Programın sonu	

Öğrencilerin hazırladığı tablolar üzerinden beyin fırtınası yoluyla eğitmen adımları aşağıdaki gibi özetler.

Tablo 15. *Örnek kod çalıştırma tablosu*

Adım	Açıklama
Başla.	Programın Başlangıcı
Sayac=0, ToplamYas=0	Burada bize iki adet değişken gerekli. Kaç kişinin bilgisi girildiğini Sayac isimli değişkende tutuyoruz. İstediğimiz sayıların toplamını ise ToplamYas isimli değişkende tutuyoruz.
Sayac<5 Olduğu Sürece	Buradaki koşulumuz 5 değerine ulaşıp ulaşmadığımız. Sayac değişkenimiz 5'e ulaşana kadar devam edeceğiz.
3.1) Oku, DogumTarihi	Klavyeden kişinin doğum yılını al.
3.2) Yas = 2020 - DogumTarihi	Yaşını hesapla
3.3) ToplamYas += Yas	Kişinin yaşını toplam yaş değerine ekliyoruz
3.4) Sayac ++	Elimizdeki Sayac değerini bir arttırıyoruz.
Yaz, ToplamYas/5	Ekrana ortalama yaş değerini (Toplam/5) değerini yazdırıyoruz.
Bitir.	Programın sonu

Eğitmene Öneriler: Eğitmen açıklama sırasında değişkenler ve değerlerine vurgu yapmalıdır. Öğrencilerin grup tablolarını word üzerinde oluşturup ödev olarak ÖYS'de paylaşmaları istenebilir.

C. Kısmi Öğrenme Görevleri

Süre: 60 dk.

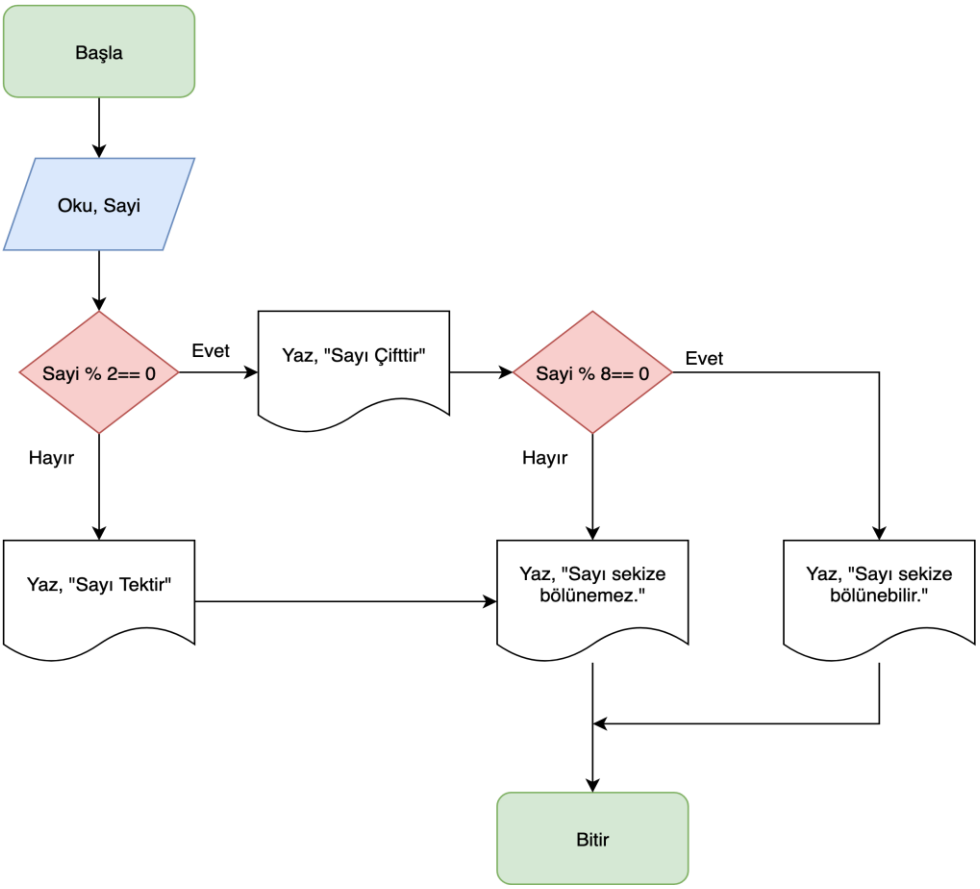
Materyal: L2. C. 1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

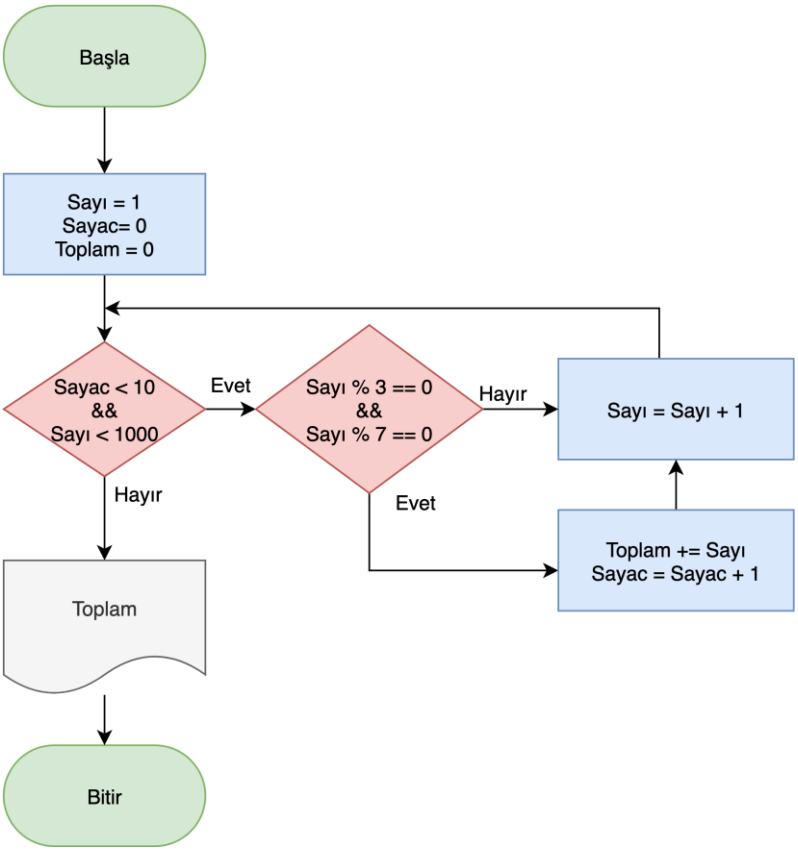
Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Tasarlayıcı 1:



Tasarlayıcı 2:



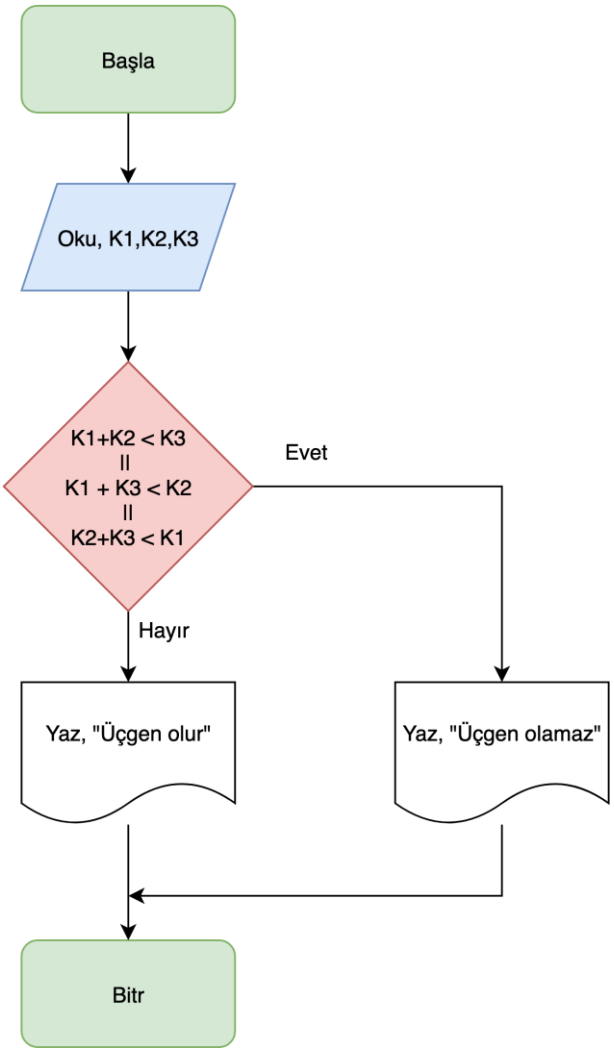
Analizci:

Eğer sayaç 10 değerinden başlatılırsa, girilen sayı 10'dan küçük olursa program doğrudan sonlanacaktır. 10'dan büyük olduğu durumlarda ise 10 ila girilen sayı arasındaki tüm sayıları çarpacaktır.

Denetleyici:

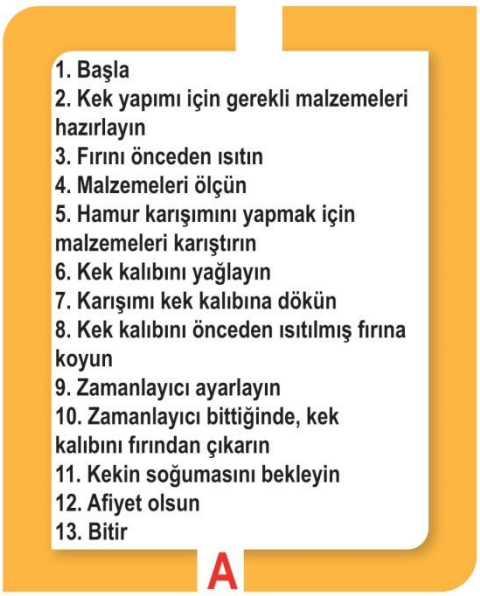
40320

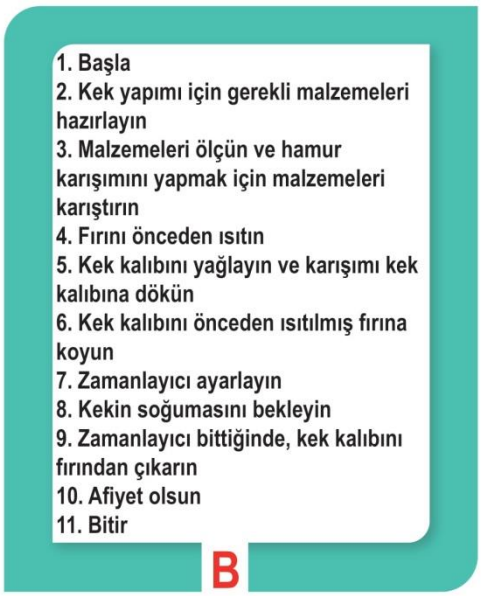
Tasarlayıcı 3:

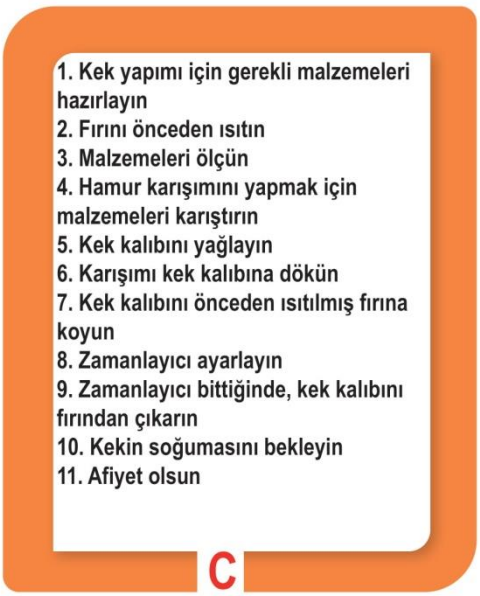


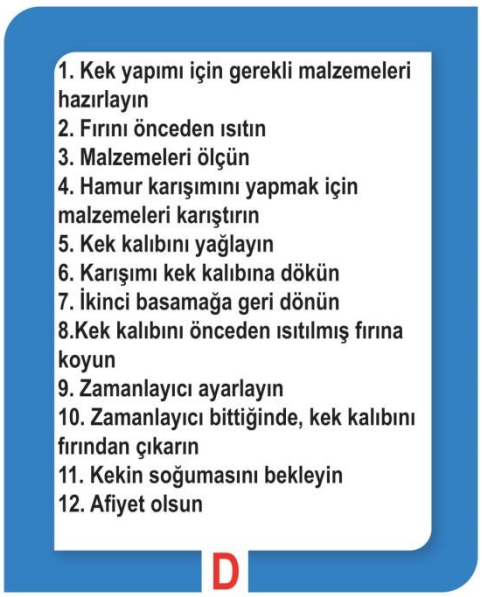
Hafta 2. Ders Materyalleri

L2. B1. 2. Doğru algoritma hangisi?





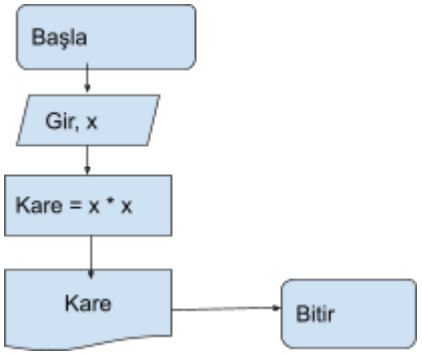
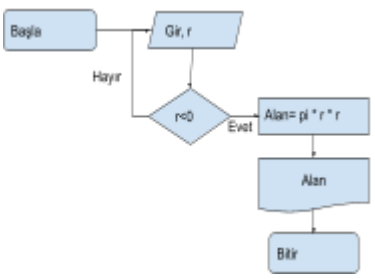


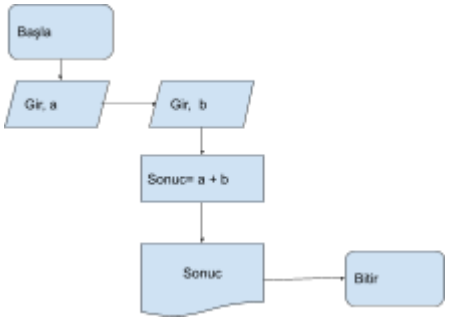


L2. B2. 1. Algoritmaları Eşleştirelim Çalışma Kâğıdı

Aşağıda metin şeklinde yazımı, sözde kod ve akış diyagramı olmak üzere bir probleme üç farklı şekilde algoritma yazılmıştır. Hangi algoritma hangi probleme aittir, lütfen eşleştiriniz. Eşleştirme için aşağıdaki kısaltmalardan uygun olanı kutucuk altına yazınız.

Problemler	**Algoritma İfade Şekli**
P1. Çemberin alanını hesaplama	MY: Metinsel Yazım
P2. İki sayıyı toplama	SK: Sözde Kod
P3. Bir sayının karesini hesaplama	AD: Akış Diyagramı

<table>
<tr>
<td>1. Başla.
2. Birinci sayıyı gir (a)
3. İkinci sayıyı gir (b)
4. İşlemi yap. (sonuc = a+b)
5. Ekrana yaz (sonuc)
6. Bitir.</td>
<td> 
</td>
<td>1. Başla.
2. Sayıyı (x) gir
3. Sayının karesini hesapla (Kare=x*x işlemini yap)
4. Sonucu (Kare) yaz
5. Bitir.</td>
</tr>
<tr>
<td>Problem:
Algoritma İfadesi:</td>
<td>Problem:
Algoritma İfadesi:</td>
<td>Problem:
Algoritma İfadesi:</td>
</tr>
<tr>
<td> 
</td>
<td>1. Başla.
2. Yarıçap gir (r).
3. Eğer yarıçap sıfırdan küçükse ikinci adıma git.
4. Alanı hesapla (alan= π ×r2)
5. Sonucu ekrana yazdır.
6. Bitir.</td>
<td>1. Başla.
2. Gir, a
3. Gir, b
4. Sonuc = a+b
5. Yaz, Sonuc
6. Bitir.</td>
</tr>
<tr>
<td>Problem:
Algoritma İfadesi:</td>
<td>Problem:
Algoritma İfadesi:</td>
<td>Problem:
Algoritma İfadesi:</td>
</tr>
</table>

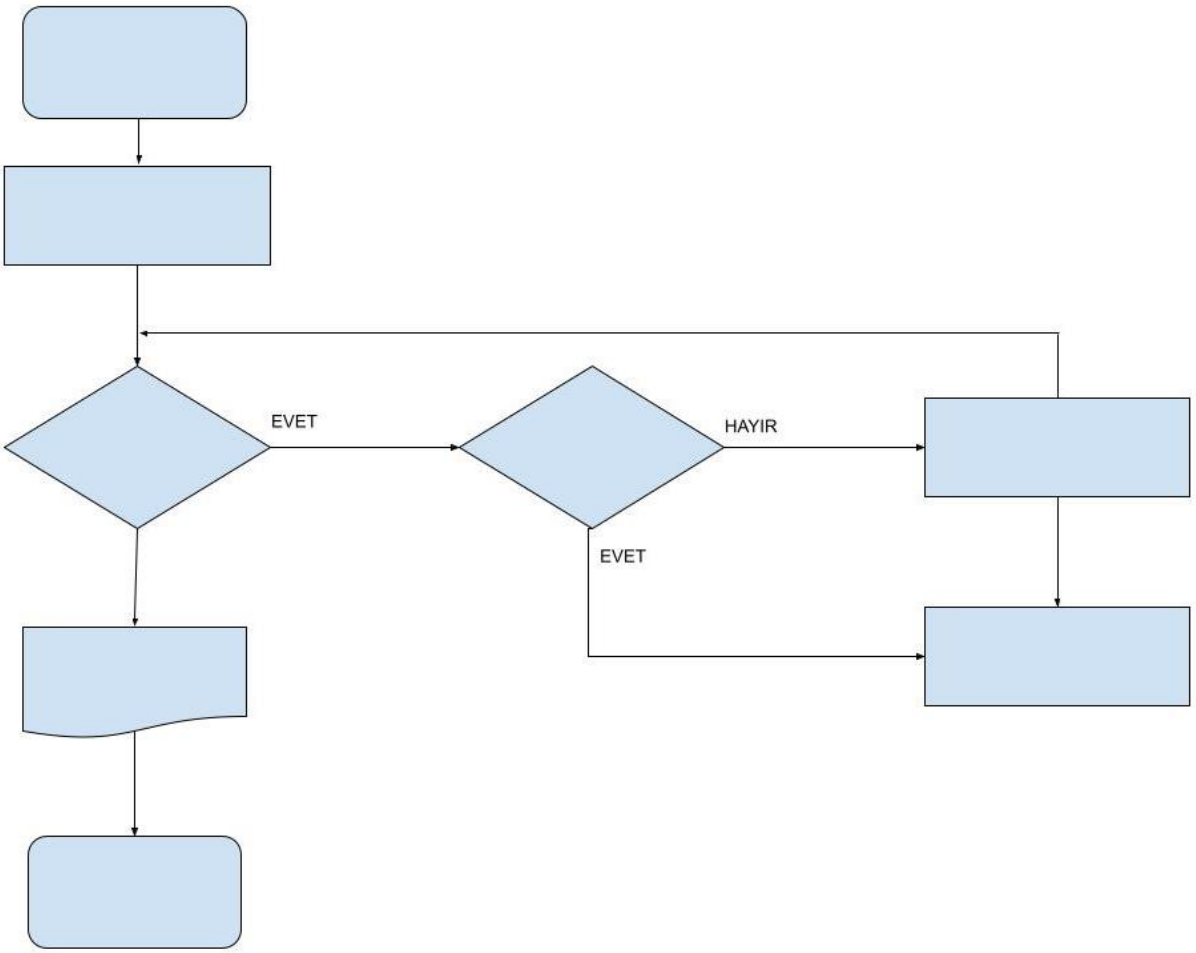
1. Başla. 2. Gir, r 3. Eğer r < 0 ikinci adıma git. 4. alan= π ×r×r 5. Yaz, Sonuc 6. Bitir		1. Başla 2. Gir, x 3. Kare = x*x 4. Yaz, Kare 5. Bitir
Problem: *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*

L2. B3. 1. Çiftleri Toplama Algoritması

Görev: Aşağıdaki problemin çözümüne yönelik sözde kodu verilen algoritmayı doğru şekilde sıralayın ve akış diyagramındaki boşluklara yazın.

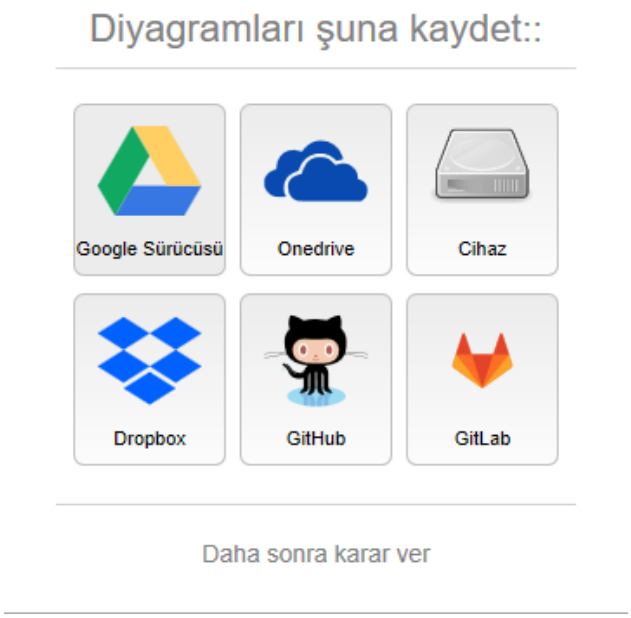
Problem: 1 ile 100 arasındaki çift sayıların toplamını yazdırma

Başla.	Sayi = 1, Toplam = 0	Sayi % 2 ==0	Sayi < 100
Yaz, Toplam	Toplam = Toplam + Sayi	Bitir.	Sayi = Sayi + 1


EVET
HAYIR
EVET

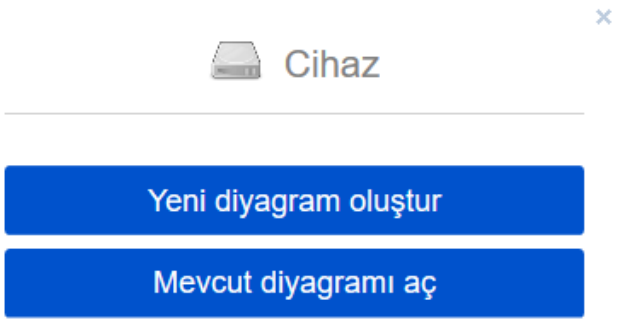
L2. B3. 2. Draw.io program kullanım kılavuzu

Akış diyagramını bilgisayar ortamında çizmek için draw.io isimli websitesi kullanacağız. Ücretsiz olan bu uygulama sayesinde çizdiğimiz şekilleri kaydedebilir, istediğimiz kişilerle paylaşabiliriz. Draw.io isimli uygulamaya ilk giriş yaptığımızda aşağıdaki görüntü karşımıza gelecektir. Çizdiğimiz diyagramı nereye kaydetmek istediğimizi belirtiyoruz. Cihaz seçeneğini tıklayarak, şimdilik kendi bilgisayarımızda kaydedelim.



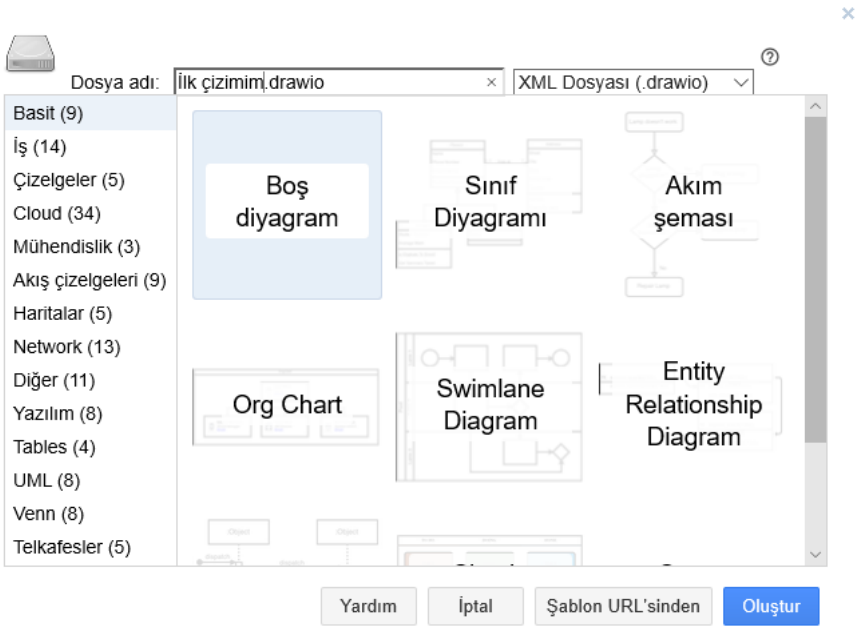
Resim 10. *Draw.io giriş ekranı*

Ardından, yeni diyagram oluştur butonuna tıklayarak boş bir sayfa oluşturalım.



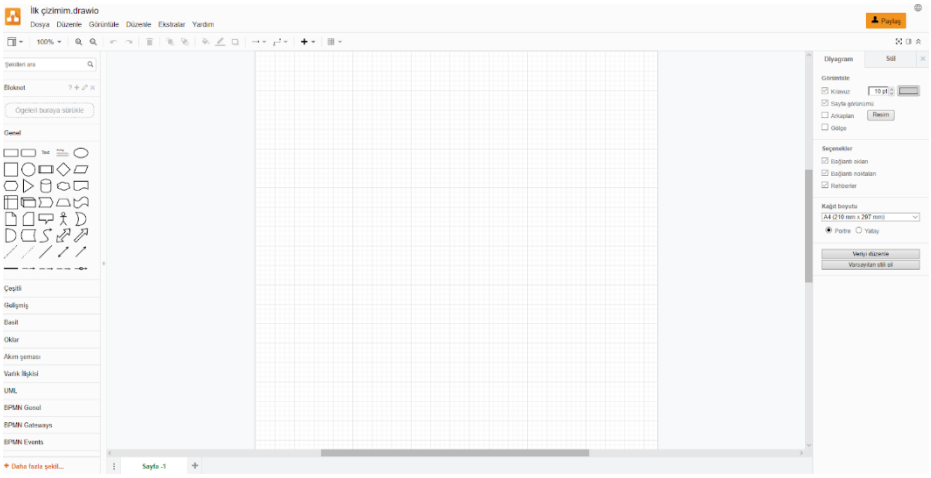
Resim 11. *Draw.io yeni diyagram oluşturma ekranı*

Draw.io ile basitten profesyonel seviyeye kadar birçok seviyede çizim yapılabilir. Biz şimdilik, boş diyagram seçeneğini seçerek bir isim verebiliriz.



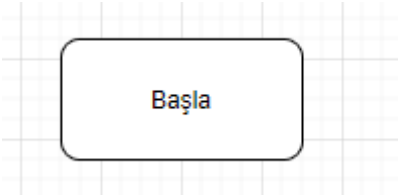
Resim 12. *Draw.io yeni diyagram tipi seçme ekranı*

Karşımıza aşağıdaki çizim ekranı gelecektir.



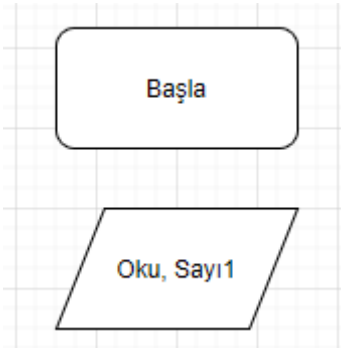
Resim 13. *Draw.io çizim ekranı*

Bu aşamada **Genel** isimli araç kutusunda öğrendiğimiz bloklar mevcuttur. Eklemek istediğimiz bloğu çizim alanına sürekle bırak yapıyoruz. Ardından blok üzerine çift tıklayarak içerisine yazı ekliyoruz.



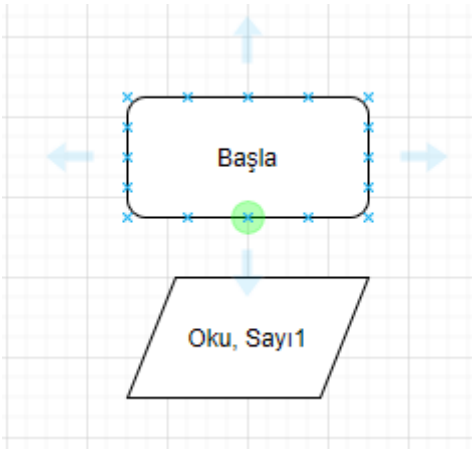
Resim 14. *Genel bloğu*

Ardından bir tane de paralel kenar ekleyelim.



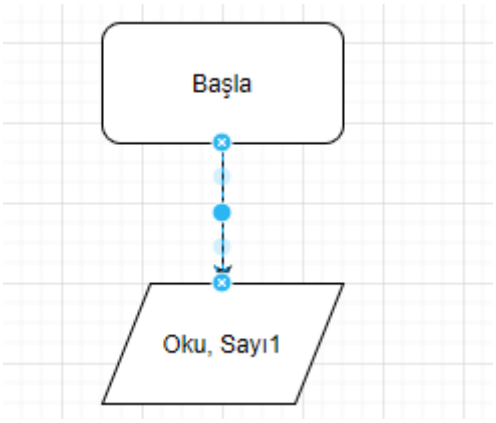
Resim 15. *Paralel kenar bloğu*

İki bloğu birbirine bağlamak için ilk bloğun üzerine fareyi getiriyoruz. Fare üzerine geldiğinde, çizgi çizebileceğimiz noktaları görüyoruz.



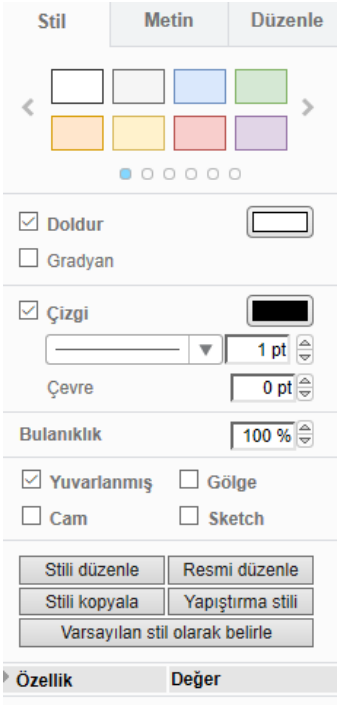
Resim 16. *Blokları bağlama öncesi*

Ardından yine fare yardımıyla, hangi sonraki blok ile bağlantıyı tamamlıyoruz.



Resim 17. *Blokları bağlama sonrası*

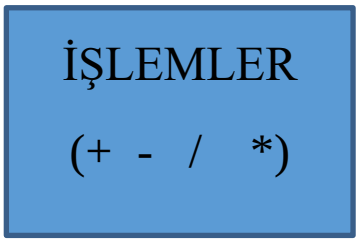
Eğer bloğun renklerini değiştirmek istersek sağ paneli kullanabiliriz.



Resim 18. *Blok özellikleri ekranı*

L2. B3. 3. Akış Diyagramları Bilgi Afişi

Başla ve Bitir adımları yandaki şekildeki gibi gösterilir. Algoritmamızı meydana getirecek olan diğer bütün adımlar bu iki şekil arasına eklenir.

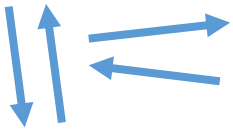


Toplama, çıkarma, aritmetik işlemlerin yapılması gerektiği durumlarda dikdörtgen kullanılır. Örneğin; üçgenin alanını hesaplama

Kullanıcı girişlerini şekilsel olarak göstermek için paralelkenar kullanılır. Örneğin; Kullanıcıdan istenen ad, soyad, yaş vb.)

Yukarıda kullanıcıdan alınan bilgileri ekrana yazdırılması gerektiğinde yandaki şekil kullanılır.

Algoritma akışında işlemler tek yönlü iken karar yapılarında iki farklı ok çıkabilir ve bazı işlemlerin tekrarlanmasını sağlayabilir. Bu



Yandaki oklar algoritmanın akış yönünü bize belirtir.

L2. B3. 4. Algoritma Terimleri Görev Kartları

Tanımlayıcı

Algoritmadaki değişkenleri, sabitleri, kayıt alanlarını ve özel bilgi tiplerini adlandırmak için programcı tarafından oluşturulan kelimelerdir. Tanımlayıcıların yerini tuttukları ifadelere çağrışım yapacak şekilde adlandırılmaları algoritmanın anlaşılması açısından önemlidir. Tanımlayıcı isimlerinde İngiliz alfabesindeki "A-Z" ve "a-z" arası harfler, "0-9" arası rakamlar, sembollerden alt çizgi "_" kullanılabilir. Bununla birlikte tanımlayıcı ismi, rakamla başlayamaz veya sadece rakamlardan oluşamaz.

Görev

Çiftleri toplama algoritmasındaki tanımlayıcı kuralları nelerdir?

Değişken

Algoritmanın her çalıştırılmasında farklı değerler alan bellek alanlarıdır. Değişken isimlendirilmeleri, tanımlayıcı kurallarına uygun biçimde yapılmalıdır. Örneğin bir ismin aktarıldığı değişken "ad" olarak, bir isim ve soy ismin aktarıldığı değişken "adsoyad", ev adresinin aktarıldığı değişken "evadres" ve iş adresinin aktarıldığı değişken "isadres" olarak tanımlanabilir. Burada özellikle İngiliz alfabesi kullanılır.

Görev

Çiftleri toplama algoritmasındaki değişkenleri bulup, isimlerini değiştirmeyi deneyin.

Sabit

Algoritmadaki değeri değişmeyen ifadelerdir. Algoritma boyunca kolay takip edilebilmesi için kullanılmaktadır. Değişkenler gibi isimlendirme kurallarına uygun olarak oluşturulmalıdırlar.

Görev

Çiftleri toplama algoritmasındaki sabitleri bulun.

Atama / Aktarma

Bir değişkene değer atamak için gerçekleştirilen işlemdir. Algoritma adımlarında sağdaki değeri, soldaki değişkene atamak için kullanılan bir işlemdir.

Görev

Çiftleri toplama algoritmasındaki değişken atama ifadelerini bulun ve bu ifadeyi farklı türde kullanmayı deneyin. Aradaki değişimi tartışın.

Sayaç

Algoritma tasarımlarında bazı işlemlerin belirli sayıda yaptırılması ve bu süreçte oluşan değerlerin sayılması gerekebilir. Bu amaçla kullanılan sayma işlemlerine sayaç denir.

Görev

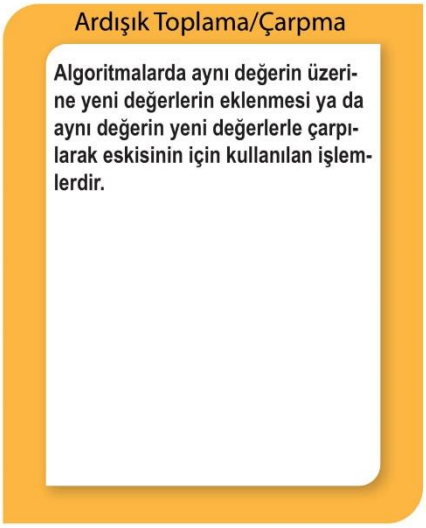
Çiftleri toplama algoritmasına yeni bir sayaç eklemek için problemi değiştirin.

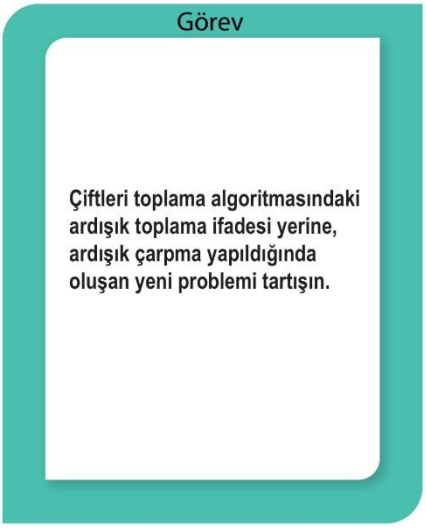
Döngü

Algoritma tasarımlarında bazı işlemlerin ardışık olarak belirli sayıda tekrar etmesi gerekmektedir. Döngüler, bu işlem bloklarını verilen sayıda gerçekleştiren işlem akışlarını sağlamaktadır. Örneğin 1 ile 100 arasındaki çift sayıların toplamını hesaplayan programda T=2+4+6 ….+100 hesabını teker teker yapmak yerine, 1 ile 100 arasında ikişer artan bir döngü kullanmak ve döngü değişkenini toplam hesabında kullanmak daha uygun olacaktır.

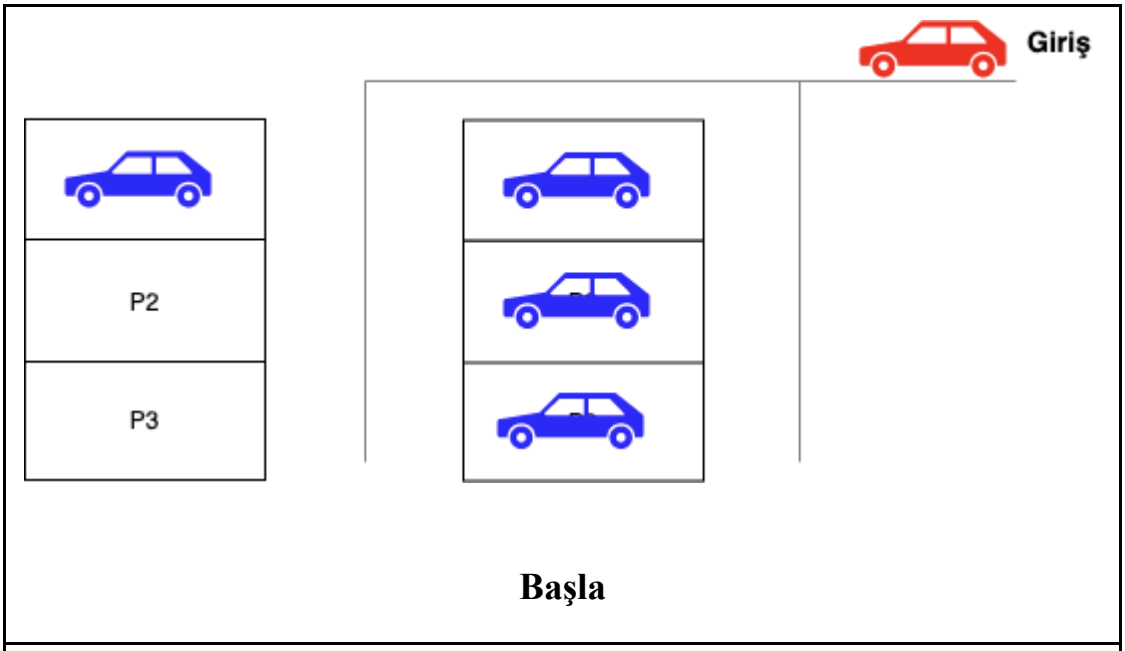
Görev

Çiftleri toplama algoritmasındaki döngüyü bulup, tartışın.

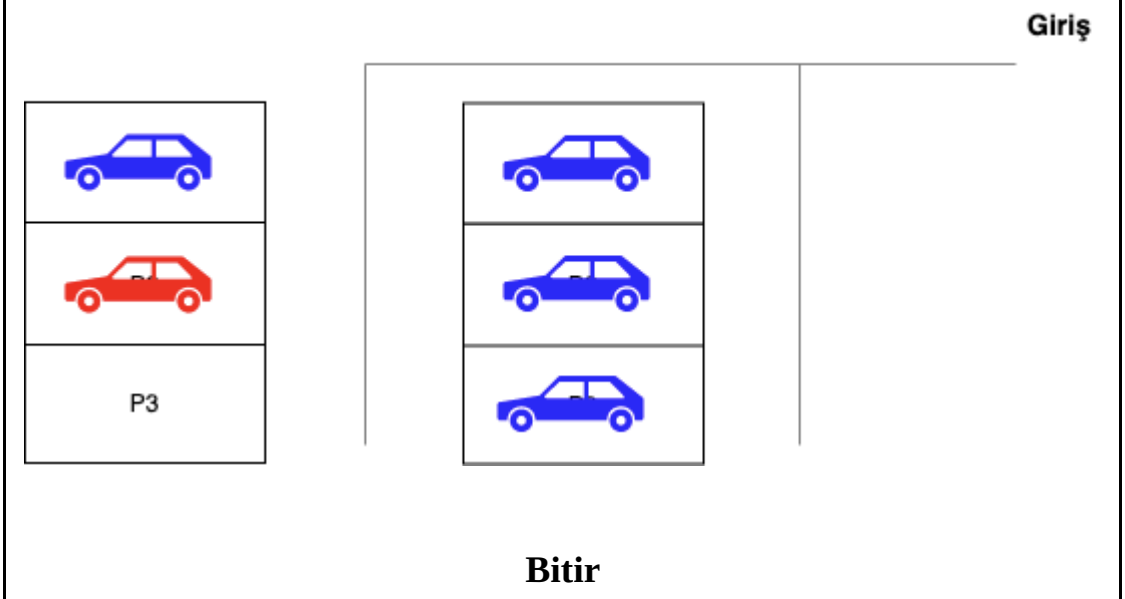

Ardışık Toplama/Çarpma
Algoritmalarda aynı değerin üzerine yeni değerlerin eklenmesi ya da aynı değerin yeni değerlerle çarpılarak eskisinin için kullanılan işlemlerdir.


Görev
Çiftleri toplama algoritmasındaki ardışık toplama ifadesi yerine, ardışık çarpma yapıldığında oluşan yeni problemi tartışın.

L2. B4. 1. Otomatik Park Etme Algoritması



Başla



Bitir

Şekilde görülen kırmızı arabanın otomatik olarak uygun alana park etme algoritması:

1. Başla.
2. Sonraki sokağa kadar ilerle.
3. Eğer sokakta yer yoksa 2. adıma git.
4. Sokağa gir.
5. İlk uygun yere park et.
6. Bitir.

L2. B5. 1. Örnek Kod Çalıştırma Tablosu

Örnek Olay: Arkadaşınız aklından iki sayı tutmuştur ve sizden bu iki sayıdan büyük olanı bulmanızı istiyor. Bunun için bir algoritma yazmak istiyorsunuz.

Tablo 16. *Örnek kod çalıştırma tablosu*

Adım	Değişken değerleri	Ekran Çıktısı
Başla.		
Oku, (Sayi1,Sayi2)	Sayi1=19, Sayi2=15	
Eğer Sayi1>Sayi2	Doğru, Evet	
3.1) Yaz, Sayi1		19
Aksi takdirde		
4.1) Yaz, Sayi2		
Bitir.		

Klavyeden Sayi1 olarak 19, Sayi2 olarak 15 girdiğimizi düşünelim. 3.Adımda 19 > 15 koşulu kontrol ediliyor. Bu koşul doğru olduğu için, 3.1. Adıma gidiyoruz. Orada da ekrana 19 değerini yazdırıyoruz. 4 ve 4.1. adımlar hiç çalıştırılmadan 5. Adıma giderek programımızı sonlandırıyoruz.

L2. B6. 1. Kod Çalıştırma Tablosu

Algoritmanın sözde kodlarını çalıştırmak için aşağıdaki tabloyu kullanabilirsiniz. Algoritmanın adım sayısına göre satırları artırabilirsiniz.

Tablo 17. *Örnek kod çalıştırma tablosu*

Adım	Değişken değerleri	Ekran Çıktısı

L2. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 3. C++ Dilinde Değişken ve Veri Tipleri

Kazanımlar

- K1. Değişken isimlendirirken uyulması gereken kuralları uygular.
- K2. Bir algoritmadaki sabitleri ayırt eder.
- K3. Veri tipine uygun veri tanımlar.
- K4. Veri tipi dönüşüm işlemlerini yapar.
- K5. Kaçış dizgelerini kullanır.
- K6. C++ temel giriş/çıkış akışlarını analiz eder.
- K7. Operatörleri bir algoritma içinde kullanarak sonucunu yordar.
- K8. Bit işlemleri ile verileri hesaplar.

Amaç

Haftanın amacı, C++ değişken tanımlama adımlarının veri tipleri ile birlikte öğrenilerek sabit tanımlama ve veri tip dönüşüm işlemlerinin nasıl gerçekleştirildiğini öğrenmektir. Ayrıca C++ temel giriş/çıkış akışlarının nasıl kullanıldığını, C++ işleçlerin ve bit işlemlerin örnekler üzerinde nasıl çalıştığını görmelerini sağlamaktadır.

Önerilen Ders Akışı

A. Giriş: Ev Sahibi ve Kiracı (10 dk.)
B. Öğrenme Görevleri
   B1. Kuralları Belirle! (20 dk.)
   B2. Sabitleri Ayıralım (15 dk.)
      ❏ Ders Arası (5 dk.)
      ❏ Motivasyon Oyunu (10 dk.)
   B3. Uzman Sensin (25 dk.)
   B4. Veri Tiplerini Ayırt Et! (20 dk.)
   B5. Kaçmaya Hazırlan (15 dk.)
      ❏ Ders Arası (5 dk.)
   B6. Çıktıları Karşılaştır (10 dk.)
   B7. Operatörlerle Yüzleş (20 dk.)
   B8. Listeyi Dolduralım (20 dk.)
      ❏ Ders Arası (5 dk.)
      ❏ Motivasyon Oyunu (10 dk.)
C. Kısmi Öğrenme Görevleri (50 dk.)

A. Giriş: Ev Sahibi ve Kiracı

Süre: 10 dk.

Uygulama: Oyuncuların 3'er kişilik gruplar oluşturmaları istenir. Oluşturulan bu gruplarda, 2 kişinin ellerini havada birleşmesini, 1 kişinin de aralarında durması istenir. Kenarlarda duran kişilerin "Ev sahibi", ortada duran kişinin "Kiracı" olduğu söylenir. Ortada bir ebe olur. Ebe, "Ev sahibi" veya "Kiracı" der. Ebe, "ev sahipleri" dediğinde; kenardaki kişiler diğer ev sahipleriyle yer değiştirir, bu sırada kiracılar aynı yerlerinde kalmalıdır. Ebe, "kiracılar" dediğinde; ortadaki kişiler diğer ortadaki kişilerle yer değişir, bu sırada ev sahipleri aynı yerlerinde kalmaya devam eder. Ebe de bu değişiklikler sırasında kendine bir yer bulmaya çalışır. Ortada kalan kişi yeni ebe olur. Eğer ortadaki kişi "Kentsel Dönüşüm" derse, bütün herkes yer değiştirir ve yeni 3 kişilik gruplar oluşturur (Kiracılar ev sahibi, ev sahipleri kiracı olabilir).

B. Öğrenme Görevleri

B1. Kuralları Belirle!

Süre: 20 dk.

Kazanımlar: K1. Değişken isimlendirirken uyulması gereken kuralları uygular.

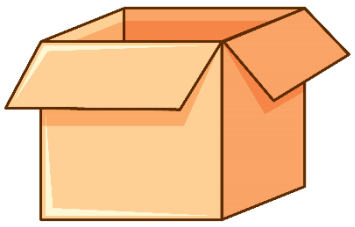
Materyaller: L3. B1. 1. Hatalı Değişkenler

L3. B1. 2. Değişken İsmi Olamazlar

Hazırlık: Eğitmen birinci materyalin çıktısını alıp, tek tek keserek materyali beş gruba ayırır. Her grupta dörder tane kesilmiş kart ve grup görev kartı bulunmalıdır. İkinci materyal ise beş gruba bir adet dağıtılmak üzere çıktı alınır.

Uygulama: Eğitmen önceki etkinliklerde değişken kavramını öğrendiklerini öğrencilere aşağıdaki hatırlatmayı kullanarak açıklar.

> *"C++ programlarında veri değerlerinin bilgisayarın belleğinde saklanabildiği saklayıcı yapılara (aşağıdaki kutu gibi düşünebiliriz) değişken adı verilir. Saklanan değere değişkenin adı kullanılarak ulaşılabilir."*



Resim 19. *Saklayıcı yapı kutu gösterimi*

> *Bir değişkenin bir programda kullanılabilmesi için önce tanımlanması gerekir. Bir değişken tanımladığınızda, tür belirtilir ve uygun miktarda bellek ayrılır. Bu bellek alanı, değişkenin adı referans alınarak ele alınmaktadır. Aynı veri türünden birden*

fazla değişken adı oluşturulmak istenirse bunlar virgülle ayrılacak şekilde aşağıdaki gibi yazılabilir:

veri-tipi degisken_adi;

veri-tipi deg_adi1, deg_adi2, deg_adi3;

Bu etkinlikte ise değişkenlere isim verme üzerinde duracaklarını ifade eder. Bunun için öğrenciler dörderli gruplara ayrılır. Eğitmen tek tek kesilip beş gruba ayrılmış "Hatalı Değişkenler' materyalini öğrencilere dağıtır. Daha sonra gruplara:

> *"Elinizde C++ dilinde değişken isimlerine ilişkin örnekler var ve grup görev kartı var. Görev kartında hatalı ya da doğru yazılan değişken isminden bir kurala erişmeniz bekleniyor. Bunun için elinizdeki değişken isimleri arasındaki benzerlikleri keşfetmeye çalışın. Bu benzerlikler değişken ismi yazma kuralını bize verecektir. Sizden beklenen bu benzerlikten yararlanıp değişkene isim verme kuralını keşfetmektir. Her grup yalnızca bir kurala erişmelidir."*

talimatı verilir. Her grup bir tane değişken belirleme kuralına erişmelidir. Bu kurala erişmek için eğitmen her öğrencinin kartında yazan değişken isimlerini grup üyelerininki ile karşılaştırmalarını ister. Bu şekilde kartlar arası ortak noktaların değişken belirleme kuralına işaret edeceği ipucunu verir. Gruplar beş dakika kartlar üzerinde çalışırlar. Eğitmen sırayla grupların belirledikleri kuralı sesli şekilde okutur. Eğitmen tüm kurallar tamamlanınca değişken ismi olmayan yasaklı kelimelerin olduğu konusunda bir hatırlatma yapar. Bu kelimeler "Değişken İsmi Olamazlar" materyali ile Öğrenme Yönetim Sistemi üzerinden öğrencilerle paylaşılır. Bununla birlikte eğitmen aşağıdaki açıklamayı kullanarak öğrencilere uyarıda bulunur.

> *"Anahtar kelimeleri yanlışlıkla değişken adı olarak kullanmayın. Anahtar kelimeleri bu şekilde kullanmak öngörülemeyen sonuçlara neden olabilir ve birçok sıkıntıya sebep olabilir. Örneğin, "short" bir sayıyı temsil etmek için ayrılmış bir kelimedir ve bu kelimeyi bir değişken adı olarak kullanmaya çalışırsanız, program oluşturulduğunda derleyici size bir hata verecektir."*

Etkinlik sonunda eğitmen değişkenlere değer atama türlerinden kısaca bahseder.

> *"Bir değişkene bir değer atandığında "ilk değeri ataması" ifadesi kullanılır. İsteğe bağlı olarak, bu atama değişken tanımlanırken ya da daha sonra kullanımda yapılabilir. Herhangi bir değişkende saklanan değerin içeriğini görüntülemek için "cout" fonksiyonu kullanılır."*

Eğitmen aşağıdaki kodları bilgisayarından tüm öğrencilerin görebileceği şekilde ekrana yansıtır ve dördüncü satıra dikkat etmelerini ister. Öğrenciler bu kodları bilgisayarlarında yazdıktan sonra çıktıları karşılaştırır.

İlk değer atamasız değişken tanımlama

```
1.  #include <iostream>
2.  int main()
3.  {
4.      int sayi;
5.      sayi = 5;
6.      std::cout << sayi;
7.      return 0;
8.  }
```

İlk değer atamalı değişken tanımlama

```
1.  #include <iostream>
2.  int main()
3.  {
4.      int sayi = 5;
5.      std::cout << sayi;
6.      return 0;
7.  }
```

Eğitmene Öneriler: Sınıfta erişilmesi gereken temel kurallar aşağıdaki gibi olmalıdır.

1. İsim alt çizgi ile başlayabilir ancak sayı ile başlayamaz. Diğer her karakter bir harf, alt çizgi veya sayı olabilir.
2. İsim bir harf ile başlayabilir ancak sayı ile başlayamaz. Diğer her karakter bir harf, alt çizgi veya sayı olabilir.
3. Değişken adları C++' da büyük/küçük harfe duyarlıdır; bu nedenle "numara", "Numara" ve "NUMARA" gibi değişkenler üç ayrı değişken olarak ele alınır.
4. İsim içerisinde Türkçe karakter kullanılmaz.
5. İsim yazarken boşluk bırakılmaz.

"Değişken İsmi Olamazlar" materyali Öğrenme Yönetim Sistemi üzerinden öğrencilerle paylaşılırsa, ihtiyaç anında öğrencilerin açıp bakma imkânı olur. Eğitmen ayrıca öğrencilere aşağıdaki ipucunu vererek önerilerde bulunabilir.

> ***İPUCU:*** *C++ programlamada, değişkenlerin adları için küçük harfler kullanmak standart bir uygulamadır. Bazı değişkenlerin adları belirli bir kullanımla ilişkilendirilme eğilimindedir. Örnek olarak;*
>
> *c, ch: karakterler için kullanılır.*
>
> *i, j, k, l, m, n: tam sayılar ve indisler için kullanılır.*
>
> *x, y, z: tam ve virgüllü sayılar için kullanılır.*
>
> *Ayrıca programlarınızın okunabilirliğini artırmak için, ogrenci_yas, sinav_notu vb. gibi daha uzun ve daha açıklayıcı adlar seçebilirsiniz.*

B2. Sabitleri Ayıralım!

Süre: 15 dk.

Kazanımlar: K2. Bir algoritmadaki sabitleri ayırt eder.

Materyaller: L3. B2. 1. Sabitler için Örnek Kod

Hazırlık: Materyal ÖYS üzerinden öğrenci erişimine açılır. Öğrenciler dörderli gruplar hâlinde oturmaktadır.

Uygulama: Eğitmen değişken ismi tanımlamalarının ardından sabitler için aşağıdaki kısa bilgi ile konuya giriş yapar.

> *"Bir programın yürütülmesi sırasında içeriği hiç değişmeyecek olan veriler, bir değişken yerine sabit bir tanımlama ile saklanmalıdır."*

Eğitmen öğrencilere örnek koddaki sabitleri tahmin edip yuvarlak içine almalarını ister. Bunu yaparken aşağıdaki üç temel kurala dikkat etmeleri gerektiğini belirtir:

1. "const" anahtar sözcüğünü kullanarak tanımlarız.
2. Sabit adları, değişken adlarından ayırt etmek için büyük harfle yazılır.
3. Sabitler her zaman tanımlama sırasında değerlerini almalıdır.

Eğitmen öğrencilerin sabitleri ayırması için gruplara 3 dk. süre tanır. Süre sonunda tahminleri alır ve konuyu aşağıdaki bilgilerden yararlanarak özetler:

```
const veriTipi SABITADI = deger;
```

```
const double PI = 3.14159265;
```

Sabitleri kullanmak çok fazla avantaj sağlar. Örneğin matematikteki Pi sayısını her kullanmamız gerektiğinde, program boyunca 3.14 gibi bir sayı yazdığımızı varsayalım. Ardından uygulamanızın daha yüksek bir hassasiyet gerektirdiğini fark edersek; her 3.14 sayısını, 3.14159265 gibi daha yüksek hassasiyet değeri ile değiştirmemiz gerekir. Bu değişikliği uygun bir şekilde hatasız olarak gerçekleştirmek de zaman alacaktır. Bunun yerine, Pi sayısını 3.14 değeri ile sabit olarak tanımlamış olsaydık, daha sonra hassasiyeti artırmamız gerektiğinde sadece en baştaki tanımlamadaki değeri değiştirmemiz yeterli olacaktır.

B3. Uzman Sensin

Süre: 25 dk.

Kazanımlar: K3. Veri tipine uygun veri tanımlar.

Materyaller: L3. B3. 1. Veri Tipleri Çalışma Kâğıtları

L3. B3. 2. Değişken Atama Kodları

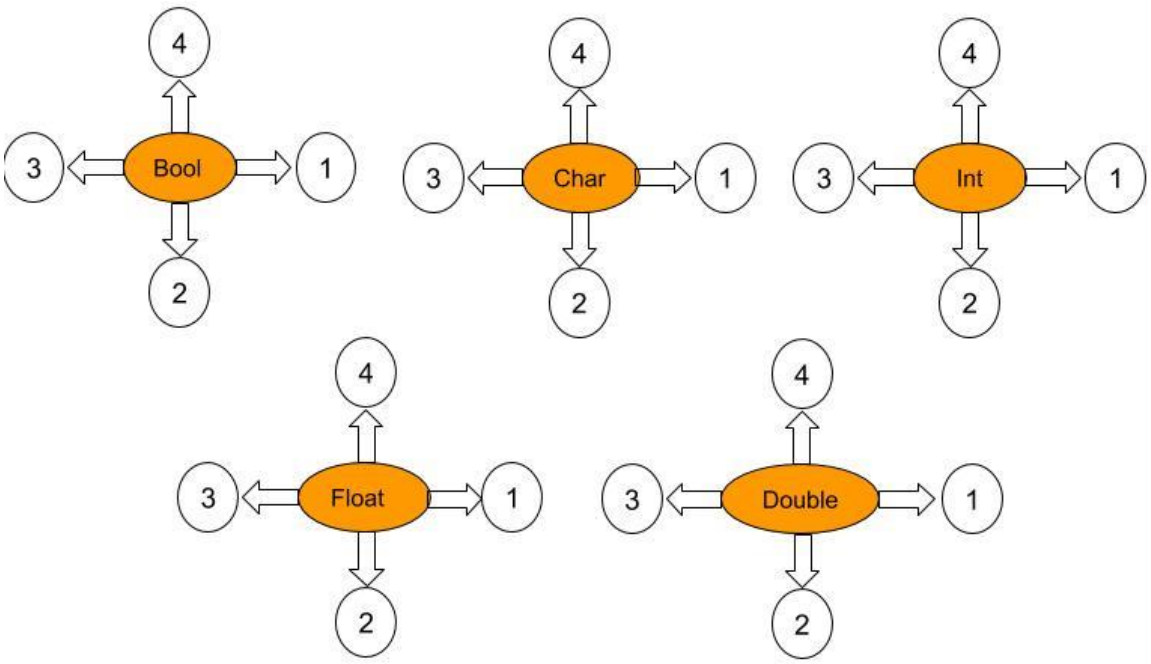
Hazırlık: Eğitmen veri tipleri çalışma kâğıtlarını ve değişken atama kodlarının çıktısını alır, keserek bunları beş gruba ayırır.

Uygulama: Öğrenciler dörderli beş gruba ayrılır. Eğitmen gruplara beş parçaya ayırdığı materyalleri dağıtır. Bu materyal ile her bir grup bir veri tipini temsil etmektedir. Eğitmen gruplardan grup üyelerinin her birinin veri tipi adi_no şeklinde kodlanmalarını ister. Örneğin Bool ekip üyeleri, Bool_1, Bool_2…. şeklinde dört kişiden oluşmaktadır. Eğitmen

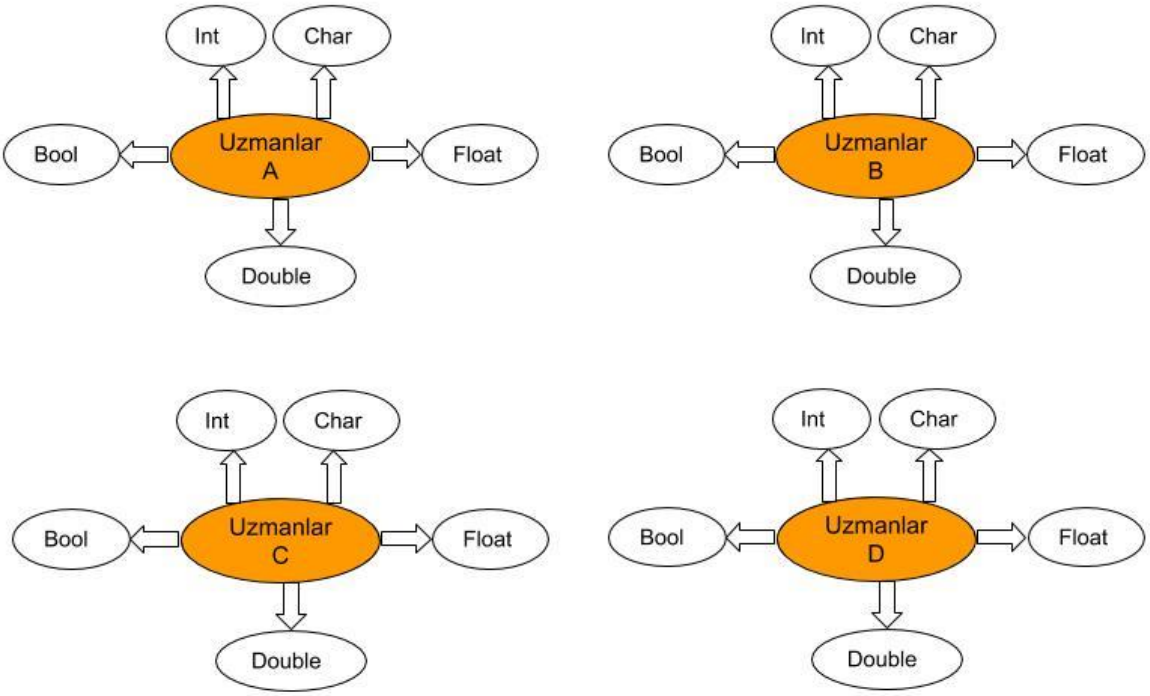
öğrencilerden temsil ettikleri veri tipinin özellikleri hakkında çalışma kâğıtlarını incelemeleri ve örnekleri grup içinde beş dakika içinde tartışmalarını ister. Eğitmen materyalin yanı sıra, internet üzerinden de araştırma yapabileceklerini de belirtebilir. Bu tartışmada her bir grup üyesinin temsil ettiği veri tipini başkasına anlatacak seviyede uzman olmaları istenir. Beş dakika ekipler kendi içlerinde çalıştıktan sonra, aynı numara ile kodlanmış öğrenciler bir araya gelir. Örneğin 1 numaralı grup üyeleri birleşerek beş kişiden oluşan uzmanlar grubu elde edilir.

Uzman gruplarının sayısı, beşer kişiden dört gruptur. Bu uzmanlar grubunda artık tüm veri tiplerinden birer kişi bulunmaktadır. Eğitmen bu yeni gruplardaki her bir bireyin temsil ettiği ve uzmanı olduğu veri tipini diğerine açıklamalarını ister. Ayrıca uzman öğrenciler temsil ettiği veri tipine ilişkin değişken atama kodlarını diğer grup üyelerine uygulatarak açıklayacaktır. Ancak kodları çalıştırmadan önce program çıktısında ne göreceklerini tahmin etmelerini ister. Örneğin bir numaralı uzman grubundaki Bool temsilcisi, diğer uzmanlara Bool değişken tanımlama kodlarını bilgisayar ortamında uygulayarak anlatır. Bu şekilde tüm uzmanlar birbirlerine değişken tanımlama kodlarını anlatarak ortak bir veri dosyası hazırlayacaktır. Artık bu kod dosyasında tüm veri tiplerine ilişkin değişken tanımlama programları yer alır. Eğitmen grup çalışmaları sırasında öğrencileri takip eder, desteğe ihtiyaç duyulduğunda geri bildirimlerde bulunur. Gruplar 10 dk. kadar birbirlerine açıklamalarda bulunur.

Eğitmene Öneriler: Eğitmen grupların dağılıp birleşmeleri için çan ya da zil sesi kullanabilir. Sınıf düzeni aşağıdaki gibi olmalıdır.



Oturma Düzeni 1



Oturma Düzeni 2

Resim 20. *Oturma düzenleri*

Öğrenciler uzman gruplarına dağılmadan önce öğrencilerin grup içinde birlikte anlamlandıramadıkları noktaları eğitmenlere danışmaları istenebilir. Ayrıca eğitmen uzman gruplarının her birinin birer öğretmen olarak rol aldıkları konusunda öğrencilere hatırlatıcı ve teşvik edici geri bildirimlerde bulunmalıdır. Örneğin "Bu işi çok iyi başardın; Konuyu diğer arkadaşlarına senin öğretiyor olman, benim açıklama yapmamdan çok daha etkili olacaktır" vb. gibi.

B4. Veri Tiplerini Ayırt Et!

Süre: 20 dk.

Kazanımlar: K4. Veri tipi dönüşüm işlemlerini yapar.

Materyaller: L3. B4. 1. Veri Tiplerini Ayırt Et!

Hazırlık: Materyalin dört gruba birer adet dağıtmak üzere 4 adet çıktısı alınır.

Uygulama: Öğrenciler B3 etkinliğindeki uzman grupları oturma düzeninde dört grup hâlindedir. Eğitmen bu etkinlikte öğrencilere materyali dağıtır. Kod üzerindeki veri tiplerine ilişkin çıktıyı uzman grubu içerisinde beş dk süresince tartışmaları istenir. Bu materyal ile öğrencilerden kod içerisindeki değişkenler ve veri tiplerinin hangileri olduğu konusunda düşünmeleri beklenir. Eğitmen öğrencilere temel problemlerini tasarlamalarını ve kod üzerinde ilgili değişken isimlerini değiştirmelerini söyler. Bu aşamadan sonra kod çıktısını tahmin etmeleri ve kodu çalıştırıp tahminlerini çıktı ile karşılaştırmaları istenir. Kodun çıktısı aşağıdaki gibidir:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5;
    char b = 'A';

    a = a + b;
    float c = a + 3.0;
    cout << "a = " << a << endl
         << "b = " << b << endl
         << "c = " << c << endl;
    double d = 3.4;
    int e = (int)d + 2;
    cout << "d = " << d << endl;
    cout << "e = " << e << endl;
}
```

```
a = 70
b = A
c = 73
d = 3.4
e = 5
```

Etkinlik sonunda öğrenciler tarafından yazılan problemler ve değişkenler beyin fırtınası yoluyla tartışılır. Neden veri tipleri arasında dönüşüme ihtiyaç duyulduğuna ilişkin konu aşağıdaki gibi özetlenir:

> *Bir değişkende saklanan herhangi bir veri, dönüşüm olarak bilinen bir işlemle farklı bir veri tipindeki bir değişkene dönüştürülebilir. Dönüşüm işleminde, tipi dönüştürülecek değişkenin adından önce parantez içinde kullanılacak veri tipi belirtilir. İki farklı sözdizimi kullanılabilir.*

> *degisken_adi = (veri-tipi) degisken_adi;*

> *degisken_adi = static_dönüsüm <veri-tipi> degisken_adi;*

> *Bir tam sayının başka bir tamsayıya bölünmesi her zaman bir tamsayı sonucu üreteceğinden, dönüşüm genellikle aritmetik bir işlemin sonucunu doğru bir şekilde saklamak için gereklidir. Örneğin, 15/2 tam sayı bölümü, kesilmiş 7 tam sayı sonucunu üretir. Bu işlemden 7,5 gibi bir virgüllü sonuç istiyorsak float tip dönüşümü kullanılması gerekir.*

```
float sayi = (float) 15/2;
```

```
float sayi = static_cast <float> 15/2;
```

İPUCU:

Bir tamsayının başka bir tamsayı ile bölünmesinin sonucu, yuvarlanmaz, kesilir. Bu nedenle 9,9'un kesilmesi sonucu 9 elde edilir.

Eğitmene Öneriler: Etkinlik sonunda zaman kalırsa eğitmen sizeof() operatörü hakkında kısa bir bilgilendirme yapabilir. Bunun için aşağıdaki içerikten yararlanılır:

C++ programlama dilinde hazır olarak var olan sizeof() operatörü, herhangi bir değişkenin tükettiği bellek miktarını bayt cinsinden bize verir. Parametre olarak değişken adını ya da veri tipi adını verebiliriz. Aşağıdaki programda örnek olarak kullanımları verilmektedir.

```
#include <iostream>
using namespace std;
int main()
{
   double sayi;
   cout << "Double boyutu: " << sizeof(sayi) << endl;
   cout << "Char boyutu: " << sizeof(char) << endl;
   cout << "Integer boyutu: " << sizeof(int) << endl;
   cout << "2 adet Float boyutu: " << 2 * sizeof(float) << endl;
   return 0;
}
```

Kodun Çıktısı:

```
Double boyutu: 8
Char boyutu: 1
Integer boyutu: 4
2 adet Float boyutu: 8
```

B5. Kaçmaya Hazırlan

Süre: 15 dk.

Kazanımlar: K5. Kaçış dizgelerini kullanır.

Materyaller: L3. B5. 1. Kaçış Dizgeleri Örnek Kod

Hazırlık: Öğrenciler ikili gruplar hâlinde oturmaktadır. Öğrenme yönetim sisteminden materyal öğrenci erişimine açılır.

Uygulama: Eğitmen tahtaya kaçış dizgelerinin listesini çıkarır ve öğrencilerin örnek kod dosyasını öğrenme yönetim sistemi indirip kendi bilgisayarlarında açmalarını ister. Öğrencilere bu listedeki kaçış dizgelerini örnek kod üzerinde sırayla değiştirip, kodları çalıştırmalarını ister. Her bir değişiklikle program çıktısını karşılaştırıp kaçış dizgelerinin anlamlarını keşfetmeleri istenir. Öğrencilerin keşfettikleri anlamlar ortaya çıktıkça eğitmen tahtada listenin karşısındaki anlam sütununu doldurur.

Eğitmene Öneriler: Etkinlik sonunda erişilecek liste aşağıdaki gibidir:

Kaçış dizgesi	Anlamı
\a	Ses ve uyarı üretir.
\b	İmleci bir pozisyon geri hareket ettirir.
\f	İmleci bir sonraki sayfanın ilk pozisyonuna getirir.
\n	İmleci bir sonraki satırın ilk pozisyonuna getirir.
\r	İmleci mevcut satırın ilk pozisyonuna getirir.
\t	İmleci bir sonraki yatay tab pozisyonuna getirir.
\v	İmleci bir sonraki dikey tab pozisyonuna getirir.
\’	Tek tırnak işareti oluşturur.
\”	Çift tırnak işareti oluşturur.
\?	Soru işareti oluşturur.
\\	Ters eğik çizgi işareti oluşturur.
\0	Boş karakter oluşturur.

B6. Çıktıları Karşılaştır!

Süre: 10 dk.

Kazanımlar: K6. C++ temel giriş/çıkış akışlarını analiz eder.

Materyaller: L3. B6. 1. Kod Çıktılarını Karşılaştır

Hazırlık: Öğrenciler ikili gruplar hâlinde oturmaktadır. Öğrenme yönetim sisteminden materyal öğrenci erişimine açılır.

Uygulama: Eğitmen öğrencilerin örnek iki kodun dosyasını öğrenme yönetim sistemi indirip kendi bilgisayarlarında açmalarını ister. Öğrencilerden cin ve cout arasındaki çıktı farklarını keşfetmeleri istenir. İlk olarak eğitmen öğrencilerden her iki kodu da bilgisayarlarında çalıştırıp karşılaştırmalarını ister. Daha sonra cin ve cout komutlarının görevlerini aralarında tartışmaları istenir. Sonunda eğitmen beyin fırtınası aracılığıyla öğrencilere sorular eşliğinde konuyu özetleyerek açıklar.

Eğitmene Öneriler: Eğitmen konuyu özetlerken aşağıdaki bilgileri kullanabilir.

> *C++ programlama, giriş ve çıkış işlemlerini gerçekleştirmek için birçok farklı kütüphane sunmaktadır. C++ 'da giriş ve çıkış, bayt serisi şeklinde veya daha yaygın olarak akış olarak bilinir. Giriş akışında, bayt akış yönü aygıttan (klavye, disk sürücü, ağ bağlantısı, vb.) ana belleğe doğrudur. Çıkış akışında ise bayt akış yönü tersine ana bellekten aygıta (ekran, yazıcı, disk sürücü vb.) doğrudur. Girdi/çıktı işlemleri için C++ 'da bulunan başlık dosyaları şunlardır:*
>
> - ***iostream:*** *Standart giriş/çıkış akışını temsil eder. Bu başlık dosyası cin, cout, cerr vb. gibi nesnelerin tanımlarını içerir.*
> - ***iomanip:*** *Giriş/çıkış manipülatörleri anlamına gelir. Bu dosyalarda bildirilen yöntemler akışları işlemek için kullanılır. Bu dosya setw, setprecision vb. tanımları içerir.*
> - ***fstream:*** *Bu başlık dosyası temel olarak dosya akışını açıklar. Bu başlık dosyası, bir dosyadan girdi olarak okunan verileri veya dosyaya çıktı olarak yazılan verileri işlemek için kullanılır.*
>
> *C++'da bulunan iki anahtar kelime cin ve cout çıktıları yazdırmak ve girdi almak için kullanılmaktadır. Bu iki ifade girdi ve çıktı almak için en temel yöntemlerdir. C++'da cin ve cout kullanmak için programda iostream başlık dosyasını eklemek gerekmektedir. Aşağıdaki örnekleri inceleyiniz.*

B7. Operatörlerle Yüzleş!

Süre: 20 dk.

Kazanımlar: K7. Operatörleri bir algoritma içinde kullanarak sonucunu yordar.

Materyaller: L3. B6. 1. Operatör Çalışma Kâğıtları

L3. B6. 2. Destekleyici Bilgiler

Hazırlık: Her bilgisayarda Code::Blocks kayıtlı ve aktif çalışır durumda olmalıdır. Çalışma kâğıtlarından 10 adet çıktı alınır ve ikişerli olacak şekilde öğrencilere dağıtılır. Destekleyici bilgiler ÖYS üzerinden materyal olarak erişime açılır.

Uygulama: Eğitmen dört kişilik gruplar oluşturur, ancak öğrencilerin bilgisayar başında ikişerli oturmalarını ister. İkişerli oturan her öğrenciye Operatör Çalışma Kâğıtları dağıtılır. Öğrenciler bu çalışma kâğıtlarında verilen kodların ekran çıktısını kâğıt üzerinde tahmin etmeye çalışacaktır. Eğitmen bu çalışma kâğıtlarını doldururken ÖYS üzerinden erişime açılan Destekleyici Bilgiler materyalinden faydalanabileceklerini belirtir. Dörtlü grup hâlinde çalışma kâğıdındaki kodları tartışmalarına 5 dk. kadar izin verilir. Daha sonra grup üyeleri çalışma yaprağındaki kodları paylaşarak tahmin ettikleri çıktıyı kontrol etmek için mevcut kodları Code::Blocks üzerinde yazıp kontrolünü yaparlar. Hatalı tahminler dörtlü grup içinde tekrar tartışılır.

Eğitmene Öneriler: Öğrenciler destekleyici bilgileri açıp incelemeden eğitmene doğrudan soru yöneltebilir. Bu durumlarda onların materyal üzerinde çalışmalarını teşvik eden "*Bu konudaki soruna destekleyici bilgiler cevap verecektir. Lütfen materyali sistemden açıp inceleyin"* şeklinde geri bildirimler verilmelidir.

B8. Listeyi Dolduralım!

Süre: 20 dk.

Kazanımlar: K8. Bit işlemleri ile verileri hesaplar.

Materyaller: L3. B7. 1. Keşfetme Kartları

Hazırlık: Öğrenciler ikili gruplar hâlinde oturmaktadır. Öğrenme yönetim sisteminden materyal öğrenci erişimine açılır.

Uygulama: Eğitmen tahtaya bit operatörleri ile ilgili aşağıdaki tabloyu görev sütunu boş kalacak şekilde çizer.

Tablo 18. *Boş operatör görev tablosu*

Operatör	İşlem	Görev
~	Bit Değil	
<<	Bit Sola öteleme	
>>	Bit Sağa öteleme	
&	Bit Ve	
^	Bit Özel Veya	
|	Bit Veya	

Öğrencilerden keşfetme kartlarını öğrenme yönetim sistemi üzerinden açmaları istenir. Öğrencilere materyalden yararlanarak bu listedeki görev sütununu tahmin etmeleri söylenir. Öğrencilerin tahminleri ortaya çıktıkça eğitmen tahtada listenin karşısındaki görev sütununu doldurur.

Eğitmene Öneriler: Etkinlik sonunda erişilecek liste aşağıdaki gibidir:

Tablo 19. *Operatör görev tablosu*

Operatör	İşlem	Görev
~	Bit Değil	Bit dizisindeki 0 olan bitleri 1, 1 olan bitleri 0 yapar.
<<	Bit Sola öteleme	En soldaki bit kaybolurken en sağdan bir adet 0 eklenir. Öteleme sayısı değişebilir.
>>	Bit Sağa öteleme	En sağdaki bit kaybolurken en soldan bir adet 0 eklenir. Buradaki öteleme sayısı değişebilir.
&	Bit Ve	İki bitin de 1 olduğu durumda sonuç 1 olurken, diğer durumlarda sonuç 0 olur.

Operatör	İşlem	Görev
^	Bit Özel Veya	İki bitin de aynı anda 0 ya da 1 olduğu durumda sonuç 0 olurken, diğer durumlarda sonuç 1 olur.
|	Bit Veya	İki bitin de 0 olduğu durumda sonuç 0 olurken, diğer durumlarda sonuç 1 olur.

C. Kısmi Öğrenme Görevleri

Süre: 50 dk.

Materyal: L3. C. 1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Kodlayıcı: Kullanıcıdan aldığınız iki sayının değerlerini değiştiren programın kodunu herhangi ekstra bir değişken kullanmadan yazınız (*ipucu: sadece toplama ve çıkarma işlemlerini kullanabilirsiniz*).

```
#include <iostream>
using namespace std;
int main()
{
   int x, y;
   cout << "Birinci sayiyi girin: ";
```

```
    cin >> x;
    cout << "Ikinci sayiyi girin: ";
    cin >> y;

    x = x + y;
    y = x - y;
    x = x - y;

    cout << "X'in yeni degeri : " << x << endl;
    cout << "Y'nin yeni degeri : " << y << endl;
   return 0;
}
```

Denetleyici: Aşağıdaki değişkenlerden isim ve değer atamasına göre uygun olmayan değişken tanımları ve değer atamalarını yuvarlak içine alınız.

int __xyz5;	int _xyz5;	~~const int XYZ;~~	~~short xyz = 34452;~~
int __5xyz;	int xyz_5;	int xyz = '*';	int xyz = 34452;
int __XYZ5;	~~int 5_xyz;~~	char xyz = '\192';	~~float xyz = 12345.12345;~~
~~int xyz=5.2;~~	int _5xyz;	unsigned xyz = '\192';	bool xyz = -1

Kodlayıcı: Aritmetik ve karşılaştırma işleçlerini kullanmadan kullanıcıdan aldığınız iki sayının eşit olup olmadığını kontrol eden ve eşitse ekrana 1, değilse 0 yazan bir program hazırlayınız (*ipucu: bit Özel Veya ve Bit değil operatörlerini kullanabilirsiniz*).

```
#include <iostream>
using namespace std;
int main()
{
    int x, y, z;
    cout << "Birinci sayiyi girin: ";
    cin >> x;
    cout << "Ikinci sayiyi girin: ";
    cin >> y;
```

```
    z = !(x ^ y);
    cout << "Sonuc: " << z << endl;
}
```

Analizci: 3 kişinin yaşları toplamını hesaplayan, program kodu verilmiştir. Kod üzerinde eksik yerleri tamamlayınız.

Eksik Kodlar

```
#include <iostream>
using namespace std;
int main()
{
    int yas1, yas2;
    cout << "1. kisinin yasini giriniz:";
    cin >> yas2;
    cout << "3. kisinin yasini giriniz:";
}
```

Tamamlanmış Kodlar

```
#include <iostream>
using namespace std;
int main()
{
    int yas1, yas2, yas3;
    cout << "1. kisinin yasini giriniz:";
    cin >> yas1;
    cout << "2. kisinin yasini giriniz:";
    cin >> yas2;
    cout << "3. kisinin yasini giriniz:";
    cin >> yas3;
    cout << "\nYaslarinizin toplami: " << yas1+yas2+yas3;
}
```

Tasarlayıcı: Aşağıdaki gibi bir ekran çıktısını kaçış dizgelerini kullanarak oluşturunuz.

\DENEYAP\ Atolyesi'ni "cok" ama "cok" seviyorum!	`#include <iostream>` `using namespace std;` `int main()` `{` `cout << "\\DENEYAP\\ \t\t Atolyesi\'ni\n\n"` `" \t\"cok\" ama \"cok\" seviyorum!\n";` `}`

Hafta 3. Ders Materyalleri

L3. B1. 1. Hatalı Değişkenler

*Her satır bir grubu temsil etmektedir. Satırdaki her sütun ise grup üyesinin kartıdır. Ortadaki sütun gruba ait görev kartını içerir. Lütfen aşağıdaki kartları kesikli noktalardan kesip öğrencilerinize dağıtın. Ancak Kural satırlarını gruplara etkinlik sonunda verin.

_sayi1	_birinci_sayi	Bu grupta sadece bir değişken ismi hatalı yazılmıştır. Kural hatalı olanda gizlidir.	_1sayi	1sayi
Kural 1: Değişken ismi alt çizgi ile başlayabilir. Ancak numara ile değil.				
ikincisayi	sayi_ikinci	Bu grupta sadece bir değişken ismi hatalı yazılmıştır. Kural hatalı olanda gizlidir.	sayi_2	2sayi
Kural 2: Değişken ismi bir harf ile başlayabilir. Ancak numara ile değil.				

ücuncüsayi	üçüncüsayı	Bu grupta sadece bir değişken ismi doğru yazılmıştır. Kural doğru olanda gizlidir.	uçuncusayi	ucuncusayı
Kural 3: Değişken isminde Türkçe karakter kullanılmaz.				
SAYIdort	sayiDORT	Bu gruptaki tüm değişkenler doğru yazılmış ve her biri farklı bir değişkendir. Kural doğru olanda gizlidir.	SayiDort	Sayidort
Kural 4: Değişken adları C++' da büyük/küçük harfe duyarlıdır; bu nedenle “numara”, “Numara” ve “NUMARA” gibi değişkenler üç ayrı değişken olarak ele alınır.				

sayi bes	SAYI bes	Bu gruptaki üç değişken hatalı yazılmıştır. Kural hatalı olanda gizlidir.	Sayı 5	sayibes
Kural 5: Değişken ismi yazılırken boşluk kullanılmaz.				

L3. B1. 2. Değişken İsmi Olamazlar!

C++ ANAHTAR KELİMELER:					
and	auto	bool	break	case	catch
char	class	concept	const	continue	default
delete	do	double	else	enum	extern
false	float	for	friend	goto	if
int	long	namespace	new	not	operator
or	private	protected	public	register	return
short	signed	sizeof	static	struct	switch
template	this	throw	true	try	typedef
union	unsigned	virtual	void	volatile	while

UYARI:

Yukarıda verilen anahtar kelimeleri yanlışlıkla değişken adı olarak kullanmayın. Anahtar kelimeleri bu şekilde kullanmak öngörülemeyen sonuçlara neden olabilir ve birçok sıkıntıya sebep olabilir. Örneğin, “short” bir sayıyı temsil etmek için ayrılmış bir kelimedir ve bu kelimeyi bir değişken adı olarak kullanmaya çalışırsanız, program oluşturulduğunda derleyici size bir hata verecektir.

L3. B2. 1. Sabitler için Örnek Kod

Aşağıda yer alan sabitleri bulup yuvarlak içine alınız. Sabitleri ayırmak için aşağıdaki üç kurala dikkat ediniz.

1. "const" anahtar sözcüğünü kullanarak tanımlarız.
2. Sabit adları, değişken adlarından ayırt etmek için büyük harfle yazılır.
3. Sabitler her zaman tanımlama sırasında değerlerini almalıdır.

```
#include <iostream>
using namespace std;
int main()
{
    const int DOGUM_YILI = 2005;
    int boy = 175, kilo=72;

    cout << "Dogum yilim: " << DOGUM_YILI << endl;
    cout << "Boyum: " << boy << endl;
    cout << "Kilom: " << kilo << endl;
}
```

L3. B3. 1. Veri Tipleri Çalışma Kâğıtları

Veri Tipi	Açıklama	Boyut	Örnek
bool	Bool veri tipi en basit veri tipidir ve bir bayt uzunluğundadır. Doğru (1) ve yanlış (0) olmak üzere iki değerden birini saklayabilir. Programınızda yalnızca iki olasılığınız olduğunda bool türünü kullanabilirsiniz. Doğru (true-1) veya yanlış (false-0) değerlerini alabilen veri tipidir.	1 Bayt	Işığın açık olup olmadığı (0: kapalı, 1: açık)
char	Bir karakter (char) veri türü, standart ASCII tablosunda gösterilen herhangi bir harf veya simge olabilecek 256 farklı karakterden herhangi birini içerebilir. Dolayısıyla karakter kodu, her karakterle ilişkilendirilmiş bir tam sayıdır. Örneğin A harfini kodu 65 ile temsil edilir. Bir karakter değişkeni iki tek tırnakla ifade edilir. Örneğin, ‘C’, ‘5’, ‘*’ ve ‘T’ karakter ifadeleridir. “char” anahtar sözcüğünü kullanarak aşağıdaki gibi bir karakter türü bildirebilirsiniz: Bir karakter tutabilen tek bir bayttır.	1 Bayt	İsminin baş harfi: ‘A’, ‘d’

int	Int (tam sayı) veri türü temel olarak tam sayıları temsil eder (içerisinde kesirli parça yoktur). Tam sayılarla işlem yapmak için üç farklı tam sayı türü vardır. Bu türler değer aralıkları ile ayırt edilir. ❖ **int** ❖ **short int** (veya **short**) ❖ **long int** (veya **long**). Programlamada en sık kullanılan veri türüdür. Tam sayı türleri virgüllü sayıları veya kesirleri saklayamaz (örneğin, 5.1 veya 1/4). 16 bit bilgisayarlar için int veri tipi short veri tipine eşdeğer olurken, 32 bit bilgisayarlar için int veri tipi long veri tipine eşdeğer olur. `int sayi = 5;` `cout << sayi;` Short veri tipi, int veri tipinin yarısı boyutunda, yani 2 bayttır. -32,768 ile 32,767 arasında değer alabilir. Nispeten küçük değerleriniz olduğunda, bu tür daha kullanışlıdır. Bellek açısından baktığımız zaman int türünün sadece yarısını kapladığı için iki kat daha verimli olmaktadır. `short sayi = 5;` `cout << sayi;` `short int sayi = 5` `cout << sayi;` Tam sayı veri tipleri arasındaki diğer veri tipimiz de long veri tipidir. Int veri tipi gibi 4 bayt uzunluktadır. Dolayısıyla -2,147,483,648 ile 2,147,483,647 arasında değer alabilir. Aşağıdaki şekillerde tanımlayabiliriz:	4 Bayt	Yaş (17), Sınıf (3)

	`long sayi = 597;` `cout << sayi;` `long int sayi = 597;` `cout << sayi;`		
float	Gerçek hayattaki veriler her zaman tam sayı olmak zorunda değil. Örneğin, kişinin boyu, marketteki bir ürünün fiyatı ya da bir sayının karekökü ondalık yani virgüllü bir sayı olabilir. Ondalık sayıları ya da kesirleri saklamak için float ve double veri türlerine ihtiyacımız vardır. Tam sayıların aksine, ondalık sayılar önceden belirlenmiş bir hassasiyette saklanmalıdır. Tiplerin değer aralığı ve hassasiyeti ayrılan bellek miktarına göre tespit edilir. Ondalık kısmında genel olarak 7 basamak hassasiyete sahiptir. Float veri tipi 4 bayt uzunluğundadır. Pozitif değerler için $1,2 \times 10^{-38}$ ile $3,4 \times 10^{38}$ ve negatif değerler için $-1,2 \times 10^{-38}$ ile $-3,4 \times 10^{38}$ aralığında değer alabilir.	4 Bayt	Boy (1,75), Kilo (60,5)
double	Double veri tipi 8 bayt uzunluğundadır. Pozitif değerler için $2,2 \times 10^{-308}$ ile $1,7 \times 10^{308}$ ve negatif değerler için -2.2×10^{-308} ile -1.7×10^{308} aralığında değer alabilir. Görüldüğü üzere son derece yüksek bir hassasiyete sahiptir. Ondalık kısmında genel olarak 16 basamak hassasiyete sahiptir. Virgüllü sayıların kullanımında bilimsel gösterim kullanılabilir. Aşağıda bazı sayıların bilimsel gösterimde nasıl yazıldığını görebilirsiniz: 2.2748e+21 -3.15479e+3 342.234e-14 44.78e-13	8 Bayt	Pi Sayısı (3.1415926535)

L3. B3. 2. Değişken Atama Kodları

Bool değişken tanımlama.

```
#include <iostream>
using namespace std;
int main(){
    bool deger = true;
    cout << "Deger : " << deger;
    return 0;
}
```

Kodun Çıktısı:

```
Deger: 1
```

Char değişken tanımlama.

```
#include <iostream>
using namespace std;
int main() {
     char harf = 'a';
     cout << "Yazilan harf: " << harf;
     return 0;
}
```

Kodun Çıktısı:

```
Yazilan harf: a
```

Int değişken tanımlama.

```
#include <iostream>
using namespace std;
int main(){
    int yas = 15;
    int kilo = 52;
    int boy = 167;
    cout << "Benim yasim: " << yas;
    cout << ", kilom: " << kilo;
    cout << " ve boyum: " << boy;

    return 0;
}
```

Kodun Çıktısı:

```
Benim yasim: 15, kilom: 52 ve boyum: 167
```

Float değişken tanımlama.

```
#include <iostream>
using namespace std;
int main(){
    float sayi1 = 309.572;
    float sayi2 = -78.271;
    cout << "Sayi 1: " << sayi1 << "\nSayi 2: " << sayi2;
    return 0;
}
```

Kodun Çıktısı:

```
Sayı 1: 309.572
Sayı 2: -78.271
```

Double değişken tanımlama.

```
#include <iostream>
using namespace std;
int main()
{
    double sayi1 = 2.2e-308;
    double sayi2 = -2.3e-308;
    cout << "Sayi 1: " << sayi1 << "\nSayi 2: " << sayi2;
    return 0;
}
```

Kodun Çıktısı:

```
Sayı 1: 2.2e-308
Sayı 2: -2.3e-308
```

L3. B4. 1. Problemi Keşfet!

Kodlar	Kod içerisindeki değişkenler ve veri tipleri nelerdir?

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5;
    char b = 'A';

    a = a + b;
    float c = a + 3.0;

    cout << "a = " << a << endl
         << "b = " << b << endl
         << "c = " << c << endl;

    double d = 3.4;
    int e = (int)d + 2;

    cout << "d = " << d << endl;
    cout << "e = " << e << endl;

    return 0;
}
```

Kodun Çıktısı: Kodun çıktısını tahmin ediniz.

L3. B5. 1. Kaçış Dizgeleri Örnek Kod

```
#include <iostream>
using namespace std;
int main()
{
    cout << "\nBu \t cumlede\n\t\t"
            " cok \"fazla\" kacis dizgesi vardir!\n";
    return 0;
}
```

Kodun Çıktısı:

```
Bu      cümlede

        cok “fazla” kacis dizgesi vardir!
```

L3. B6. 1. Kod Çıktılarını Karşılaştır

```
#include <iostream>

using namespace std;

int main()

{

    cout << "DENEYAP projesi ile gelecegimizi sekillendiriyoruz!";

    return 0;

}
```

```
#include <iostream>

using namespace std;

int main()

{

    int yas;

    cout << "Yasinizi giriniz:";

    cin >> yas;

    cout << "\nYasiniz: " << yas;

    return 0;

}
```

L3. B7. 1. Operatör Çalışma Yaprağı

```
#include <iostream>
using namespace std;
int main()
{
    int a = 23, b = 5;
    cout << "Ekleme: " << (a + b) << endl;
    cout << "Cikarma: " << (a - b) << endl;
    cout << "Carpma: " << (a * b) << endl;
    cout << "Bolme: " << (a / b) << endl;
 cout << "Mod alma: " << (a % b) << endl;
 cout << "Arttirma: " << a++ << endl;
 cout << "Arttirma: " << ++a << endl;
 cout << "Azaltma: " << b-- << endl;
 cout << "Azaltma: " << --b << endl;

return 0;
}
```

Kodun Çıktısı:

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 5, y = 4;
    cout << (x < y) << endl;
    cout << (x > y) << endl;
    cout << ((x-1) <= y) << endl;
    cout << (x >= (y+2)) << endl;
    cout << (x == y) << endl;
    cout << (x != y) << endl;
}
```

Kodun Çıktısı:

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 5, y = 0;
    cout << ((x <= y) || (y>0)) << endl;
    cout << ((x > 4) && (y==0)) << endl;
    cout << (x && !y) << endl;
    cout << (!(x-2) || (y+2 > 2)) << endl;
}
```

Kodun Çıktısı:

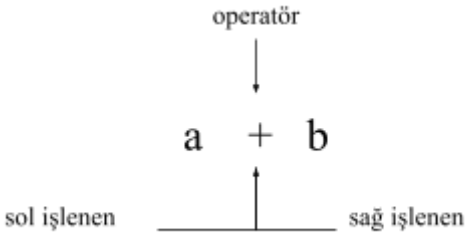
```
#include <iostream>
using namespace std;
int main()
{
    float x, y, z;
    x = 2.4;
    y = x;
    z = x + 1.3 + (y * 2.0);

    cout << "x: " << x << endl;
    cout << "y: " << y << endl;
    cout << "z: " << z << endl << endl;

    x = y = 3.4;
    cout << "x: " << x << endl;
    cout << "y: " << y << endl << endl;

    x+= 5.1;
    y+= y * 2.0;
    cout << "x: " << x << endl;
    cout << "y: " << y << endl << endl;

    x/= 2.0;
    y*= 3.0;
    cout << "x: " << x << endl;
    cout << "y: " << y << endl << endl;
}
```

Kodun Çıktısı:

L3. B6. 2. Destekleyici Bilgiler

Aritmetik Operatörler

Matematiksel hesaplamalar bilgisayar programlarında yaygın olarak kullanılmaktadır. Programlama dilinde bir operatör, bir değer veya değişken üzerinde çalışan bir semboldür. Aşağıda görüldüğü üzere a değişkeni sol işlenen, b değişkeni sağ işlenen ve + sembolü de operatörü oluşturmaktadır.



Aşağıdaki tabloda, bilgisayar programlamasında kullanılan önemli aritmetik operatörler listelenmiştir.

Tablo 20. *Önemli aritmetik operatörler*

Operatör	Açıklama	Kullanım
+	İki işleneni birbirine ekler.	x + y
-	İkinci işleneni birinciden çıkarır.	x - y
*	İki işleneni çarpar.	x * y
/	İkinci işlenenle birinciyi böler.	x / y
%	Tamsayı bölümünün kalanını verir.	x % y
++	Sayıyı bir arttırma	x++ veya ++x
--	Sayıyı bir azaltma	x-- veya --x

Tekli operatör olan ++ veya -- işleçleri değişken değerlerini 1 arttırmak ya da 1 azaltmak için kullanılır. “x++” ifadesinde x değişkeni var olan değeri ile işleme girer ve işlem tamamlandıktan sonra x değişkeninin değeri bir arttırılır. “++x” ifadesinde ise önce değişkenin değeri bir arttırılır ve daha sonra işlem yapılır. Aynı şekilde -- operatörü içinde benzer kullanım geçerlidir. Aşağıdaki örneği inceleyiniz.

```
a = 5
b = ++a          // a değeri 6 olur, b değeri 6 olur
c = 2
```

```
d = c++;          // c değeri 3 olur, d değeri 2 olur

a = 5
b = --a          // a değeri 4 olur, b değeri 4 olur

c = 2
d = c--;          // c değeri 1 olur, d değeri 2 olur
```

UYARI:

Bir ön-ek operatörünün değişken değerini hemen değiştirdiğini, bir son-ek operatörünün değişken değerini daha sonra değiştirdiğini unutmayalım.

Karşılaştırma Operatörleri

Bu işleçlerin amacı iki değişken veya değişken grubunu belirtilen şarta göre karşılaştırmaktır. Bu karşılaştırmalar aynı türdeki değişkenler arasında olmalıdır. Bu işleçleri özellikle ilerleyen haftalarda göreceğimiz koşul yapıları ve döngülerde kullanılır. Yapılan karşılaştırmalar doğruysa "1" yanlışsa "0" sonucunu döndürür.

Tablo 21. *Önemli aritmetik operatörler*

İşleç	Açıklama	Kullanım
<	Küçüktür	x < y
>	Büyüktür	x > y
<=	Küçük eşittir	x <= y
>=	Büyük eşittir	x >= y
==	Eşittir	x == y
!=	Eşit değildir	x != y

UYARI:

Programlamada en çok yapılan hatalardan biri iki ifadeyi karşılaştırmak için atama operatörünün (=) kullanılmasıdır. Böyle bir atama yaptığınızda soldaki değer bir değişkense derleyici bir hata mesajı oluşturmaz. Bu hata, programlamaya yeni başlayanların en çok dikkat etmesi gereken bir hatadır.

Atama Operatörleri

Atama işleçleri, bir değişkene değer atamak için kullanılır. Atama operatörünün sol taraftaki işleneni değişkendir ve atama operatörünün sağ taraftaki işleneni bir değerdir. Farklı tür atama operatörleri aşağıda tabloda verilmektedir.

Tablo 22. *Atama operatörleri*

İşleç	Açıklama	Kullanım
=	Sağdaki değeri soldaki değişkene atamak için kullanılır.	x = y z=10
+=	Bu işleç, '+' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere ekler ve ardından sonucu soldaki değişkene atar.	x+=y (x=x+y)
-=	Bu işleç, '-' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerden çıkarır ve ardından sonucu soldaki değişkene atar.	x-=y (x=x-y)
*=	Bu işleç "*" ve "=" operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerle çarpar ve ardından sonucu soldaki değişkene atar.	x*=y (x=x*y)
/=	Bu işleç "/" ve "=" operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere böler ve ardından sonucu soldaki değişkene atar.	x/=y (x=x/y)
%=	Bu işleç "%" ve "=" operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere göre modunu alır ve ardından sonucu soldaki değişkene atar.	x%=y (x=x%y)

Mantıksal Operatörler

Mantıksal operatörler programlama dillerinde çok önemli bir yere sahiptir ve belirli koşullara göre karar vermemize yardımcı olurlar. İki koşulun sonucunu birleştirmek istediğimizde mantıksal VE ve VEYA istediğimiz sonucu üretmemize yardımcı olur. Bütün şartların sağlanması için koşullar arasına VE, herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA ve koşulu sağlamayanlar isteniyorsa DEĞİL işleci kullanılmalıdır.

Tablo 23. *Mantıksal operatörler*

İşleç	Açıklama	Kullanım
&&	Mantıksal VE	x && y
||	Mantıksal VEYA	x || y
!	Mantıksal DEĞİL	!x

Mantıksal işleçlerin sonucu şu şekilde belirlenmektedir. Eğer x&&y kullanılıyor ise her iki değişkenin değeri "1" olursa sonuç "1" olur, aksi hâlde sonuç "0" olur. Eğer x||y kullanılıyor ise her iki değişkenin değeri "0" olursa sonuç "0" olur, aksi hâlde sonuç "1" olur. Eğer !x kullanılıyor ise değişkenin değeri "1" ise sonuç "0", "0" ise sonuç "v1" olacaktır. A ve B değişkenleri için oluşturulan aşağıdaki tabloyu inceleyebilirsiniz.

Tablo 24. *Mantıksal operatörlerin kullanımı*

A	B	A && B	A || B	! A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

UYARI:
x veya x+1 gibi sayısal bir değer, değeri 0 ise "false", 0 dışında bir değer ise "true" olarak yorumlanır.

Operatör Önceliği

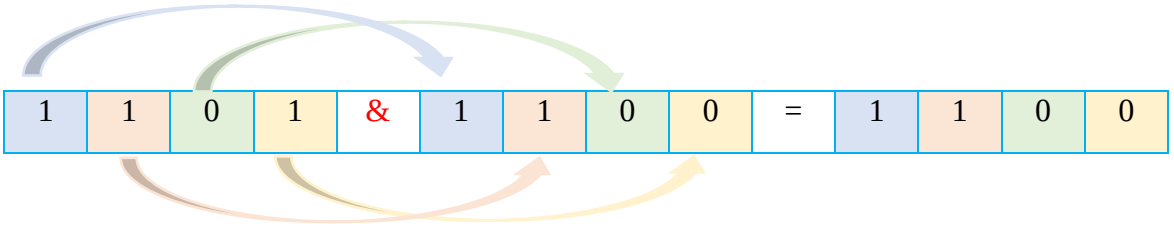
C++ programlarınızı oluştururken aritmetik operatörlerin önceliğine dikkat etmeniz gerekmektedir. Operatör önceliği değerlendirme sırasını, yani operatörlerin ve işlenenlerin nasıl gruplandığını belirler. Aşağıdaki tablodaki öncelik sırasını inceleyiniz.

Tablo 25. *Operatör öncelikleri*

Öncelik	Operatör	İşlem Yönü
1	()	Soldan sağa
2	~ ++ - -	Sağdan sola
3	* / %	Soldan sağa
4	+ -	Soldan sağa
5	<< >>	Soldan sağa
6	< <= > >=	Soldan sağa
7	== !=	Soldan sağa
8	&	Soldan sağa
9	^	Soldan sağa
10	|	Soldan sağa
11	&&	Soldan sağa
12	||	Soldan sağa
13	= += -= *= /= %=	Sağdan sola

L3. B7. 1. Keşfetme Kartları

Bit Ve (&) Operatörü



1	1	0	1	&	1	1	0	0	=	1	1	0	0

1	0	1	1	&	1	1	1	0	=	1	0	1	0

1	0	0	1	&	1	1	0	1	=	1	0	0	1

0	0	1	1	&	1	1	0	0	=	0	0	0	0

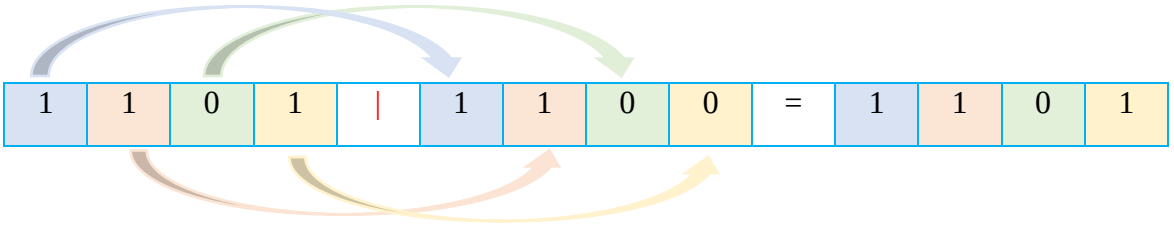
Örnek Kod:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 27;          // x = 0001 1011
    int y = 58;          // y = 0011 1010
    int z = x & y;       // z = 0001 1010
    cout << "Sonuc: " << z << endl;
}
```

Kodun Çıktısı:

```
Sonuc: 26
```

Bit Veya (|) Operatörü



0	0	0	0	|	0	0	0	0	=	0	0	0	0

1	0	0	0	|	1	1	0	0	=	1	1	0	0

0	0	1	1	|	1	1	0	0	=	1	1	1	1

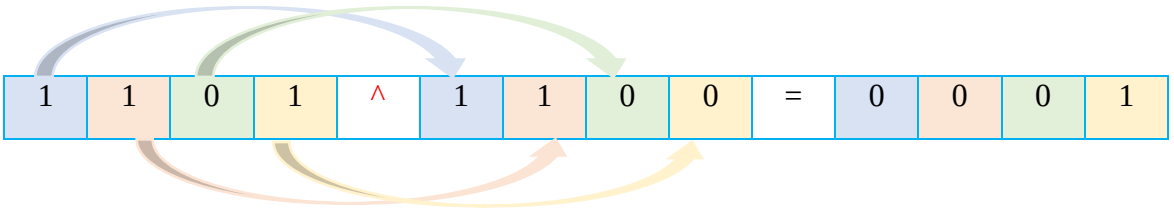
Örnek Kod:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 23;          // x = 0001 0111
    int y = 78;          // y = 0100 1110
    int z = x | y;       // z = 0101 1111
    cout << "Sonuc: " << z << endl;
}
```

Kodun Çıktısı:

```
Sonuc: 95
```

Bit Özel Veya (^) Operatörü



0	1	1	0	^	0	1	1	0	=	0	0	0	0
1	0	0	0	^	1	1	0	1	=	0	1	0	1
0	0	1	1	^	1	1	0	0	=	1	1	1	1

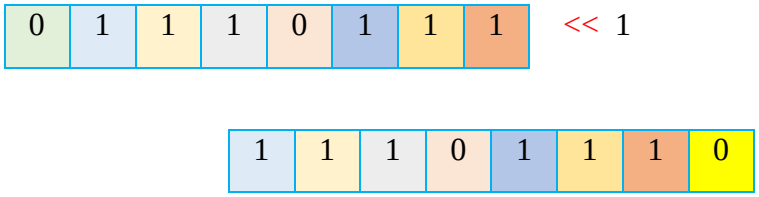
Örnek Kod:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 23;          // x = 0001 0111
    int y = 78;          // y = 0100 1110
    int z = x ^ y;       // z = 0101 1001
    cout << "Sonuc: " << z << endl;
}
```

Kodun Çıktısı:

```
Sonuc: 89
```

Bit Sola (<<) Öteleme Operatörü



Örnek Kod:

```
#include <iostream>
using namespace std;
int main()
{
    y = x << 1;          // y = 0010 1110
    cout << "23 << 1: " << y << endl;

    y = x << 3;          // y = 1011 1000
    cout << "23 << 3: " << y << endl;
    return 0;
}
```

Kodun Çıktısı:

```
23 << 1: 46
23 << 3: 184
```

Bit Değil (~) Operatörü

~	1	0	0	1	=	0	1	1	0

~	1	1	0	0	=	0	0	1	1

~	1	0	0	0	=	0	1	1	1

~	1	1	1	1	=	0	0	0	0

Örnek Kod:

```
#include <iostream>
using namespace std;
int main()
{
    unsigned char x = 23;    // x = 0001 0111
    unsigned char y = 78;    // y = 0100 1110
    unsigned char z = ~x;    // z = 1110 1000
    cout << "~23: " << (int)z << endl;

    z = ~y;                  // z = 1011 0001
    cout << "~78 : " << (int)z << endl;
    return 0;
}
```

Kodun Çıktısı:

```
~23: 232
~78 : 177
```

L3. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 4. Karar Mantık Yapıları

Kazanımlar

- K1. C++ programlama dilinde tekli karar yapısını kullanarak programı test eder.
- K2. C++ programlama dilinde çoklu karar yapısını kullanarak programı test eder.
- K3. C++ programlama dilinde iç içe karar yapısını kullanarak programı test eder.

Amaç

Bu haftanın amacı öğrencilerin programlamada karar yapılarını kullanarak programın akışını kontrol edebilmelerini sağlamaktır.

Önerilen Ders Akışı

A. Giriş (10 dk.)
B. Öğrenme Görevleri
B1. Karar Yapılarını Tanıyalım! (40 dk.)

- ❑ Ders Arası (10 dk.)

B2. Görevleri Kodlayalım! (60 dk.)

- ❑ Ders Arası (10 dk.)

B3. İç İçe Koşula Farklı Bir Bakış (40 dk.)

- ❑ Ders Arası (10 dk.)

C. Kısmi Öğrenme Görevleri (60 dk.)

A. Giriş:

Süre: 10 dk.

Uygulama: Sınıfta bulunan her öğrenci kendine bir oyun arkadaşı seçer. Herkes seçtiği oyun arkadaşı ile taş, kâğıt ve makas oyununu karşılıklı oynar, oyunu kaybeden kazandığı kişinin arkasına geçerek onun takipçisi olur. Kazananlar sürekli bir diğer kazananla karşılaşarak aynı şekilde taş, kâğıt ve makas oyununu tekrar eder, kaybeden her kişi kazananın arkasına geçmektedir ve kazananın takipçisi olmaktadır. En uzun kuyruğa ulaşan bir başka ifadeyle hiç kaybetmeyen kişi oyunu kazanır.

Eğitmene Öneriler: Oyundaki amaç kazananı belirlemekten çok grup dinamiğini canlı tutmaktır. Oyunu kaybeden öğrenci, kazananın arkasına geçmektedir ve kaybettiği arkadaşı bir sonraki diğer kazananla yarıştığında arkasında bulunduğu için aslında farkında olmadan kaybettiği arkadaşına destek vermektedir.

B. Öğrenme Görevleri

B1: Karar Yapılarını Tanıyalım

Süre: 40 dk.

Kazanımlar: K1: C++ programlama dilinde tekli karar yapısını kullanarak programı test eder.

K2: C++ programlama dilinde çoklu karar yapısını kullanarak programı test eder.

K3: C++ programlama dilinde iç içe karar yapısını kullanarak programı test eder.

Materyaller: L4.B1.1 Karar Yapılarını Tanıyalım Görev Kâğıtları

Hazırlık: Eğitmen bu konu için hazırlanan 5 sayfalık görev kâğıtlarının çıktısını alarak derse gelir.

Uygulama: Sınıf her bir grupta öğrenci sayısı eşit olacak şekilde 5 gruba ayrılır. Daha sonra her bir grubun ortak karar verecek şekilde 1 ile 100 arasında sayı belirlemeleri istenir. Eğitmen bu bölüm için hazırlanan görev kâğıtlarının çıktısını alarak her bir gruba birer karar yapısı içeren görevleri verir. Gruplardan daha önceden belirledikleri sayı ile birlikte elindeki karar yapılarının bulunduğu görev kâğıtlarını karşılaştırılıp, görev kâğıtlarındaki algoritmanın hangi çıktıyı vereceğine yönelik cevap istenir. Eğitmen diğer dört görevden farklı olarak 5. görevde bulunan algoritmanın neye yönelik yazıldığını öğrencilerin cevap kâğıtlarına yazmasını ister. Eğitmen bulunan çıktı sonuçlarının doğru veya yanlışlığı hakkında öğrencilere geri bildirim vererek her bir gruptaki öğrencilerin doğru sonucu bulması hedeflenir. Daha sonra her bir görev kâğıdının her bir gruba dolaşması sağlanarak öğrencilerin çeşitli karar yapı örneklerinin

görünmesi sağlanır. Görev kâğıtları dolaşırken her değişimde eğitmen karar yapılarına uygun sayılar belirleyerek ve her değişimde bu sayıları değiştirerek grupların o sayıya yönelik programda hangi çıktıyı vereceği öğrenciler tarafından bulunması istenir.

NOT: Her bir görev kâğıdının her gruba ulaşması sağlanır. Öğrenciler etkinliği bitirdikten sonra her bir görev kâğıdının günlük yaşamda hangi problemi çözmek için yazıldığını öğrencilerden tahmin edilmesi istenir ve bununla ilgili sınıfta eğitmen eşliğinde tartışma gerçekleştirilir.

Eğitmene Öneriler: Yukarıdaki etkinliğin amacı; öğrencinin program akışının bir noktasında verilen karara göre yapılacak işlemleri göstermek ve verilen karara bağlı olarak ekran çıktısının nasıl değişebileceğini öğrencilerin görmesini sağlamaktır.

Yukarıdaki etkinlik bittikten sonra eğitmenin karar yapılarının C++ dilinde nasıl kodlandığına yönelik aşağıdaki gibi kısa bir bilgilendirme yaparak özetleme yapması beklenir.

Programlama dillerinin olmazsa olmazı karar ifadeleridir. Karar ifadeleri program akışına yön vermek için kullanılır. Belirli şartlara (sıcaklık değeri, klavyede bir tuşa basılıp basılmadığı, bir değer girilip girilmediği) göre yapılması gereken işlemleri karar ifadeleri ile gerçekleştiririz. C++ programlama dilinde bu amaç için kullanılan “if”,”if-else” ve “switch” ifadeleridir.

Eğer koşula göre yapılacak bir veya daha çok işlem varsa ilgili blok ‘{‘ ve ‘}’ arasına yazılır. Tek satırlık işlemler için ‘{‘ ve ‘}’ karakterlerine gerek yoktur.

```
if(koşul)
{
        koşul doğru ise yapılacak işlemler
}
```

yerine aşağıdaki şekilde yazılabilir.

```
if(koşul)
        koşul doğru ise yapılacak işlem
```

Bazı durumlarda koşul gerçekleşmediğinde de işlem yapmak isteriz. Koşulun gerçekleşmemesi durumunu işleme almak için “aksi takdirde” anlamına gelen “else” komutu yazılır.

```
if(koşul)
{
```

```
        koşul doğru ise yapılacak işlemler
}
else
{
        koşul yanlış ise yapılacak işlemler
}
```

Bazı durumlarda koşullarımız birden fazla olabilir. Bu durumlarda her bir koşul için "else if" kullanmaktayız.

```
if(koşul1)
        koşul1 için yapılacak işlem
else if(koşul2)
        koşul2 için yapılacak işlem
else if(koşul3)
        koşul3 için yapılacak işlem
else
        diğer tüm durumlar için yapılacak işlem
```

B2: Görevleri Kodlayalım

Süre: 60 dk.

Kazanımlar: K1: C++ programlama dilinde tekli karar yapısını kullanarak programı test eder.

K2: C++ programlama dilinde çoklu karar yapısını kullanarak programı test eder.

K3: C++ programlama dilinde iç içe karar yapısını kullanarak programı test eder.

Materyaller: L4. B1.1 Karar Yapılarını Tanıyalım Görev Kâğıtları

Hazırlık: Eğitmen bir önceki etkinlikteki materyalleri kullanmaya devam eder.

Uygulama: Öğrencilerin materyalde belirtilen ve bir ders önceki yaptıkları "Karar Yapılarını Tanıyalım Görev Kâğıtları" etkinliği içerisinden seçtiği gerçek yaşamla ilgili iki karmaşık görevi kendi başlarına kodlaması beklenir. Öğrenciler kodlama yaparken eğitmenlerden biri sınıfta dolaşarak öğrencilerin ihtiyaç duyduğu anda yöntemsel bilgi vermesi beklenir. Diğer

eğitmen ise kodları bilgisayarda yazarak öğrencilerin seçtiği görevlerdeki bazı aşamaları ihtiyaç hâlinde görmesi sağlanır.

Eğitmene Öneriler: Her bir bilgisayara iki öğrenci oturduğu düşünülürse, iki öğrencinin birer görev yapması sağlanarak zamanın etkili bir şekilde kullanımı sağlanabilir.

B3: İç içe Koşula Farklı Bir Bakış

Süre: 40 dk.

Kazanımlar: K3: C++ programlama dilinde iç içe karar yapısını kullanarak programı test eder.

Materyaller: L4. B3.1. SWITCH Yapısı Kullanımı 1

L4. B3.2. SWITCH Yapısı Kullanımı 2

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan materyalleri ÖYS üzerinden paylaşır.

Uygulama: Bu bölüm için hazırlanan materyal ÖYS üzerinden açılarak if yapısı ile switch yapısı konusunda kod yazımında ne gibi farklılıkların olduğuna yönelik sınıf içi tartışma ortamı oluşturulur. Daha sonra eğitmen tarafından aşağıdaki gibi konu öğrencilerin de katılımı ile özetlenir.

Bir değişkenin değerini birden çok "if" koşulunu ile kontrol etmek yerine daha "switch" ve "case" bloklarını kullanabiliriz. Her koşul durumunu "case" komutları ile belirterek daha kolay okunabilir bir kod parçası oluşturabiliriz. Burada kullanılacak değerler sabit olacaktır ve kullanılacak olan değişkenin kullanacağımız durumlarını önceden biliyor olmamız gerekiyor. Örneğin int tipindeki Not değişkeni için sabit 1, 2, 3, 4, 5 değerleri için switch-case söz dizimi şu şekildedir;

```
switch(Not)
{
        case 1:
                Yapılacak işlemler
        break;
        case 2:
                Yapılacak işlemler
        break;
        …
```

```
        default:
                Yapılacak işlemler
        break;
}
```

break: Bu komut, koşul bloğundan çıkmamızı sağlar.

default: Verilen koşulların gerçekleşmemesi hâlinde çalışır.

Eğitmen rehberliğinde "L4. B3.2. SWITCH Yapısı Kullanımı 2" adlı materyal kısmında verilen switch kullanımına yönelik iki etkinlikten birinin öğrenciden seçerek kodlama yapması istenir.

C. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L4. C.1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilerin kolay ulaşabileceği noktalara asılır ya da öğrencilerin Öğrenme Yönetim Sistemi ortamında materyal olarak erişmeleri sağlanır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Her bir görevi tamamlayan öğrencilere, göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi öğrenme görevinin ismi beceri rozetini tanımlamaktadır. Beceri rozetine ait görevlerin yanıtları ise aşağıda verilmektedir.

1)Kodlayıcı:

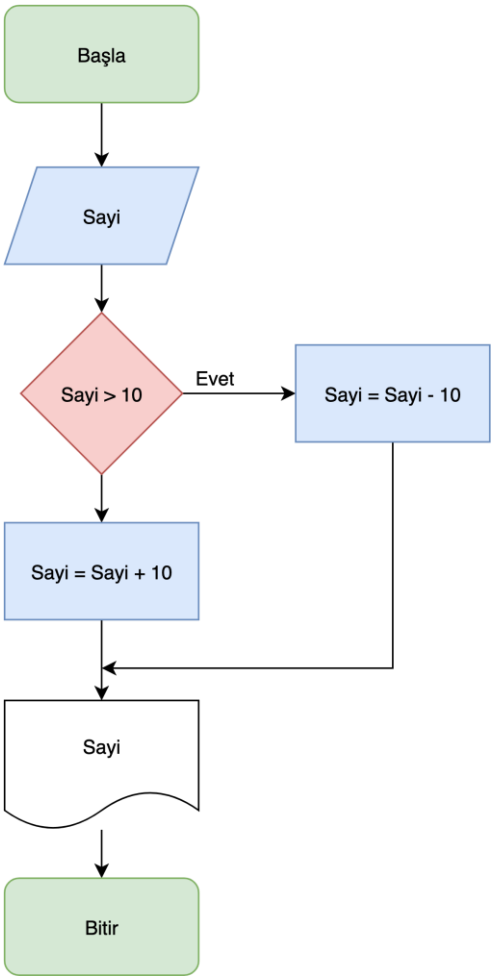
```
#include <iostream>
using namespace std;

int main()
{
    int sayi;
    cin>>sayi;

    if(sayi %2 == 0)
    {
        if(sayi %4 == 0)
            cout << "Sayi cifttir ve 4 ile bolunur";
        else
            cout << "Sayi cifttir ve 4 ile bolunmez";
    }
    else
    {
        if(sayi %3 == 0)
            cout << "Sayi tektir ve 3 ile bolunur";
        else
            cout << "Sayi tektir ve 3 ile bolunmez";
    }
}
```

2)Analizci

```
#include <iostream>
using namespace std;
int main()
{
    int sayi = 13;
    if(sayi % 2 == 0)
        cout << sayi /2;
    else
        cout << (sayi-1) /2;
}
```

Cevap: 6

3)Tasarlayıcı



Cevap:

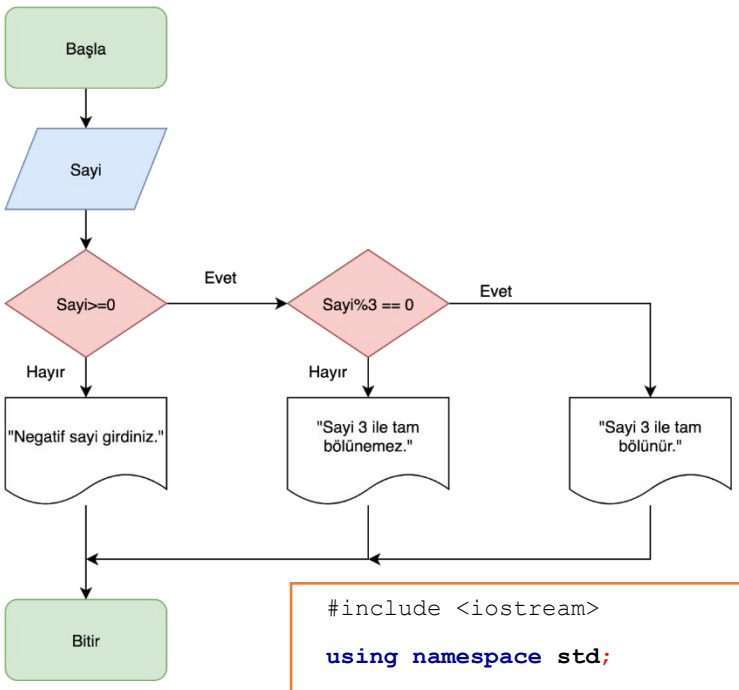
```
#include <iostream>
using namespace std;
int main()
{
    int sayi;
    cin >> sayi;
    if(sayi>10)
        sayi = sayi -10;
    else
        sayi = sayi +10;
    cout << sayi;
}
```

Hafta 4. Ders Materyalleri

L4. B1.1 Karar Yapılarını Tanıyalım Görev Kâğıtları

Not: Her görev, eğitmenler için kodu ile birlikte verilmiş olup öğrencilere görevlerin çıktısı alınırken kodların silinmesi gerekmektedir. Öğrencilerin akış diyagramına bakarak görevleri yerine getirmesi beklenmektedir.

1. Görev




```
#include <iostream>
using namespace std;
int main()
{
      int sayi;
      cin >> sayi;

      if(sayi >= 0)
      {
          if(sayi%3 == 0)
              cout << "Sayi 3 ile tam bolunur.";
          else
                  cout << "Sayi 3 ile tam bolunemez.";
      }
      else
      {
          cout << "Negatif sayi girdiniz.";
      }
      return 0;
 }
```

1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

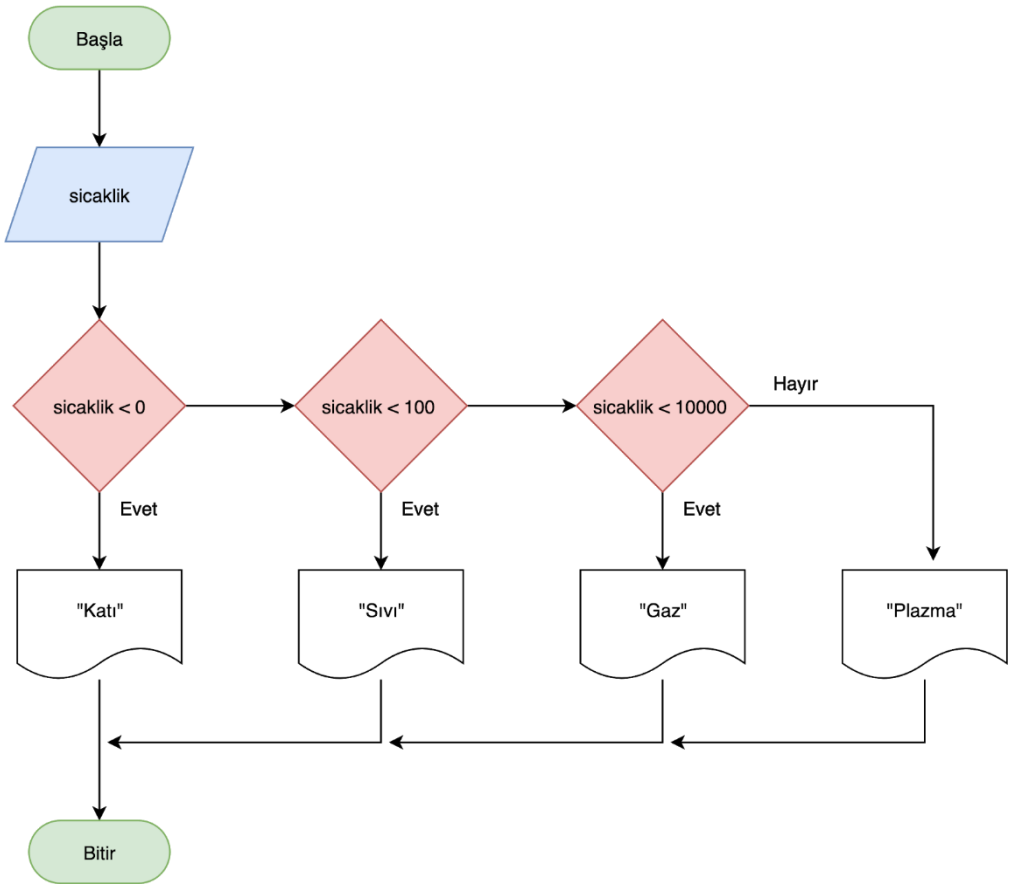
4. **Grup Ekran Çıktısı:**

   ……...

5. **Grup Ekran Çıktısı:**

   ……...

2. Görev



1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

4. **Grup Ekran Çıktısı:**

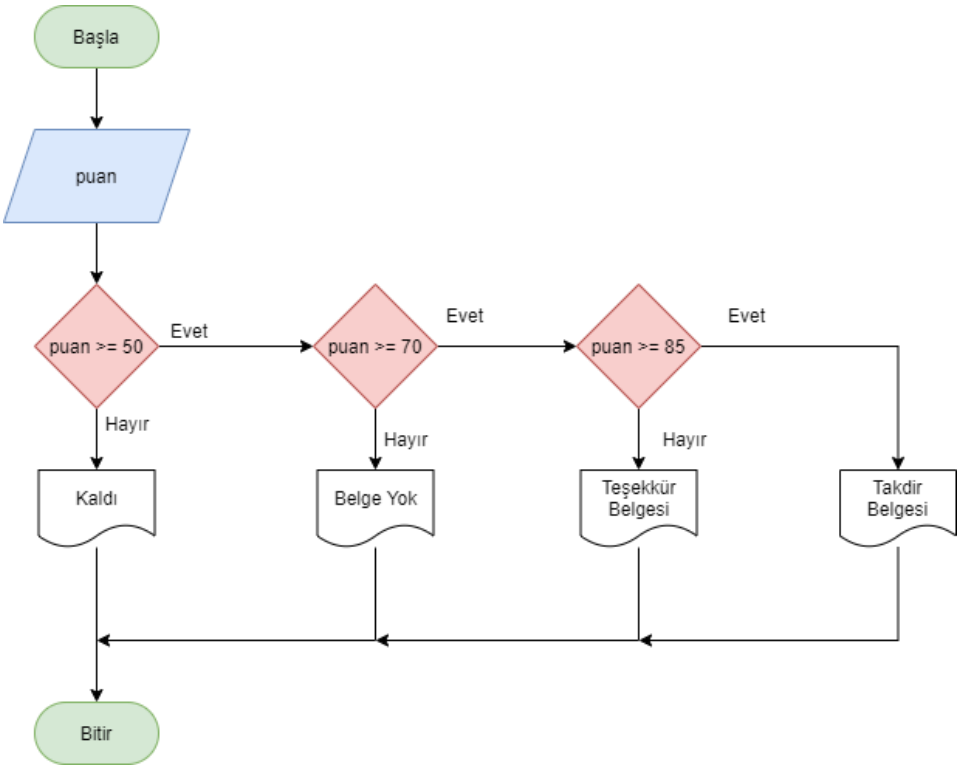
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

```
#include <iostream>
using namespace std;

int main()
{
    int sicaklik;
    cin >> sicaklik;

    if(sicaklik < 0)
        cout << "Kati";
    else if(sicaklik < 100)
        cout << "Sivi";
    else if(sicaklik < 10000)
        cout << "Gaz";
    else
        cout << "Plazma";
    return 0;
}
```

3. Görev



1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

4. **Grup Ekran Çıktısı:**

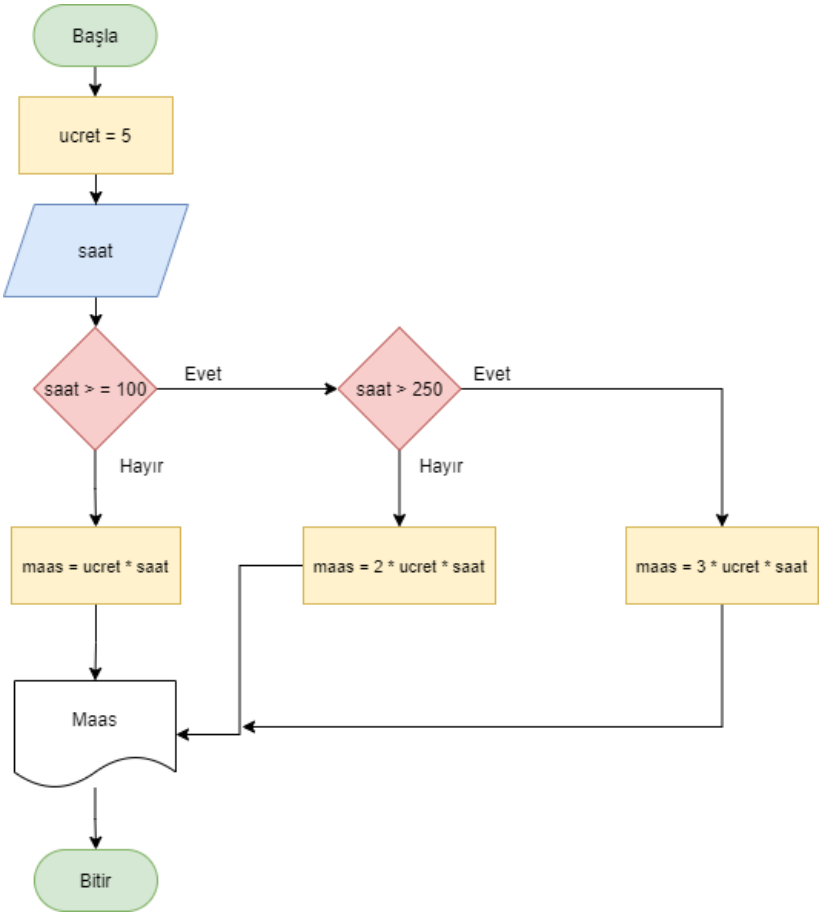
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

```
#include <iostream>
using namespace std;
int main()
{
    int puan;
    cout << "notu giriniz:";
    cin >> puan;
    if(puan<50)
        cout << "kaldı";
    else if(puan<70)
        cout << "belge yok";
    else if(puan<85)
        cout << "teşekkür belgesi";
    else if(puan<=100)
        cout << "takdir belgesi";
    else
        cout <<"Hatali giris";
    return 0;
}
```

4. Görev



1. Grup Ekran Çıktısı:

……

2. Grup Ekran Çıktısı:

……

3. Grup Ekran Çıktısı:

……

4. Grup Ekran Çıktısı:

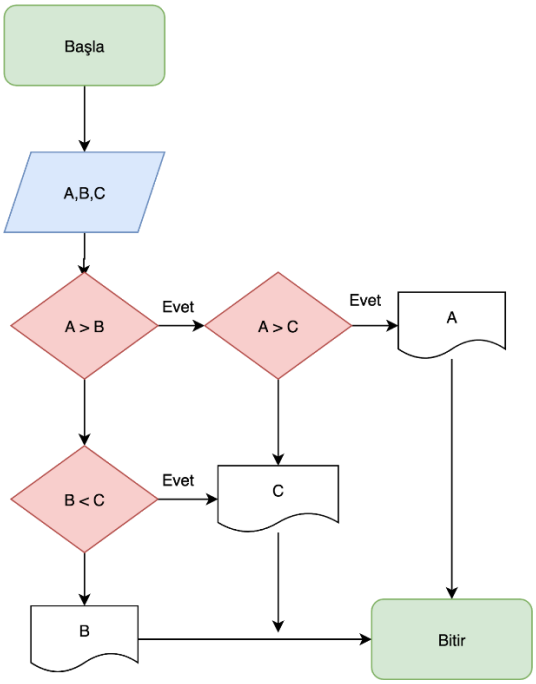
……

5. Grup Ekran Çıktısı:

……

```
#include <iostream>
using namespace std;
int main()
{
    int saat,maas;
    cout << "kac saat calisti:";
    cin >> saat;
    if(saat<100)
        maas = saat * 5;
    else if(saat<250)
        maas = saat * 5 * 2;
    else
        maas = saat * 5 * 3;
   cout << "Maasiniz: "<< maas;
    return 0;
}
```

5. Görev



1. Grup Ekran Çıktısı:

……

2. Grup Ekran Çıktısı:

……

3. Grup Ekran Çıktısı:

……

4. Grup Ekran Çıktısı:

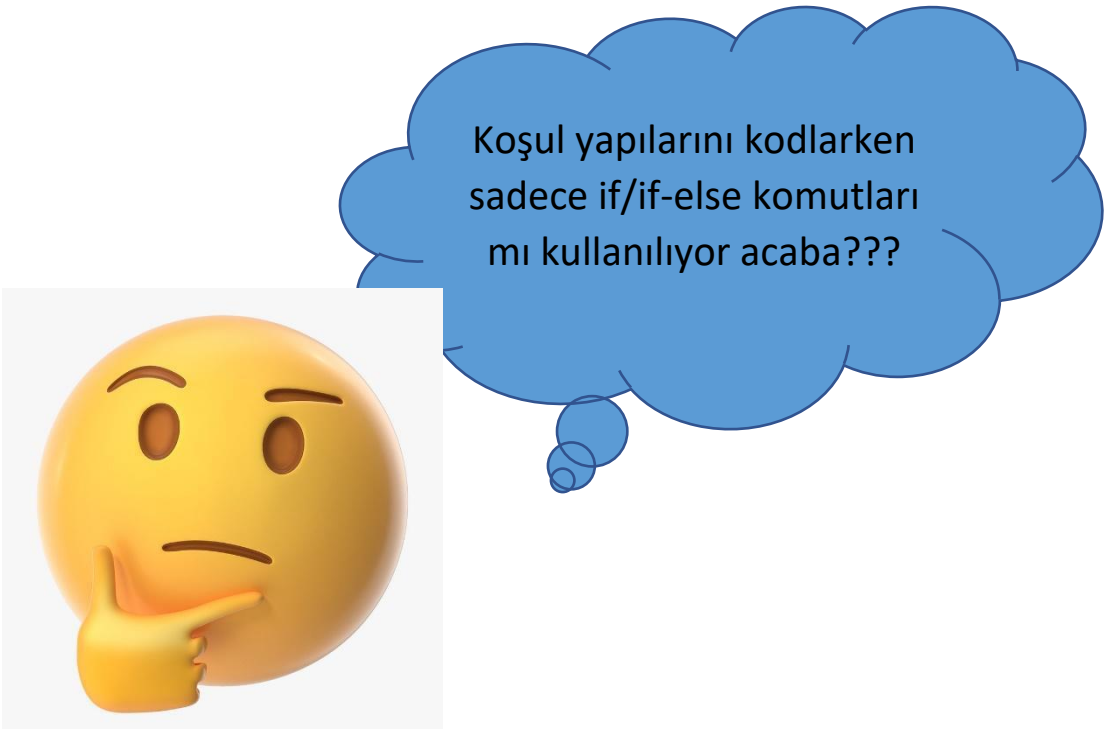
……

5. Grup Ekran Çıktısı:

……

```
#include <iostream>
using namespace std;

int main()
{
    int A,B,C;
    cout << "Sayi 1:";
    cin >> A;
    cout << "Sayi 2:";
    cin >> B;
    cout << "Sayi 3:";
    cin >> C;

    if(A>B)
    {
        if(A>C)
            cout << A;
        else
            cout << C;
    }
    else if(B<C)
    {
            cout << C;
    }
    else
    {
        cout << B;
    }
    return 0;

}
```

L4. B3.1 SWITCH Yapısı Kullanımı 1



Cevap: Birden çok koşul olduğu durumlarda if/if-else yapılarını kullanabileceğimiz gibi "switch" bloğunu da kullanabiliriz. Gelin if ile switch kullanımını hesap makinesi yapımı için gereken kodlamayı yaparak kıyaslayalım. Ardından verilen diğer örneği de beraber inceleyelim.

L4. B3.2 SWITCH Yapısı Kullanımı 2

If/ If Else Kullanımı

```
#include <iostream>
using namespace std;
int main()
{
    char islem;
    cin >> islem;
    if(islem == '+')
        cout << "Toplama islemi";
    else if(islem == '-')
        cout << "Cikarma islemi";
    else if(islem == '*')
        cout << "Carpma islemi";
    else if(islem == '/')
        cout << "Bolme islemi";
    else
        cout << "Hatali giris.";
}
```

Switch/Case Kullanımı

```
#include <iostream>
using namespace std;
int main()
{
    char islem;
    cin >> islem;
    switch(islem){
    case '+':
        cout << "Toplama islemi";
        break;
    case '-':
        cout << "Cikarma islemi";
        break;
    case '*':
        cout << "Carpma islemi";
        break;
    case '/':
        cout << "Bolme islemi";
        break;
    default:
        cout << "Hatali giris.";
    }
}
```

If/ If Else Kullanımı

```
#include <iostream>
using namespace std;

int main()
{
    int ay;
    cin >> ay;

    if(ay == 1 || ay == 2 || ay == 12)
        cout << "Kis";
    else if(ay == 3 || ay == 4 || ay ==
5)
        cout << "Ilkbahar";
    else if(ay == 6 || ay == 7 || ay ==
8)
        cout << "Yaz";
    else if(ay == 9 || ay == 10 || ay
== 11)
        cout << "Sonbahar";
    else
        cout << "Hatali giris";

    return 0;
}
```

Switch/Case Kullanımı

```
#include <iostream>
using namespace std;

int main()
{
    int ay;
    cin >> ay;

    switch(ay)
    {
    case 1:
    case 2:
    case 12:
        cout << "Kis";
        break;
    case 3:
    case 4:
    case 5:
        cout << "Ilkbahar";
        break;
    case 6:
    case 7:
    case 8:
        cout << "Yaz";
        break;
    case 9:
    case 10:
    case 11:
        cout << "Sonbahar";
        break;
    default:
        cout << "Hatali giris";
    }
    return 0;
}
```

L4. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 5. Döngü Yapıları

Kazanımlar

- K1. Problem çözme süreçlerinde döngü yapılarını kullanarak algoritma tasarlar.
- K2. Problem çözümünde döngü yapısını kullanır.
- K3. Problemin çözümünde iç içe döngü yapısını uygular.

Amaç

Bu haftanın amacı öğrencilerin programlamada döngüleri kullanarak programın akışını kontrol edebilmelerini sağlamaktır.

Önerilen Ders Akışı

A. Giriş: Sona Kalan Kazanır!(10 dk.)
B. Öğrenme Görevleri
B1. Döngüleri Tanıyalım (30 dk.)
- Ders Arası (10 dk.)

B2. Döngülerin Ne İçin Kullanıldığını Keşfediyorum (50 dk.)
- Ders Arası (10 dk.)

B3. Döngü Görevlerini Kodlayalım! (60 dk.)
- Ders Arası (10 dk.)

C. Kısmi Öğrenme Görevleri (60 dk.)

A. Giriş: Sona Kalan Kazanır!

Süre: 10 dk.

Uygulama: Bütün öğrencilerin oldukları yerde ayağa kalkmalarını ister. Basit sorular soracağını ve Eğitmen bu sorular için cevabı "evet, o benim" diyenlerin yerlerine oturmaları gerektiğini, en son ayakta kalanın ise oyunu kazanacağını belirtir. Ayakta kalan son öğrenciye sürpriz olarak küçük bir hediye verilir. Bu buz kırıcı teknik, basitten karmaşığa sorular sorularak uygulanabilir. Örneğin: Kimin erkek kardeşi var? / Kim kova burcu? / Kim ekim ayında doğdu? / Kimin gözleri kahverengi? / Kimin kolunda saat var?

Eğitmene Öneriler: Süreye bağlı olarak sorular çeşitlendirilip arttırılabilir.

B. Öğrenme Görevleri

B1. Döngüleri Tanıyalım

Süre: 30 dk.

Kazanımlar: K1. Problem çözme süreçlerinde döngü yapılarını kullanarak algoritma tasarlar..

Materyaller: L5.B1.1. Döngüleri Tanıyalım 1

L5.B1.2. Döngüleri Tanıyalım 2

Hazırlık: Eğitmen iki materyali de dört gruba dağıtmak için bir adet çıktısını alır.

Uygulama: Öğrenciler beşerli dört grup olarak çalışmaktadır. Döngülerin mantığını kavratmak için hazırlanan iki görev kâğıdının çıktısı her gruba bir adet olacak şekilde dağıtılır. Birinci görev arının tüm çiçeklerdeki nektarı almak için "ilerle" ve "nektarı al" bloklarının kaç kez kullanılması gerektiğidir. Öğrenciler blok kodları dizdikten sonra ikinci görev kâğıtları dağıtılır ve yine verilen blok kodlar kullanılarak arıların çiçeklerdeki nektarı alması için gerekli kodu yazması beklenir. Etkinliklerin cevabı aşağıda verilmiştir.

Resim 21. Çözüm 1

Resim 22. Çözüm 2

Eğitmene Öneriler: Öğrencilerin Görev 1'deki kodları ile Görev 2'deki kodları karşılaştırması istenir ve öğrencilere şu soru sorulur? Görev 1'de 1000 tane çiçek olsaydı ve arıya tüm nektarı aldıracak olsaydık 1000 kez mi ilerler ve nektarı al kodunu kullanırsınız? Öğrencilerin bu soru üzerinde tartışması sağlanarak döngünün neden kullanıldığı hakkında fikir sahibi olması sağlanır. Daha sonra eğitmen aşağıdaki bilgiler doğrultusunda döngüler konusunu özetler.

Döngüler, birden çok kez tekrar eden görevleri gerçekleştirmek için kullanılmaktadır. Tekrar sayısı önceden bilinebildiği gibi, programın çalışma sırasında güncellenebilir. Tekrar sayısı kullanıcı girişine de bağlı olabilir. Örneğin ekrana 10 kez uyarı göndermek için veya kullanıcı şifresini doğru girene kadar uyarı göndermek için döngüler kullanılabilir.

Döngü kullanmamak bir kaba 12 bardak süt dökmek için 12 tane ayrı bardak kullanmaya benzemektedir. Yapılabilir ama mantıklı değildir.

B2. Döngüleri Ne İçin Kullanıldığını Keşfetme

Süre: 50 dk.

Kazanımlar: K2. Problem çözümünde döngü yapısını kullanır.

Materyaller: L5.B2.1 Koddaki Döngülerin Ne İçin Yazıldığını Keşfediyorum

L5.B2.2 Döngü Çeşitleri Afişi

Hazırlık: Eğitmen L5. B2. 2 kodlu materyali ÖYS üzerinden erişime açar. Diğer L5. B2. 1. kodlu materyalden ise 10 adet çıktı alır.

Uygulama: Her iki öğrenciye bir tane verilecek şekilde "L5. B2.1 Koddaki Döngülerin Ne İçin Yazıldığını Keşfediyorum." adlı materyal öğrencilerle paylaşılır. Öğrencilerden verilen kodun hangi problemi çözmek için veya hangi amaca yönelik olduğunu bulup yazmaları istenir. Eğitmen destekleyici bilgi olarak döngüler için hazırlanan afişi öğrencilerle paylaşarak döngüler hakkında aşağıdaki gibi bir özetleme yapar.

Döngüleri C++ programlama dilinde "for", "while" ve "do-while" komutları ile gerçekleştirebiliriz. Bir döngüyü istediğimiz komut ile yapabiliriz. Üç farklı komut olmasının nedeni ise bazı işlemleri bazı döngüler ile yapmak daha kolay olmaktadır. Döngüler için oluşturulan afiş öğrencilerle paylaşılır.

Eğitmene Öneriler: Yukarıdaki kodlamalar ve problem bulma bittikten sonra döngüler aşağıdaki gibi özetlenir.

Bir döngüde olması gerekenler şunlardır;

1) Kontrol değişkeni: Döngü başlangıç ve bitişi hangi değişken üzerinden kontrol edilecek.
2) Değişken başlangıç değeri
3) Bitiş koşulu
4) Değişken güncellemesi: Her tekrar sonrasında kontrol değişkeni nasıl etkilenecek?
5) Döngü gövdesi: Döngü içerisinde gerçekleştirilecek işlemler.

B3. Döngü Görevlerini Kodlayalım!

Süre: 60 dk.

Kazanımlar: K3. Problemin çözümünde iç içe döngü yapısını uygular.

Materyal: L5.B3. 1 Döngü Görevlerini Bilgisayarlarımızda Kodlayalım !

Hazırlık: Eğitmen materyali ÖYS üzerinden erişime açar. Code Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Eğitmenler döngü görevi etkinliğindeki uygulamalarından iki tanesini öğrencilerin seçmesini sağlayarak, öğrencilerin bu derste kodlama yapmasını ister. Öğrenciler kodlama esnasında daha önceki derste hazırlanan afişi destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmenlerden biri sınıfı dolaşarak, biri de kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler: Eğitmen görevleri öğrencilere dağıtmadan önce C++ programlama dilinde nasıl rastgele sayı üretildiği hakkında aşağıdaki gibi çok kısa bir bilgi verir.

C++ programlama dilinde rastgele sayı üretmek için rand() hazır fonksiyonu kullanılır. Bu fonksiyon 0 ila üst sınır (en az 32767 en çok RAND_MAX) arasında rastgele sayı üretir. Üst sınırı sınırlandırmak için mod (%) operatörü kullanılır. 0-100 (100 hariç) arasında rastgele sayı üretmek istersek rand()%100 şeklinde kullanırız. Alt sınırı arttırmak/azaltmak istersek toplama işlemini kullanırız. Örneğin 10 ile 100 arasında rastgele sayı üretmek için 10 + (rand() % 90) şeklinde kullanırız. srand() fonksiyonu rand() fonksiyonu için hazırlayıcı fonksiyondur. Rastgele sayı üreteci için kullanılacak başlangıç değerini ayarlar. Eğer srand() kullanılmaz ise program her çalıştırıldığında aynı rastgele değerler elde edilir. "srand(time(0));" satırı main bloğunun başına eklenerek kullanılabilir.

Döngü görevleri ve cevapları aşağıdaki gibidir:

Görev 1: Ahmet okul kütüphanesindeki raflara herkesin kolayca kitapları bulabilmesi için sayı etiketleri yapıştırmak istiyor. Kütüphanede 100 tane raf olduğu düşünülürse Ahmet'in 1'den 100'e kadar sayıları sıralayıp ekranda göstermesi gerekmektedir. Buradan hareketle Ahmet'in nasıl bir kod yazması gereklidir, bilgisayarımızda kodlayalım.

Cevap 1:

```
int sayi = 1;
    while(sayi<100)
    {
        cout << sayi <<endl;
        sayi ++;
    }
```

Görev 2: Ali kardeşi Buğra'nın 1'den 100'e kadar 7'şerli olarak sayı saymasını istemektedir. Buradan hareketle kardeşinin doğru sayıp saymadığını kontrol etmesi için bilgisayarda 1'den 100'e kadar küçükten büyüğe olacak şekilde 7'ye bölünen sayıları ekranda göstermek istiyor. Bunun için nasıl bir kod yazmalı?

Cevap 2:

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1; i < 100; i++)
    {
        if(i%7==0)
            cout << i <<endl;
    }
```

```
    return 0;
}
```

Görev 3: Rafet öğretmen sınıfında bulunan 10 öğrencinin matematik dersinde aldığı notları klavyeden teker teker girerek sınıfın matematik dersi not ortalamasını bulan bir program yazmak istiyor. Bunun için nasıl bir kod yazmalıdır?

Cevap 3:

```
#include <iostream>
using namespace std;
int main()
{
    int toplam = 0;
    for(int i=0; i < 10; i++)
    {
        int puan;
        cout << i+1 <<". ogrenci puani:";
        cin >> puan;
        toplam += puan;
    }
    cout << "ortalama : " << toplam /10 ;
}
```

Görev 4: Defne öğretmen, aldığı 10 tane kitabı sınıfındaki öğrencilere çekiliş yoluyla dağıtmak istemektedir. Sınıftaki öğrencilerin okul numaraları 50 ile 100 arasındadır (100 hariç). Defne öğretmen bunun için 50 ile 100 arasında 10 tane rastgele bir sayı üreten program yazarak kitapları okul numarası rastgele çıkan öğrencilere verecektir. Bunun için nasıl bir kod yazmalıdır?

Cevap 4:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    srand(time(0));
    for(int i=0; i<10;i++)
        cout << 50 + rand() % 50 <<endl;
}
```

Görev 5: Duru öğretmen öğrencilerine 1'den 9'lara kadar olan çarpım tablosunu öğretmek için bilgisayardan ekran çıktısı alıp yazdırmak istemektedir. Bunun için program yazmak isteyen Duru öğretmen bilgisayarda nasıl kodlamalıdır?

Bu görev öncesi öğrencilere iç içe döngülerin kullanımı hakkında giriş yapılarak nasıl kodlanacağı hakkında temel bilgiler gösterilir.

Not: Birden fazla kontrol değişkeni kullanmamız gerekir ise, iç içe döngüleri kullanırız. Bunun için en güzel örnek çarpım tablosu olabilir.

Cevap 5:

```
#include <iostream>

using namespace std;

int main()

{

    for(int i=1;i<10;i++)

    {

        for(int j=1; j<10;j++)

        {

            cout << i << "*" << j << "=" << i*j <<"\t";

        }

        cout << endl;

    }

}
```

C. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L5.C.1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

1) Kodlayıcı

Klavyeden girilen 10 tam sayının toplamını ekrana yazdıran kodu nasıl yazarsınız?

```
 #include <iostream>
#include<cstdlib>
using namespace std;
int main()
{
    int toplam=0;
    for(int i=0;i<10;i++)
    {
        int sayi;
        cin >> sayi;
        toplam += sayi;


    }
     cout << "toplam:" << toplam;
}
```

2) Kodlayıcı

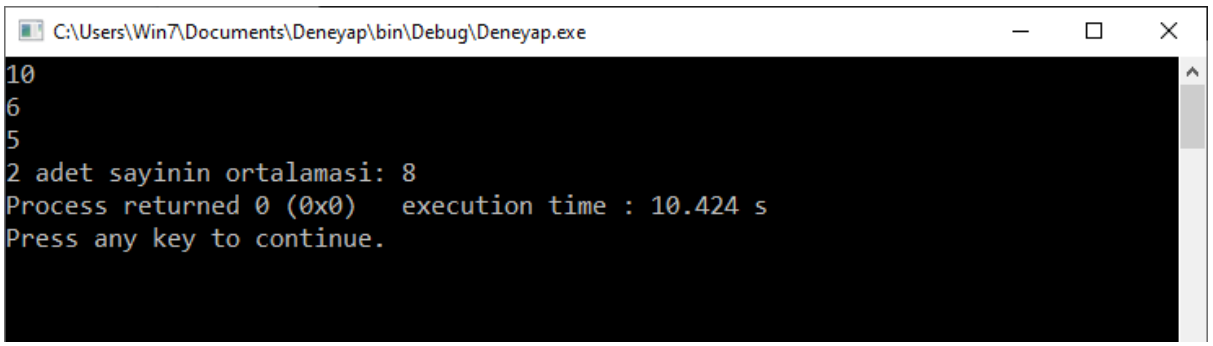
0-100 arasında rastgele üretilen 10 tam sayıdan tek olanların adedini ve toplamını ekrana yazdıran kodu nasıl yazardınız?

```
 #include <iostream>
#include<cstdlib>
#include<ctime>
using namespace std;
int main()
{
    srand(time(0));
    int adet=0,toplam=0;
    for(int i=0;i<10;i++)
    {
        int rastgele_sayi = rand()%100;
        if(rastgele_sayi % 2 == 1)
```

```
        {
            adet++;
            toplam += rastgele_sayi;
        }
    }
     cout << adet << " adet tek sayinin toplami:" << toplam;
}
```

3) Kodlayıcı

Klavyeden çift sayı girildikçe toplama işlemi yapan, tek sayı girildiği durumda ise girilen çift sayıların ortalamasını gösteren programı yazalım.



Resim 23. *Ekran çıktısı*

```
#include <iostream>
using namespace std;
int main()
{
    int sayi,toplam = 0, sayac = 0;
    do
    {
        cin >> sayi;
        if(sayi %2 == 0)
        {
            toplam += sayi;
            sayac++;
        }
    }while(sayi %2 == 0);
    cout << sayac << " adet sayinin ortalamasi: "<<toplam/sayac;
}
```

4) Kodlayıcı

1-100 arasında üçe bölünüp yediye bölünmeyen sayıların miktarını bulan kodu nasıl yazardınız?

```
#include <iostream>
using namespace std;
int main()
{
    int adet=0;
    for(int i=0;i<10;i++)
    {
        if (i%3 == 0 && i%7 != 0)
             adet++;
    }
     cout << adet << " sayi vardir.";
}
```

Hafta 5. Ders Materyalleri

L5. B1.1. Döngüleri Tanıyalım Görev 1

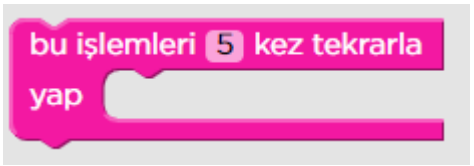
** https://studio.code.org/s/express-2020/lessons/22/levels/1?section_id=2984849*

Yukarıdaki arının tüm çiçeklerdeki nektarı almasını isteseydiniz **blok** kodlardan hangisini kaç kere kullanırdınız?

CEVAP:

** https://studio.code.org/s/express-2020/lessons/22/levels/1?section_id=2984849*

L5. B1.2. Döngüleri Tanıyalım Görev 2 (Yazılımcılar programlarında neden döngüleri kullanmayı çok severler? ☺)



** https://studio.code.org/s/express-2020/lessons/22/levels/1?section_id=2984849*

Yukarıdaki arının tüm çiçeklerdeki nektarı almasını isteseydiniz **blok** kodlardan hangisini kaç kere kullanırdınız?

CEVAP:

L5.B2.1 Koddaki Döngülerin Ne İçin Yazıldığını Keşfediyorum

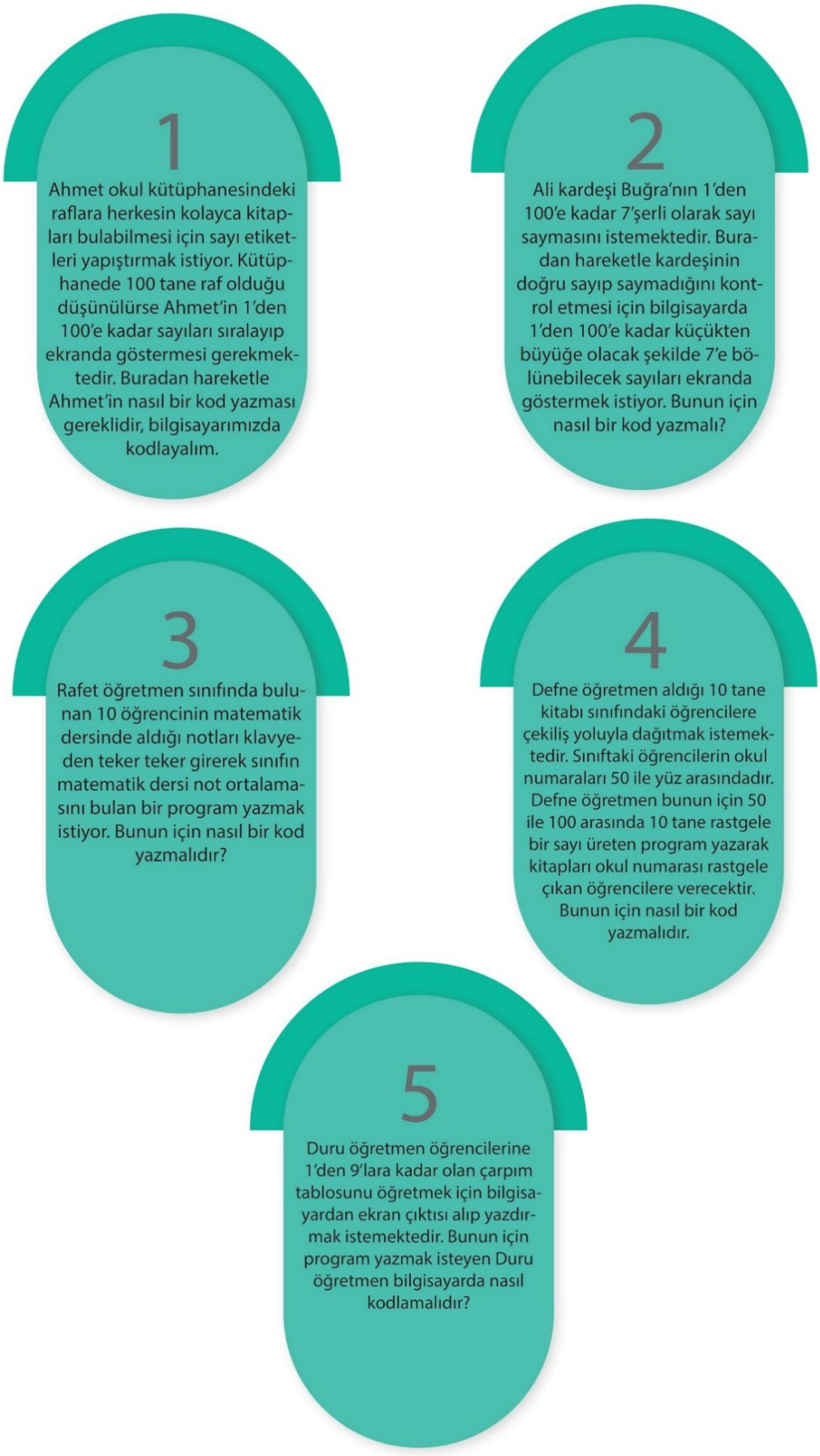
Kod	Problem Sol kısımda bulunan kodlar hangi temel problem ya da amaç için yazılmış olabilir? Bu sütunda belirtiniz.
for(int sayi=0;sayi<10;sayi++)	
for(int sayi=10;sayi>=0;sayi--)	
for(int sayi=10;sayi>=0;sayi-=2)	
for(int i=15;i>=0;i-=3)	
`#include <iostream>` `using namespace std;` `int main()` `{` `for(int i=0;i<10;i++)` `cout << "DENEYAP" << endl;` `return 0;` `}`	
`#include <iostream>` `using namespace std;` `int main()` `{` `for(int i=1; i<10; i++)` `{` `cout << i <<endl;` `}` `return 0;` `}`	
`int sayi = 1;` `while(sayi<100)` `{` `cout << sayi <<endl;` `sayi ++;` `}`	

`int i;` `for(i=1;i<20;i++)` `if(i%2==1)` `cout << i << endl;`	
`int i=1;` `while(i<20)` `{` `if(i%2==1)` `cout << i << endl;` `i++;` `}`	
`int i=1;` `do` `{` `if(i%2==1)` `cout << i << endl;` `i++;` `}while(i<20);`	

L5. B2.2 Döngü Çeşitleri Afişi

Resim 24. Ekran çıktısı

L5. B3. 1 Döngü Görevlerini Bilgisayarlarımızda Kodlayalım!



L5. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 6. Diziler ve Katarlar

Kazanımlar

- K1. C++ programlamada dizi kavramını açıklar.
- K2. C++ programlamada tek boyutlu ve çok boyutlu dizileri kullanır.
- K3. C++ programlamada dizilere farklı türlerde değerler atar.
- K4. Dizileri döngü içinde kullanır.
- K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.
- K6. Diziler ve katarlar arasındaki farkı ayırt eder.

Amaç

Haftanın amacı, tek boyutlu ve çok boyutlu dizi tanımlama ve diziler üzerinde farklı işlemlerin gerçekleştirilmesi ile örnek çözümlerin öğretilmesi amaçlanmaktadır. Katarların tanımlanması ve kullanımını uygulayarak dizilerden farkını öğrenir.

Önerilen Ders Akışı

A. Giriş: Ekip İşi (15 dk.)
B. Öğrenme Görevleri
   B1. Dizileri Tanıyalım! (20 dk.)

   B2. Dizilere Değer Verelim! (25 dk.)

   - ❏ Ders Arası (5 dk.)
   - ❏ Motivasyon Oyunu (10 dk.)

   B3. Döngülerle Diziler (20 dk.)

   B4. Dizilerle Kodlayalım (25 dk.)

   - ❏ Ders Arası (5 dk.)

   B5. Kodlama Ekibi (25 dk.)

   B6. Farkı Bul (25 dk.)

   - ❏ Ders Arası (5 dk.)
   - ❏ Motivasyon Oyunu (10 dk.)

C. Kısmi Öğrenme Görevleri (50 dk.)

A. Giriş: Ekip İşi

Süre: 15 dk.

Uygulama: Oyuncular isimlerinin alfabetik sırasına göre sıraya dizilir ve sıradan devam edecek şekilde beşerli gruplar oluşturur. Çeşitli görevlerin yazılı olduğu liste oyuncuların rahatça görebileceği bir yere asılır ya da görev listesi tahtaya yazılır. Oyunculardan 10 dakika içinde listedeki tüm görevlerin bitmiş olması istenir. Saati rahat takip etmeleri için kronometre kullanılabilir. Oyunculara, görev dağılımlarını nasıl yapacakları gibi konularda kendilerinin karar vermeleri istenir. Süre bitiminde, listedeki görevlerin nasıl tamamlandığına bakılır. Gruba ve süreye bağlı olarak görev sayısı ve görevler değişebilir.

Örnek liste:

*Grubun burç haritasını çıkarın.

*Grubun uzmanlık alanlarını sıralayın.

*Grubun ortalama yaşını bulun.

*Herkesin dahil olduğu yaratıcı bir fotoğraf çekilin.

*Kolaylaştırıcılara bir iyilik yapın.

(Kaynak: https://app.ogrenmetasarimlari.com)

B. Öğrenme Görevleri

B1. Dizileri Tanıyalım!

Süre: 20 dk.

Kazanımlar: K1. C++ programlamada dizi kavramını açıklar.

K2. C++ programlamada tek boyutlu ve çok boyutlu dizileri kullanır.

Materyaller: L6. B1. 1. Destekleyici Bilgi: Dizileri Tanıyalım!

Çeşitli madenî paralar

Hazırlık: Materyal ÖYS üzerinden erişime açılır.

Uygulama: Eğitmen öğrencileri ikili gruplar hâlinde çalıştırır. Sınıfta iç içe geçmiş iki çember oluşturulur. İç çemberdekiler tek boyutlu dizi, dış çemberdekiler ise çok boyutlu dizi temsilcileridir. Materyaldeki destekleyici bilgiler ikiye ayrılarak iç çemberdekilere tek boyutlu dizi, dıştakilere ise çok boyutlu dizi materyali dağıtılır. 1 dk. kadar temsilcilerin destekleyici bilgileri incelemeleri istenir. Daha sonra 1 dk. süre içinde iç çemberdekiler, dıştakilere tek boyutlu diziler hakkında anladıklarını aktarır. Eğitmen süre bitimini hatırlatmak için bir çan ya da zil kullanabilir. Daha sonra dış çemberdekilere sıra gelir ve onlar da 1 dk. süre içinde iç çemberdekilere çok boyutlu dizileri açıklamaya çalışır.

Öğrencilerin birbirlerine açıklama yapmalarının ardından eğitmen dış çemberin saat yönünde birer öğrenci yana kayacak şekilde dönmesini ister. Bu şekilde öğrenci çiftleri

değiştirilir. Her çifte çeşitli büyüklüklerde (bir liradan 3 tane, 50 kuruştan 4 tane gibi...) madenî para dağıtılır. Öğrenci çiftlerinden madenî paralar ile tek ve çoklu boyutlarda diziler oluşturmaları istenir. Ayrıca dizilerin boyutlarını değiştirerek, bunları boyut bilgileri ve indis numaraları ile etiketlemeleri beklenir. Etkinlik sonunda eğitmen grup çalışmalarını takip eder ve özellikle hatalı diziler üzerinden örnekler alarak konuyu özetler.

Eğitmene Öneriler: Etkinlik materyali olarak madenî para yerine farklı renk ve büyüklükte minik ponponlar kullanılabilir. Çember konumunda sınıf düzeni, karşılıklı dikdörtgen şeklinde ya da sınıfın ikiye ayrılması ile de planlanabilir. Bu sınıf düzeninde amaç öğrencilerin grup oluşturma hareketini kolaylaştırmaktır.

B2. Dizilere Değer Verelim!

Süre: 25 dk.

Kazanımlar: K2. C++ programlamada tek boyutlu ve çok boyutlu dizileri kullanır.

K3. C++ programlamada dizilere farklı türlerde değerler atar.

Materyaller: L6. B2. 1. Destekleyici Bilgiler: Dizilere Değer Verelim!

L6. B2. 2. Görev Kartları: Dizilere Değer Verelim!

Hazırlık: Öğrenciler B1 etkinliğindeki oturma düzeninde iç içe çember şeklinde oturmaya devam eder. Birinci materyal ÖYS üzerinden erişime açılır. İkinci materyalden 10 adet çıktı alınır ve her biri ikiye ayrılır.

Uygulama: Eğitmen süreci B1 etkinliğindeki gibi yönetir. İç çemberdekilere çok boyutlu dizilere değer atama materyali verilirken, dış çemberdekilere tek boyutlu dizilere değer atama materyali verilir. Çiftler birbirlerine birer dakika arayla materyaldekileri açıklamaya çalışır. İlk olarak bir dakika kadar temsilcilerin destekleyici bilgileri incelemeleri istenir. Daha sonra 1 dk. süre içinde iç çemberdekiler, dıştakilere tek boyutlu diziler hakkında anladıklarını aktarır. Eğitmen süre bitimini hatırlatmak için bir çan ya da zil kullanabilir. Daha sonra dış çemberdekilere sıra gelir ve onlar da 1 dk. süre içinde iç çemberdekilere çok boyutlu dizileri açıklamaya çalışır.

Öğrencilerin birbirlerine açıklama yapmalarının ardından eğitmen dış çemberin saat yönünde birer öğrenci yana kayacak şekilde dönmesini ister. Birleşen yeni çiftlere görev kartları verilir. Bu sefer görev kartları verilirken, iç çembere tek boyutlu, dış çembere çok boyutlu dizi görev kartları verilir. Ancak öğrencilerden bu kartlarda yer alan görevleri tamamlarken birbirlerine yardım etmeleri gerektiği belirtilir. Öğrenciler görevleri tamamladıktan sonra kartların üzerine grup cevaplarını ve isimlerini yazarak, eğitmene teslim eder.

Eğitmene Öneriler: Eğitmen aralıklı olarak görevlerin yapılma aşamasında öğrencileri takiptedir. Gerektiğinde yanlış öğrenme ya da açıklamaları engellemek ya da öğrenci sorularını yanıtlamak için anlık geri bildirimlerde bulunur.

B3. Döngülerle Diziler

Süre: 20 dk.

Kazanımlar: K4. Dizileri döngü içinde kullanır.

Materyaller: L6. B3. 1. Destekleyici Bilgi: Döngülerle Diziler

L6. B3. 2. Görev Kartı: Döngülerle Diziler

Hazırlık: Öğrenciler B2 etkinliğindeki oturma düzenindedir. Materyaller ÖYS üzerinden öğrenci erişimine açılır.

Uygulama: Öğrenciler B2 etkinliğinde oturdukları gibi çiftler hâlinde B3 etkinliğine devam etmektedir. İç çemberdekiler çok boyutlu diziyi, dıştakiler ise tek boyutlu diziyi temsil eder. Bu sefer öğrenciler çemberlerin karşısındaki arkadaşı ile değil de yanında oturan arkadaşıyla eşleşir. Tüm öğrencilerden Öğrenme Yönetim sisteminde erişime açılan etkinliğin iki materyali açmaları istenir. İlk olarak öğrencilerin 2 dakika kadar birinci materyali incelemeleri istenir. Eğitmen burada aşağıdaki açıklamayı yapar.

> *Dizilere ilk değer atamanın diğer bir yolu da döngüleri kullanmaktır. Bunu gerçekleştirmek için, önce diziyi normalde yaptığımız gibi tanımlar ve daha sonra oluşturacağımız döngü içerisinde istediğimiz değerleri atarız. Bunun için ilk materyaldeki örneği inceleyiniz.*

Daha sonra dış çemberde yan yana oturan çiftlere Görev dış çember, iç çemberde yan yana oturan çiftlere ise Görev iç çember üzerinde çalışmaları söylenir. Çiftlerin 5 dk. görev üzerinde çalışmaları beklenir. Görev tamamlama aşamasında destekleyici bilgiden yararlanmaları istenir. Eğitmenler anlık geri bildirimler ve görev üzerinde çalışmaya motive etmek için öğrencileri takip etmelidir. Görevler üzerinde çalışma süresi bittikten sonra, öğrencilere "Şimdi karşınızdaki arkadaşınızla eşleşin ve çalıştığınız görevler hakkında birbirinizle tartışın. Kodunuz üzerinde hatalı olan noktalar varsa, bunu birlikte düzeltmeye çalışın" denir. Burada öğrencilerin çiftler hâlinde yaptıklarını artık karşılarındaki arkadaşlarıyla paylaşmaları istenir. Bunun için tekrar 5 dk. süre verilir. Süre sonunda eğitmen tahtaya görevlerin doğru yanıtlarını yazar. Eğitmen çiftlerin çalışmalarını gezerek takip eder.

Eğitmene Öneriler: Eğitmen doğru yanıtları öğrenme yönetim sistemi üzerinden de paylaşabilir. Öğrencilerin yaptıkları görevlerin doğru yanıtları aşağıdaki gibidir:

Görev İç Çember Yanıtı	Görev Dış Çember Yanıtı
`#include <iostream>` `using namespace std;` `int main()` `{` `int d_yili[6] = {2005, 2004, 2003, 2008, 2006, 2002};` `int i;` `for(i=0; i<6; i++){` `cout << i+1 <<". arkadasimin dogum yili: " << d_yili[i] << endl;` `}` `return 0;` `}`	`#include <iostream>` `using namespace std;` `int main(){` `int d_yili[2][3] = {{2005, 2004, 2003},` `{2008, 2006, 2002}};` `int k = 1;` `for(int i=0; i<2;i++){` `for(int j=0; j<3; j++,k++){` `cout << k << ". arkadasimin dogum yili: " << d_yili[i][j] << endl;` `}` `}` `return 0;` `}`

B4. Dizilerle Kodlayalım!

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: L6. B4. 1. Görev Kodu

Hazırlık: Öğrenciler B3 etkinliğindeki gibi iç içe çember şeklinde oturur ve çiftler hâlinde çalışırlar. Materyaller kesikli çizgiden kesilerek bir torba ya da fanus içerisine atılır. Bir görevden iki tane çıktı alınarak, beş görev toplamda 10 göreve çıkarılır.

Uygulama: Eğitmen, çiftlere torba içinden bir görev kodu seçmelerini ister ve aşağıdaki talimatı verir.

> *Kod içindeki değişkenleri ayırt edin. Bu değişkenlerin nasıl bir veri tutabileceğini düşünün ve onlara daha anlamlı isimler vererek kodu değiştirin. Değiştirdiğiniz kodu bilgisayarda çalıştırıp çıktısını inceleyin. Bu şekilde görev kodunun günlük hayatta hangi probleme çözüm üretebileceği konusunda bir öneri getirin.*

Öğrenciler ikili gruplar hâlinde çıkan görev üzerinde 5 dk. kadar çalışır. Torbada toplam 10 görev kodu olmasına rağmen aynı görev kodundan iki tane bulunmaktadır. Bu nedenle beş dk. sonra aynı kod üzerinde çalışan çiftlerin bir araya gelmeleri istenir. Bu şekilde dörtlü grup olan

öğrenciler çalıştıkları görev üzerinde yaptıklarını birbirlerine aktarır. Eğitmen dörtlü grupların 3-5 dk. kadar kendi aralarında tartışmalarına izin vermelidir. Süre sonunda grupların yaptıklarını incelemek için onları dinler, sorularını cevaplar ve yanlış öğrenmelerin önüne geçer. Eğitmen öğrencilerden tüm grupların üzerinde çalıştıkları görevleri ve çözümlerini Öğrenme Yönetim Sisteminde paylaşmalarını ister. Öğrencilerden üzerinde çalıştıkları görev dışında kalan diğer dört görevi evde tamamlamaları istenebilir.

Eğitmene Öneriler: Verilen görevlerin doğru yanıtları aşağıdadır:

```
#include <iostream>

using namespace std;

int main(){

    int d1[4], d2[4], d3[4];

    int i;

    for(i=0; i<4;i++){

        cout << "1. Dizinin " << (i+1) <<
". elemanini giriniz: ";

        cin >> d1[i];

    }

    cout << endl;

    for(i=0; i<4;i++){

        cout << "2. Dizinin " << (i+1) <<
". elemanini giriniz: ";

        cin >> d2[i];

    }

    cout << endl;

    for(i=0; i<4;i++){

        d3[i] = d1[i] * d2[i];

        cout << "3. Dizinin " << (i+1) <<
". elemani: " << d3[i] << endl;

    }

    return 0;

}
```

Görev Kodu 1

Yandaki program iki dizinin karşılıklı olarak elemanlarını çarpıp yeni bir diziye kaydetmeyi amaçlamaktadır.

```
#include <iostream>
```

```
using namespace std;
int main(){
    float dizi[] = {19.23, 26.43, 14.72, 28.71,
15.04, 10.06, 22.96};
    int i, y, el_sayisi = 7;
    float x;
    cout << "Dizi: ";
    for(i=0; i < el_sayisi; i++)
        cout << dizi[i] << " ";
    x = dizi[0];
    y = 0;
    for(i=1; i < el_sayisi; i++){
        if(x > dizi[i]){
            x = dizi[i];
            y = i;
        }
    }
    cout << x <<" ve "<< y;
  return 0;
}
```

Görev Kodu 2

Yandaki program dizideki en küçük elemanı ve bu elemanın indis numarasını yazdırmayı amaçlamaktadır.

```
#include <iostream>
using namespace std;
int main(){
    int sayilar[] = {5, 3, 2, 5, 1, 2, 6, 9, 8, 1};
    int i, j, boyut = 10;
    cout << "Dizi: ";
    for(i = 0; i < boyut; i++)
        cout << sayilar[i] << " ";
    cout  << "\nElemanlar: ";
    for(i = 0; i < boyut-1; i++)
        for(j = i+1; j < boyut; j++)
            if(sayilar[i] == sayilar[j])
                cout <<  sayilar[i] << " ";
    return 0;
}
```

Görev Kodu 3

Yandaki program dizide tekrar eden elemanları bulmayı amaçlamaktadır.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int sayilar[100];
    int i, n, bol3=0, bol5=0;
    cout << "Eleman sayisini girin : ";
    cin >> n;
    cout<<"\nDizi elemanlarini girin :\n";
    for(i=0; i<n; i++){
        cout << "Elemani girin dizi[" << i << "] :
";
        cin >> sayilar[i];
    }
    for(i=0; i<n; i++){
        if(sayilar[i]%3==0)
            bol3++;
        if(sayilar[i]%5==0)
            bol5++;
    }
    cout << bol3 << endl;;
    cout << bol5 << endl;;
}
```

Görev Kodu 4

Yandaki program girilen dizide 3'e bölünebilen ile 5'e bölünebilen sayıların adedini ayrı ayrı ekrana yazdırmayı amaçlamaktadır.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int dizi[100];
    int i, boyut, tek=0, cift=0;
    cout << "Eleman sayisini girin : ";
    cin >> boyut;
    cout<<"\nDizi elemanlarini girin :\n";
    for(i=0; i<boyut; i++){
        cout << "Elemani girin dizi[" << i <<
"] : ";
```

Görev Kodu 5

Yandaki program girilen dizide tek ve çift sayıların adedini ekrana yazdırmayı amaçlamaktadır.

```
        cin >> dizi[i];
    }
    for(i=0; i<boyut; i++){
        if(dizi[i]%2==0)
            cift++;
        else
            tek++;
    }
    cout << cift << endl;;
    cout << tek << endl;;
}
```

B5. Kodlama Ekibi!

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: L6. B5. 1. Problem

L6. B5. 2. Görev Kartı

Hazırlık: Öğrenciler B3 etkinliğindeki gibi iç içe çember şeklinde oturur. Problem öğrenme yönetim sisteminden materyal öğrenci erişimine açılır. İkinci materyalden ise 10'ar tane çıktı alınır. Bu materyal kesikli çizgilerden kesilerek kart hâlinde hazırlanır.

Uygulama: Eğitmen bu etkinlikte öğrencilerin kodlama ekibi oluşturarak, bir problemi birlikte kodlamalarını bekler. Bunun için dört kişiden oluşan grupları kendi içlerinde çiftlere ayırır ve iş bölümü yaptırır. Eğitmen ilk etapta öğrencileri iç içe geçmiş çember şeklinde oturtur. İç çemberde yer alanlar çiftler hâlinde eşleşmiştir. Dış çemberdekiler de aynı şekildedir. İç çemberdekilere iç çember görevi, dıştaki çiftlere ise dış çember görevi verilir. 5 dk. kadar öğrenciler görevler üzerinde çalışır. Daha sonra eğitmen çiftlerin karşılarında oturan dış çember çiftiyle eşleşmelerini ister. Bu şekilde gruplar artık dört kişiye ulaşmıştır. Eğitmen öğrencilere aşağıdaki açıklamayı yapar.

İç çember ve dış çember görevleri birbirini tamamlayan kod yazma görevidir. İki görev birleştirilince bir problemin kodlarını oluşturmaktadır. Probleme öğrenme yönetim sisteminden erişebilirsiniz. Görevleri yaparken dikkat edeceğiniz önemli bir nokta ise şudur: Problem tek olduğu için değişken isimlerinin ortak olmasına dikkat edin.

5 dk. kadar öğrenciler görevler üzerinde çalışır. Daha sonra eğitmen çiftlerin karşılarında oturan dış çember çiftiyle eşleşmelerini ister. Bu şekilde gruplar artık dört kişiye ulaşmıştır. Dört kişilik ekiplere kodlama ekibi görev kartı verilir. Artık bu grup, temel problemi çözecek kodlama ekibini oluşturur. Kodlama ekipleri yazdıkları kod satırlarını birleştirerek bir araya

getirir ve kodu bilgisayar ortamında çalıştırır. Eğitmen sonuca ulaşamayan grupların, tamamlayan gruplardan destek almalarını sağlar.

Eğitmene Öneriler: İç çember ve dış çember grupları, sınıfın düzenine göre öğrenci mevcudunun ikiye bölünmesi ile grup 1 ve grup 2 şeklinde de oluşturulabilir. Problemin kodlarını aşağıda bulabilirsiniz. Dört kişilik grup toplandığında, turuncu kod satırlarına iç çemberdekilerin, yeşil kod satırlarına dış çemberdekilerin erişmesi beklenir. Sarı kod satırları ve diğer tamamlamalar ise kodlama ekibinin görevidir.

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int i, j, sat, sut, matris1[5][5], matris2[5][5];
6.      sat = 5;
7.      sut = 5;
8.
9.    cout << "Matrisin elemanlarini sirayla giriniz: " << endl;
10.   for(i =0; i < sat; i++) {
11.       for(j=0; j < sut; j++) {
12.           cin >> matris1[i][j];
13.       }
14.   }
15.   for(i=0; i < sat; i++) {
16.       for(j=0; j < sut; j++) {
17.           matris2[j][i] = matris1[i][j];
18.       }
19.   }
20.   cout<<"Matrisin Yer Degistirilmis Hali: " << endl;
21.   for(i=0; i < sut; i++) {
22.       for(int j=0; j < sat; j++) {
23.           cout << matris2[i][j] << " ";
24.       }
25.       cout << endl;
26.   }
27. }
```

B6. Farkı Bul!

Süre: 25 dk.

Kazanımlar: K6. Diziler ve katarlar arasındaki farkı ayırt eder.

Materyaller: L6. B6. 1. Farkı Bul!

Hazırlık: Öğrenciler ikili gruplar hâlinde oturmaktadır. ÖYS üzerinden materyal öğrenci erişimine açılır.

Uygulama: Öğrenciler materyali sistem üzerinden açar. Gruplara materyaldeki kodun basamaklarını adım adım çalıştırarak incelemeleri ve kod çıktısını kontrol etmeleri istenir. Eğitmen materyal öncesi şu açıklamayı yapar.

> *Koda bakacak olursanız klavyeden girilen karakterlerden sırasıyla "Arda" ismini karakter dizisi kullanarak, "Duru" ismini ise "katar" kullanarak ekrana yazdırıyoruz. Buradaki dizi ve katar arasındaki farkı kodun basamaklarını tek tek inceleyerek tahmin etmeye çalışın.*

Eğitmen çiftlere 5 dk. kadar tartışma süresi tanır. Her çiftten tahminlerini isimlerinin yazılı olduğu kâğıda yazmaları istenir. Tahminler yazıldıktan sonra, öğrenciler kâğıtlarını tahtaya yapıştırır. Beyin fırtınası eşliğinde tüm tahminler eğitmen tarafından okunur ve aradaki fark açıklanır.

Eğitmene Öneriler: Eğitmen öğrenci tahminlerini sınıfta tahtaya yapıştırmak yerine, padlet.com kullanarak dijital tartışma panosu da oluşturabilir. Eğitmen aradaki farkı açıklarken aşağıdaki içerikten yararlanabilir.

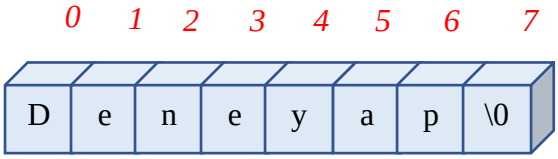
> *Programlamada metin türünde verilerimizi saklamak için kullanılan özel karakter dizileridir. Katarlar, null ('\0') karakter ile sonlandırılmış tek boyutlu karakter dizileri olarak tanımlanabilir. Aşağıda verilen örnekte, "Deneyap" sözcüğünden oluşan bir katar oluşturulmaktadır. Normalde verilen kelime 7 harften oluşsa da sondaki null karakteri tutmak için de bir karakterlik alan gerektiği için bellekte toplamda 8 karakterlik alan ayrılması gerekmektedir.*

```
char katar[] = {'D', 'e', 'n', 'e', 'y', 'a', 'p', '\0'};
```

> *Dizilerdeki ilk değer atama yöntemlerini hatırlarsanız aşağıdaki gibi bir ilk değer ataması yapabiliriz. Dizi değişkeni, çift tırnak işareti içine alınmış bir karakter dizisi içerir.*

```
char katar[] = "Deneyap";
```

Aslında yukarıdaki tanımlamada gördüğünüz üzere null karakteri bir katar sabitinin sonuna yerleştirmezsiniz. C++ derleyicisi diziyi oluşturduğunda '\0' değerini dizinin sonuna otomatik olarak ekler. Yukarıdaki tanımlama sonucu bellekte şu şekilde bir yerleşim söz konusu olacaktır.



C. Kısmi Öğrenme Görevleri

Süre: 50 dk.

Materyal: L6. C. 1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Tasarlayıcı: Okul müdürü, Kimya ve Biyoloji dersini alan 6 öğrencinin not ortalamalarını merak etmektedir. Bunun için Bilişim Teknolojileri uzmanından kendisi için bir program yazmasını ister. Bilişim teknolojileri uzmanı bu programı yazmak için iki boyutlu dizilerden yararlanır. Buna göre hazırlanan programı tasarlayınız.

```
#include <iostream>
using namespace std;
int main()
```

```
{
    int notlar[2][6] = {{85, 73, 92, 95, 80, 78},
                        {69, 76, 87, 65, 90, 50}};
    int top1 = 0, top2 = 0, n = 6;
    float ort1, ort2;
    cout << "Kimya Notlari: " << endl;
    for (int i=0; i < n; i++) {
        cout << notlar[0][i] << " ";
        top1 += notlar[0][i];
    }
    cout << "\nBiyoloji Notlari: " << endl;
    for (int i=0; i < n; i++) {
        cout << notlar[1][i] << " ";
        top2 += notlar[1][i];
    }
    ort1 = (float)top1 / n;
    cout << "\nKimya ortalamasi: " << ort1 << endl;
    ort2 = (float)top2 / n;
    cout << "\nBiyoloji ortalamasi: " << ort2 << endl;
    return 0;
}
```

Analizci: Arkadaşın (bilgisayar) aklından 1-9 arasında rastgele bir sayı tutar. Sen de tutulan bu sayıyı 3 tahminde bulmaya çalışan bir program yazıyorsun. Kural gereği tutulan sayıyı 3 tahminde bulamazsan oyun sona erer. Eğer 3 tahminden birinde sayıyı bulursan program tutulan sayıyı kaçıncı tahmin hakkında bulduğunu ekrana yazdırır.

```
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
    int sayi;
    int tahmin = -1;
    int tahmin_sayisi = 0;
    int tahmin_limiti = 3;
    bool outOfGuesses = false;
```

```
    srand(time(NULL));


    sayi = rand() % 9 + 1;
    cout << sayi;
    while(tahmin != sayi && tahmin_sayisi < tahmin_limiti){
        cout << "Tahmininizi girin: ";
        cin >> tahmin;
        tahmin_sayisi++;
    }
    if(tahmin == sayi){
         cout << "Tebrikler, " << tahmin_sayisi <<". denemede kazandiniz!" << endl;
    } else {
         cout << "Uzgunum, 3 hakkinizda bilemediniz!" << endl;
    }
    return 0;
}
```

Kodlayıcı: Aşağıdaki tabloda voleybol oyuncularının numaraları verilmektedir. Koç, oyuncuları maç öncesi çıktıdaki gibi sıraya dizmek istiyor. Bu sırayı oluşturacak kodu tasarlayınız. Program içerisinde tanımlama bölümünde oyuncuların ilk dizilimi aşağıdaki matristeki gibi olmalıdır.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Oyuncu Sırası:

1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

```
#include <iostream>
using namespace std;
int main()
{
```

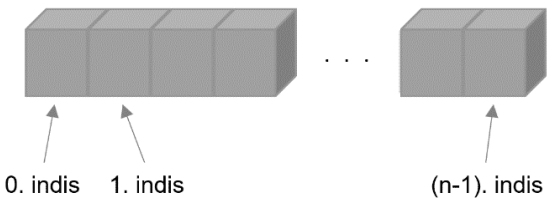
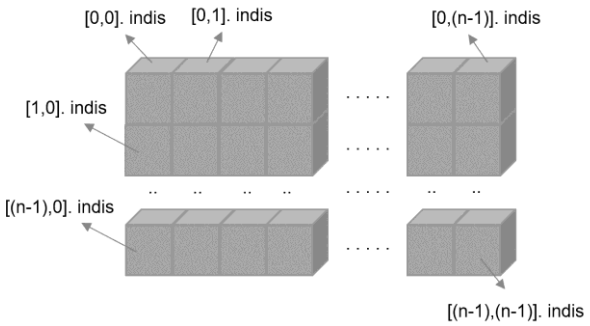
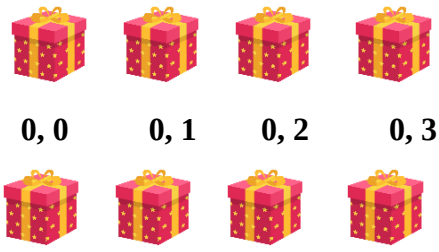
```
    int a[4][4] = {{1, 2, 3, 4},
                   {5, 6, 7, 8},
                   {9, 10, 11, 12},
                   {13, 14, 15, 16}};
    int m = 4, n = 4, i, j = 0, k = 0;
    while (k < m && j < n) {
        for (i = j; i < n; ++i) {
            cout << a[k][i] << " ";
        }
        k++;
        for (i = k; i < m; ++i) {
            cout << a[i][n - 1] << " ";
        }
        n--;
        if (k < m) {
            for (i = n - 1; i >= j; --i) {
                cout << a[m - 1][i] << " ";
            }
            m--;
        }
        if (j < n) {
            for (i = m - 1; i >= k; --i) {
                cout << a[i][j] << " ";
            }
            j++;
        }
    }
    return 0;
}
```

Hafta 6. Ders Materyalleri

L6. B1. 1. Destekleyici Bilgi: Dizileri Tanıyalım!

Diziler: Dizi, tek bir veri parçasını depolayabilen klasik bir değişkenin aksine, birden çok veri öğesini depolayabilen bir veri yapısıdır. Diziler tek boyutlu ve çok boyutlu olmak üzere ikiye ayrılır.

İndis: Bir dizi, bir veri kümesi tutar. Kümenin her üyesine bir eleman denir. İndis, dizinin hangi elemanına eriştiğinizi gösteren bir sayıdır.

Tek Boyutlu Diziler: Tek bir veri türü içeren ve birden fazla değişkeni bir arada tutmaya yarayan veri yapısıdır.	***Çok Boyutlu Diziler:*** Dizilerin elemanları da bir dizi tutabilir. Bu dizileri ifade etmek için dizilerin dizisi ya da dizilerden oluşan diziler ifadesi kullanılır. Aşağıda iki boyutlu dizi örneği verilmektedir.
	
UYARI: Dizilerin indislerinin numaralandırması sıfırdan başlar. Bu nedenle n elemandan oluşan tek boyutlu bir dizideki son elemanın indisi n değil (n-1) olur.	***UYARI***: Ayrıca çok boyutlu bir dizide saklanabilecek toplam eleman sayısı, tüm boyutların çarpımı ile hesaplanabilir. 3*2, 6 elemanlı dizi anlamına gelir.
5 elemanlı tek boyutlu dizi  **0 1 2 3 4** İndisler	2*4'lük 8 elemanlı iki boyutlu dizi  İndisler: **0, 0 0, 1 0, 2 0, 3** **1, 0 1, 1 1, 2 1, 3**

3 elemanlı tek boyutlu hatalı dizi: Dizi aynı veri türünde olmalıdır.	2*3'lik iki boyutlu ancak hatalı dizi: Dizi aynı veri türünde olmalıdır.
	

L6. B2. 1. Destekleyici Bilgi: Dizilere Değer Verelim!

Tek Boyutlu Diziler: Tek bir veri türü içeren ve birden fazla değişkeni bir arada tutmaya yarayan veri yapısıdır.	***Çok Boyutlu Diziler:*** Dizilerin elemanları da bir dizi tutabilir. Bu dizileri ifade etmek için dizilerin dizisi ya da dizilerden oluşan diziler ifadesi kullanılır. Aşağıda iki boyutlu dizi örneği verilmektedir.
veri_tipi dizi_adi[eleman_sayisi]={değer1, değer2,...}; Beş adet öğrenciye ait öğrenci notunu saklamak için oluşturulan "notlar" dizisine değer atamak için: `int notlar[5] = {90, 70, 50, 80, 85};` **Uyarı!** Notlar dizisi 5 elemana sahiptir.	veri_tipi dizi_adi[boyut1][boyut2]…[boyutN]= {değer1, değer2,..değerN}; Dört adet öğrencinin bir dersten aldığı iki yazılı notunu saklamak için oluşturulan "notlar" dizisine değer atamak için: `int notlar[4][2] = {{90, 70},` `{50, 80},` `{85, 86},` `{50, 70}};` **Uyarı!** Notlar dizisi 4*2 = 8 elemana sahiptir.
veri_tipi dizi_adi[] = {değer1, değer2, değer3, …}; Beş adet öğrenci adlarının ilk harflerini saklamak için oluşturulan "harfler" dizisine değer atamak için: `char harfler[] = {'H', 'K', 'A', 'R', 'S'};`	Beş adet öğrenci ad ve soyadlarının ilk harflerini saklamak için oluşturulan "harfler" dizisine değer atamak için: `char harfler[2] [5] = {{'H', 'K', 'A', 'R', 'S'},` `{'M', 'N', 'P', 'S', 'D'} };` **Uyarı!** Dizi 2*5 = 10 elemana sahiptir ve başlangıç değeri ataması gerçekleştirilmiştir. Değer atamada bu yöntem satır ve sütunları gösterdiği için daha çok tercih edilir.

Uyarı! Derleyici, bu şekilde dizinin elemanları verildiği zaman dizi boyutunun ne olacağına kendisi karar verecektir.	
`int notlar[5] = {90, 70, 50};` `int notlar[5] = {90, 70, 50, 0, 0};` **Uyarı!** Dizi ilk değer atamasında eleman sayısı 5'tir. Ancak diziye sadece üç değer girilmiş, yani eleman sayısı kadar değer girilmemiştir. Bu durumda 5 elemanlı dizinin 4. ve 5. elemanlarının değeri 0 olacaktır. Yukarıdaki iki tanımlamada aynıdır.	`int sayilar [4][3][5];` Yukarıda tanımlanan *sayilar* dizisi 4 * 3 * 5 = 60 elemana sahiptir. **Uyarı!** Çok boyutlu bir dizide istediğiniz sayıda boyuta sahip olabilirsiniz. Ancak, çok fazla boyut tanımlamanız bilgisayarın belleğini hızlı bir şekilde doldurabilir. Çok boyutlu dizilerin en basit formu iki boyutlu dizilerdir. İki boyutlu bir dizi dizi elemanlarının da bir dizi olması hâlidir.

L6. B2. 2. Görev Kartları: Dizilere Değer Verelim!

Tek Boyutlu Diziler	***Çok Boyutlu Diziler***
Sıra Sizde! `int notlar[5] = {90, 70, 50, 80, 85};` Örnekteki notlar dizisine benzer bir başka dizide siz oluşturun ve diziye değer atayın. Yazdığınız dizinin eleman sayısını birlikte tartışın.	**Sıra Sizde!** `int notlar[3][5]={{90, 70, 50, 80, 85},` `{86, 50, 70, 90, 95} };` Örnekteki notlar dizisine benzer bir başka dizide siz oluşturun ve diziye değer atayın. Yazdığınız dizinin eleman sayısını birlikte tartışın.
Sıra Sizde!	**Sıra Sizde!** `char harfler[2] [5] = {{'H', 'K', 'A', 'R', 'S'},`

char harfler[] = {'H', 'K', 'A', 'R', 'S'}; Örnekteki harfler dizisinde 2. indis'ten sonra gelen dizi elemanı hangisidir?	{'M', 'N', 'P', 'S', 'D'} }; Örnekteki harfler dizisinde (1,2) numaralı indis'ten (2.satır 3. sütun olmakta) önce gelen dizi elemanı hangisidir?
Sıra Sizde! **int** notlar[6] = {90, 70, 50}; Yukarıdaki dizi başka hangi şekilde yazılabilirdi?	**Sıra Sizde!** **int** sayilar [2][5][4]; Yukarıdaki dizi kaç elemanlıdır?

L6. B3. 1. Destekleyici Bilgi: Döngülerle Diziler

Problem: Ayşe öğretmen sınıfındaki beş öğrencisinin Yabancı Dil sınav notlarını bir program kullanarak listelemek istemektedir. Bunun için öğrencilerinden biri aşağıdaki kodları tasarlamaktadır.

```
#include <iostream>
using namespace std;
int main()
{
    int notlar[5];
    int i;
    for(i=0; i<5; i++){
        cout << i+1 << ". ogrenci notunu giriniz: ";
        cin >> notlar[i];
    }
    return 0;
}
```

Kodun Çıktısı:

1. ogrenci notunu giriniz:
75
2. ogrenci notunu giriniz: *65*
3. ogrenci notunu giriniz: *90*
4. ogrenci notunu giriniz: *87*
5. ogrenci notunu giriniz: *81*

L6. B3. 2. Görev Kartı: Döngülerle Diziler

Problem: 6 arkadaşının doğduğu yılı ekrana yazdıran program.

```
#include <iostream>
using namespace std;
int main(){
   int d_yili1 = 2005;
   int d_yili2 = 2004;
   int d_yili3 = 2003;
   int d_yili4 = 2008;
   int d_yili5 = 2006;
   int d_yili6 = 2002;

   cout << "1. arkadasimin dogum yili: " << d_yili1 << endl;
   cout << "2. arkadasimin dogum yili: " << d_yili2 << endl;
   cout << "3. arkadasimin dogum yili: " << d_yili3 << endl;
   cout << "4. arkadasimin dogum yili: " << d_yili4 << endl;
   cout << "5. arkadasimin dogum yili: " << d_yili5 << endl;
   cout << "6. arkadasimin dogum yili: " << d_yili6 << endl;

   return 0;
}
```

Kodun Çıktısı:

```
1. arkadasimin dogum yili: 2005
2. arkadasimin dogum yili: 2004
3. arkadasimin dogum yili: 2003
4. arkadasimin dogum yili: 2008
5. arkadasimin dogum yili: 2006
6. arkadasimin dogum yili: 2002
```

Görev dış çember: Yukarıdaki problemin çözümünü, kod çıktısı aynı olacak şekilde tek boyutlu dizi kullanarak yeniden programlayınız.

Görev iç çember: Yukarıdaki problemin çözümünü, kod çıktısı aynı olacak şekilde çok boyutlu dizi kullanarak yeniden programlayınız.

L6. B4. 1. Görev Kodu

Görev Kodu 1

```
#include <iostream>
using namespace std;
int main(){
    int d1[4], d2[4], d3[4];
    int i;
    for(i=0; i<4;i++){
        cout << "1. Dizinin " << (i+1) << ". elemanini giriniz: ";
        cin >> d1[i];
    }
    cout << endl;
    for(i=0; i<4;i++){
        cout << "2. Dizinin " << (i+1) << ". elemanini giriniz: ";
        cin >> d2[i];
    }
    cout << endl;
    for(i=0; i<4;i++){
        d3[i] = d1[i] * d2[i];
        cout << "3. Dizinin " << (i+1) << ". elemani: " << d3[i] << endl;
    }
    return 0;
}
```

Görev Kodu 2

```
#include <iostream>
using namespace std;
int main(){
    float dizi[] = {19.23, 26.43, 14.72, 28.71, 15.04, 10.06, 22.96};
    int i, konum, el_sayisi = 7;
    float minimum;
    cout << "Dizi: ";
    for(i=0; i < el_sayisi; i++)
        cout << dizi[i] << " ";
    minimum = dizi[0];
    konum = 0;
    for(i=1; i < el_sayisi; i++){
        if(minimum > dizi[i]){
            minimum = dizi[i];
            konum = i;
        }
    }
    cout << "\nDizinin en kucuk elemani "<< konum <<". indisteki "<<
minimum;
  return 0;
}
```

Görev Kodu 3

```
#include <iostream>
using namespace std;
int main()
{
    int sayilar[] = {5, 3, 2, 5, 1, 2, 6, 9, 8, 1};
    int i, j, boyut = 10;
    cout << "Dizi: ";
    for(i = 0; i < boyut; i++)
        cout << sayilar[i] << " ";

    cout  << "\nTekrar eden elemanlar: ";
    for(i = 0; i < boyut; i++)
        for(j = i+1; j < boyut; j++)
            if(sayilar[i] == sayilar[j])
                cout <<  sayilar[i] << " ";
    return 0;
}
```

Görev Kodu 4

```
#include <iostream>
using namespace std;
int main()
{
    int sayilar[100];
    int i, n, bol3=0, bol5=0;
    cout << "Dizi boyutunu girin : ";
    cin >> n;
    cout<<"\nDizi elemanlarini girin :\n";
    for(i=0; i<n; i++){
        cout << "Elemani girin dizi[" << i << "] : ";
        cin >> sayilar[i];
    }
    for(i=0; i<n; i++){
        if(sayilar[i]%3==0)
            bol3++;
        if(sayilar[i]%5==0)
            bol5++;
    }
    cout << "\n3 ile bolunebilen eleman sayisi: " << bol3;
    cout << "\n5 ile bolunebilen eleman sayisi: " << bol5;
    return 0;
}
```

Görev Kodu 5

```
#include <iostream>
using namespace std;
int main()
{
    int dizi[100];
    int i, boyut, tek=0, cift=0;
    cout << "Dizi boyutunu girin : ";
    cin >> boyut;
    cout<<"\nDizi elemanlarini girin :\n";
    for(i=0; i<boyut; i++){
        cout << "Elemani girin dizi[" << i << "] : ";
        cin >> dizi[i];
    }
    for(i=0; i<boyut; i++){
        if(dizi[i]%2==0)
            cift++;
        else
            tek++;
    }
    cout << "\nCift eleman sayisi: " << cift;
    cout << "\nTek eleman sayisi: " << tek;
    return 0;
}
```

L6. B5. 1. Problem

Problem

Haftalık oyun oynayan 5 arkadaşın ilk hafta oyunu sonunda aldıkları puanlar aşağıdaki tabloda verilmiştir. Ancak ikinci hafta skor tablosu farklı bir versiyonla ekranda gösteriliyor. Buna göre ilk haftanın skor tablosunu, ekranda ikinci haftadaki gibi görüntüleyecek programı tasarlayınız.

Hafta 1

Skor Tablosu:

Oyuncu	Skor				
A	0	0	1	0	0
B	0	0	0	1	1
C	0	2	0	0	1
D	2	0	2	0	0
E	0	0	2	0	0

Hafta 2

Skor Tablosu:

	Oyuncu				
	A	B	C	D	E
Skor	0	0	0	2	0
	0	0	2	0	0
	1	0	0	2	2
	0	1	0	0	0
	0	1	1	0	0

L7. B5. 2. İç Çember-Dış Çember Görev Kartı

İç Çember Görev Kartı

Problemde yer alan ilk hafta skorlarını sırayla ekrana yazdıran kod satırlarını tamamlayınız.

Dış Çember Görev Kartı

İlk hafta skorlarını, ikinci hafta skorlarını oluşturacak şekilde yer değiştirerek ekrana yazdıran kod satırlarını yazınız.

Kodlama Ekibi Görev Kartı

Şimdi elinizde temel problemi çözecek kod satırı parçaları vardır. Bunları bir araya getirerek temel problemin çıktısını oluşturacak programı yazıp, çalıştırın.

L6. B6. 1. Farkı Bul!

```
#include<iostream>
using namespace std;
int main()
{
    char dizi[4];
    char katar[5];
    int i;

    cout << "Ilk ismin karakterlerini giriniz: " << endl;
    for(i=0; i < 4; i++) {
        cin >> dizi[i];
    }
    cout << "Ikinci ismin karakterlerini giriniz: " << endl;
    for(i=0; i < 4; i++) {
        cin >> katar[i];
    }
    katar[4] = '\0';

    cout << "Ilk isim: ";
    for(i=0; i < 4; i++) {
        cout << dizi[i];
    }

    cout << "\nIkinci isim: ";
    cout << katar;

    return 0;
}
```

L6. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 7. Fonksiyonlar

Kazanımlar

- K1. Verilen problemin çözümü için fonksiyon tanımlar.
- K2. Verilen problemin çözümü için fonksiyon içeren programlar geliştirir.
- K3. Verilen problemin çözümünde fonksiyon çağırma kodunu geliştirir.

Amaç

Bu haftanın amacı öğrencilerin programlamada fonksiyon oluşturmalarını, fonksiyon kullanabilmelerini sağlamaktır.

Önerilen Ders Akışı

A. Giriş: Taş, Kâğıt, Makas (10 dk.)
B. Öğrenme Görevleri
   B1. Fonksiyonları Tanıyalım (40 dk.)
      ❑ Ders Arası (10 dk.)
   B2. Fonksiyonların Nasıl Kullanıldığını Keşfediyorum (40 dk.)
      ❑ Ders Arası (10 dk.)
   B3. Fonksiyonları Kullanarak Kodlama Yapalım (60 dk.)
      ❑ Ders Arası (10 dk.)
C. Kısmi Öğrenme Görevleri (60 dk.)

A. Giriş: Taş, Kâğıt, Makas

Süre: 10 dk.

Uygulama: Sınıfta bulunan her öğrenci kendine bir oyun arkadaşı seçer. Herkes seçtiği oyun arkadaşı ile taş, kâğıt ve makas oyununu karşılıklı oynar, oyunu kaybeden kazandığı kişinin arkasına geçerek onun takipçisi olur. Kazananlar sürekli bir diğer kazananla karşılaşarak aynı şekilde taş, kâğıt ve makas oyununu tekrar eder, kaybeden her kişi kazananın arkasına geçmektedir ve kazananın takipçisi olmaktadır. En uzun kuyruğa ulaşan bir başka ifadeyle hiç kaybetmeyen kişi oyunu kazanır.

B. Öğrenme Görevleri

B1. Fonksiyonları Tanıyalım

Süre: 40 dk.

Kazanımlar: K1. Verilen problemin çözümü için fonksiyon tanımlar.

Materyaller: L7. B1.1. Fonksiyonların Özelliklerini Keşfediyorum!

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan materyalin beş tane olacak şekilde çıktısını alır, her bir kutucuğu keserek derse hazırlar.

Uygulama: Eğitmen öncelikle her bir grup dörderli olacak şekilde sınıfı beş gruba böler. Her bir görev kâğıdı kutucuklarından kesilerek her bir gruba aynı materyal 5 parça olacak şekilde dağıtılır. Daha sonra eğitmen sınıftaki gruplara görevleri şu şekilde verir:

Şimdi her grubun elinde fonksiyonların ne işe yaradıklarını ve özelliklerini anlatan; günlük yaşamla ile ilişkisi kurularak keşfedebileceğiniz örnekler var. Şimdi her grubun her bir örnekten yola çıkarak fonksiyonların ne gibi özellikleri olabileceğine yönelik grupça bir şeyler yazmasını istiyorum. Daha sonra gruplardan gelen fonksiyonlarla ilgili her bir özelliği beyin fırtınası yoluyla tahta da özetleyeceğiz. Görevleriniz başladı ve süreniz 10 dk.

Eğitmen etkinlik sonunda fonksiyonlarla ilgili özellikleri öğrencilerle birlikte aşağıdaki gibi özetlemeye çalışır.

Bir işi alt parçalara bölmek hem işin takibini kolaylaştırır hem de aynı işin iki kez yapılmasını engeller. Bilgisayar programlamada da bu böyledir. Birden fazla kez kullanılacak işler bir çatı altında fonksiyon yazılarak toplanır. Bu sayede;

1. Program takibi kolaylaşır.
2. Hata çözümü kolaylaşır.
3. Tek noktadan değişiklik yapılır.

Fonksiyonlar işleri bölerek daha az satır kod yazmamızı sağlar. Örneğin; yazdığımız programın 10 farklı noktasında 5 satırlık işlem yaptırmamız gerekiyor. Eğer bunun için fonksiyon

kullanmazsak 50 satır kod oluşacaktır. Fonksiyon kullanarak satır sayısı 15'e düşecektir. 5 satır fonksiyon için 10 satır da fonksiyon çağırma için kullanılacaktır.

Eğitmene Öneriler: Yukarıdaki etkinlik bitimiyle beraber öğrencilerden gelen yanıtlar neticesinde fonksiyonların özellikleri aşağıdaki gibi çıkarılmaya çalışılır. Biz yazılımcılar programlarımızda fonksiyonları kullanarak;

1) Daha kolay hatalarımızı bulabiliriz. (Materyaldeki dördüncü örnek.)
2) Daha doğru çözüm üretebiliriz. (Materyaldeki beşinci örnek.)
3) İhtiyaç duyduğumuz anda belli bir görevi yapması için çağırırız. (Materyaldeki birinci örnek.)
4) İhtiyaç duyduğumuz anda belli bir görevi yapması için çağırırız ve kullanılmasını sağlarız (Materyaldeki üçüncü örnek.)
5) Daha az satır kod yazarız ve programın yönetimi kolaylaştırırız. (Materyaldeki ikinci ve beşinci örnek)
6) İstenilen yerlerde kullanılıyorlar ve kod tekrarını önlüyorlar. (Materyaldeki ikinci ve beşinci örnek)

B2. Fonksiyonların Nasıl Kullanıldığını Keşfediyorum

Süre: 40 dk.

Kazanımlar: K2. Verilen problemin çözümü için fonksiyon içeren programlar geliştirir.

Materyaller: L7. B2.1 C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri

L7. B2.2 Fonksiyonların Kullanım Afişi

Hazırlık: Eğitmen birinci materyalin çıktısını alır. Fonksiyonların kullanımına yönelik hazırlanan afiş her öğrencinin görebileceği şekilde ÖYS üzerinden erişime açılır.

Uygulama: Öğrenciler dörderli gruplar hâlinde çalışır. Eğitmen her grupta iki öğrenciye bir çıktı olacak şekilde "L7. B2. 1 C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri" adlı materyalin çıktısını alır. Öğrencilerden verilen iki göreve yönelik fonksiyon tanımlamaları istenir. Eğitmen destekleyici bilgi olarak döngüler için hazırlanan afişi (L7. B2. 2) öğrencilerle ÖYS üzerinden paylaşılır. Öğrencilerin ihtiyaç duyduğu aşamada eğitmenler konu ile ilgili yöntemsel bilgiyi öğrenciye verir.

Eğitmene Öneriler: Yukarıdaki etkinlikler bitirildikten sonra fonksiyonlar aşağıdaki gibi özetlenir.

C++ programlama dilinde fonksiyon yazmak için üç kısım vardır. Bunlardan ilki geriye döndürülecek değişkenin tipi (dönüş tipi), ikinci olarak fonksiyonun ismi, son olarak fonksiyon içerisinde ihtiyaç duyulan bilgilere parametre denir.

Instagram'daki her resmin bir numarası vardır ve bu numarayı kullanarak işlemler yapılır. Bu numarayı bir fonksiyona parametre olarak göndeririz. Bir fotoğrafa yorum göndermek için fonksiyon yazarsak; fonksiyonun ismi: yorum_yap, alacağı değer: yorum metni, geriye de işlemin başarılı olup olmadığı döndürülür.

dönüş_tipi fonksiyon_ismi(parametreler)

{

Yapılacak işlemler

}

Fonksiyonları amacını belirtecek şekilde isimlendirmeye özen gösterilmelidir. Çünkü başka biri fonksiyonu kullanmak istediğinde amacını kolayca anlayabilmelidir. Örneğin verilen sayıların ortalamasını alan bir fonksiyon yazıyorsak ismini ortalama_al veya OrtalamaAl şeklinde belirtebiliriz. Benzer şekilde kullanıcılara mail atacak bir fonksiyon için mail_at veya MailAt şeklinde isimlendirebiliriz.

B3. Fonksiyonları Kullanarak Kodlama Yapalım!

Süre: 60 dk.

Kazanımlar: K1. Verilen problemin çözümü için fonksiyon tanımlar.

K2. Verilen problemin çözümü için fonksiyon içeren programlar geliştirir.

K3. Verilen problemin çözümünde fonksiyon çağırma kodunu geliştirir.

Materyaller: L7.B3.1. Fonksiyonları Kullanarak Kodlama Yapma!

L7.B3.2. Fonksiyon Kodlama ile İlgili Destekleyici Bilgi Sunusu

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan materyallerin her ikisini ÖYS üzerinden erişime açar. Code Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Eğitmenler "Fonksiyonlar kullanarak kodlama yapma" etkinliğindeki görevlerden yedi tanesini de kodlamasını sağlamalıdır. Her görev fonksiyonların farklı özelliklerini yansıtmaktadır. Bu bağlamda öğrencilerin her bir görevi kodlaması sağlanarak öğrencilerin fonksiyonların her bir özelliğini öğrenmesi beklenir. Öğrenciler kodlama esnasında verilen görevler için hazırlanan sunu destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmenlerden biri sınıfı dolaşarak, biri de kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Ekrana 10 kez deneyap ardından 2 kez "Merhaba!" yazan bir ekrana_yaz isimli bir fonksiyon yazalım.

CEVAP 1:

```
void ekrana_yaz()
{
```

```
    for(int i=0; i<10;i++)
    {
        cout << "Deneyap" <<endl;
    }
    for(int i=0; i<2;i++)
    {
        cout << "Merhaba!" <<endl;
    }
}
```

Fonksiyonumuzdan geriye herhangi bir bilgi dönmeyeceği için "void" yani "boş" olarak belirtiyoruz. Eğer bir geri dönüş tipi belirtirsek (int, double vs.), kesinlikle geriye o türde bir değer döndürmemiz gerekiyor. Fonksiyon içerisinde ilk olarak 10 kez "Deneyap" yazdırıyoruz. Ardından da 2 kez "Merhaba!" yazdırıyoruz.

Şimdi bunu sadece fonksiyonun ismini yazarak ana programdan çağıralım:

```
int main()
{
    ekrana_yaz();
}
```

Görev 2: Dikdörtgen şeklinde olan büyük bir arazi üçgensel bölgelere ayrılmak istenmektedir. Bunun içinde araziye ne kadar üçgen sığabileceğini bulmak isteyen bir yazılımcı ihtiyaç duyduğu anda çağırabileceği üçgen alanının hesaplamasına yönelik bir fonksiyon yazmak istemektedir. Yazılımcı bu üçgen alan bulma fonksiyonunu nasıl kodlaması gerekmektedir?

Cevap 2:

```
void ucgen_alan_hesapla(double taban, double yukseklik)
{
    double alan = (taban*yukseklik) / 2;
    cout << alan << endl;
}
```

Fonksiyonumuz hazır hâle geldi. Artık kullanıma hazır, üçgen alanı hesaplayıp ekrana yazdıran bir fonksiyona sahibiz. Programın istediğimiz yerinde çağırıp kullanabiliriz. Tabanı 2 ve yüksekliği 4 olan bir üçgen için alan aşağıdaki gibi hesaplanır. parametreleri doğrudan sayı olarak girebildiğimiz gibi değişken ismi de girebiliriz.

```
int main()
```

```
{
    ucgen_alan_hesapla(2,4);
}
```

Görev 3: Fonksiyona gönderilen tam sayı tipindeki dizinin en büyük sayısını ekrana yazan fonksiyonu yazalım.

Cevap 3:

```
void en_buyuk(int dizi[5])
{
    int enbuyuk = dizi[0];
    for(int i=1;i<5;i++)
    {
        if(enbuyuk<dizi[i])
            enbuyuk=dizi[i];
    }
    cout << enbuyuk;
}
```

Ana programımız ise aşağıdaki şekilde olacaktır.

```
int main()
{
    int sayilar[] = {5,3,4,5,8};
    en_buyuk(sayilar);
}
```

Görev 4: İki dizi içerisindeki en büyük iki sayının toplamını bulan fonksiyonu yazalım.

Cevap 4:

Önceki örneği sadece 1 düzenleme ile kullanabiliriz. Ekrana yazdırmak yerine en büyük sayıyı ana programa göndermemiz gerekiyor. Böylece sonraki işlemlerde kullanabiliriz.

```
int en_buyuk(int dizi[5])
{
    int enbuyuk = dizi[0];
    for(int i=1;i<5;i++)
    {
        if(enbuyuk<dizi[i])
```

```
            enbuyuk=dizi[i];
    }
    return enbuyuk;
}
```

Ana programı da aşağıdaki gibi yazabiliriz. İlk olarak dizilerimizi tanımlıyoruz. Ardından dizi1'i fonksiyona gönderip en büyüğünü buluyoruz. Sonra dizi2'nin en büyük elemanını bulup ekrana yazdırıyoruz. Son olarak yapmak istediğimiz toplama işlemini gerçekleştiriyoruz.

```
int main()
{
    int dizi1[] = {5,3,4,5,8};
    int dizi2[] = {9,3,4,5,8};

    int en_buyuk_1 = en_buyuk(dizi1);
    cout << "1. dizinin en buyugu:" << en_buyuk_1 << endl;
    int en_buyuk_2 = en_buyuk(dizi2);
    cout << "2. dizinin en buyugu:" << en_buyuk_2 << endl;
    cout << en_buyuk_1 << "+" << en_buyuk_2 << "=" << en_buyuk_1 + en_buyuk_2;
}
```

Görev 5: Bir bilgisayar programında, iki adet fonksiyon bulunmaktadır. İlk fonksiyonda, yaş bilgisi alınmakta ve fonksiyon içerisinde güncellenmektedir. Bu güncellenen değer ana program bloğunu etkilememektedir. İkinci fonksiyonda alınan yaş bilgisi fonksiyon içerisinde güncellenmekte ve değişiklik ana programda etkili olmaktadır. Bunun için nasıl bir program yazmalıyız?

Cevap 5:

Referans gönderme:

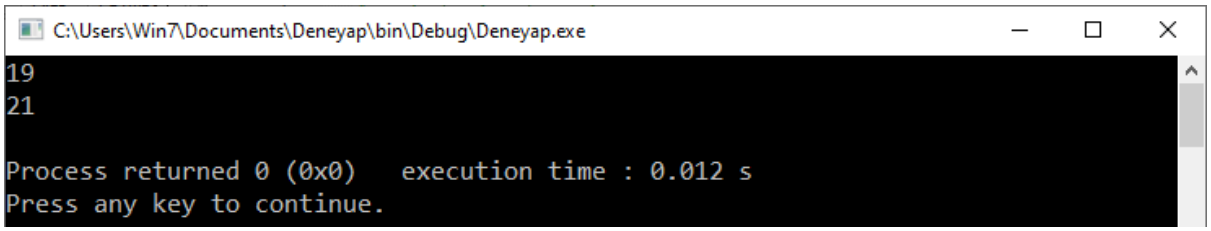
Eğer fonksiyon içerisinde parametre olarak gönderilen değişkenin değeri değişecek ise, değişkenin değeri yerine adresini göndeririz. Böylece fonksiyon içerisindeki değişiklikler değişken üzerine yansıyacaktır. Aşağıdaki örnek bu konuyu açıklayacaktır.

Örnek: Değişken değeri değiştirme referanslı ve referanssız.

```
void fonksiyon1(int sayi)
{
```

```
    sayi = 20;
}
void fonksiyon2(int& sayi)
{
    sayi = 21;
}
int main()
{
   int yas = 19;
   fonksiyon1(yas);
   cout<<yas <<endl;

   fonksiyon2(yas);
   cout<<yas <<endl;
}
```

fonksiyon1'e yas isimli değişkenin değerini yani 19 değerini gönderiyoruz. fonksiyon içerisinde sayi isimli değişken üzerinde yapılan değişiklik ana programdaki "yas" isimli değişkeni etkilemeyecektir. Fakat fonksiyon2'de fonksiyona "&" işareti kullanarak değişkenin adresini gönderiyoruz. Dolayısıyla fonksiyon içerisindeki tüm değişiklikler ana programa yansıyacaktır. Sonuç olarak program çıktısı aşağıdaki gibi olacaktır. Bu sayede birden fazla değişkenin değerini fonksiyon içerisinde değiştirebiliriz.



Resim 25. *Ekran çıktısı*

Görev 6: Harita mühendisi olan Ali kendisine gönderilen arazinin kenarları bir tane gönderildi ise karenin, iki tane gönderildi ise dikdörtgenin çevresini bulduracak bir fonksiyon tanımlamak istemektedir. Sizce Ali nasıl bir kod yazmalıdır?

Aşırı Yükleme:

Bir fonksiyonu birden farklı şekilde kullanabilmek için fonksiyonları farklı şekilde tanımlayabiliriz. Örneğin: bir fonksiyon ile dikdörtgen/kare çevresini hesaplayalım. Eğer kullanıcı fonksiyona iki değişken gönderir ise bu bir dikdörtgen, tek değişken gönderirse kare olarak ele alır ve işlemleri buna göre yaparız.

Cevap 6:

```
#include <iostream>
using namespace std;
int cevre_hesapla(int a)
{
    int cevre = a*4;
    return cevre;
}
int cevre_hesapla(int a, int b)
{
    int cevre = 2*a + 2*b;
    return cevre;
}
int main()
{
   int cevre1 = cevre_hesapla(5);
   int cevre2 = cevre_hesapla(5,4);

   cout << cevre1 <<endl;
   cout << cevre2 <<endl;
}
```

Görev 7: Matematik Öğretmeni Hasan, öğrencilerine gösterebilmek adına ekrana girilen sayının faktöriyelini bulduran bir program oluşturmak istemektedir. Bunun için nasıl bir kod yazmalıdır?

Cevap 7:

```
#include <iostream>
using namespace std;
int faktoriyel(int sayi)
{
    if(sayi == 1)
        return 1;
    else
        return sayi*faktoriyel(sayi-1);
}
int main()
{
```

```
    int sonuc = faktoriyel(5);
    cout << sonuc <<endl;
}
```

C. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L7. C.1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

1) **KODLAYICI-** Parametre olarak gönderilen iki sayının büyüğünün küçüğüne bölümünden kalanı geriye döndüren bir fonksiyon yazınız.

```
#include <iostream>
using namespace std;
int  kalan_bul(int sayi1, int sayi2)
{
    if(sayi1 > sayi2)
        return sayi1 % sayi2;
    else
        return sayi2%sayi1;
}
int main()
{
   int sonuc = kalan_bul(9,12);
```

```
    cout << sonuc;
}
```

2) **KODLAYICI**- Parametre olarak gönderilen dizi içerisindeki sıfırdan büyük sayıların toplamını bulup geriye döndüren bir fonksiyon yazınız.

```
#include <iostream>
using namespace std;
int  dizi_topla(int dizi[5])
{
    int toplam = 0;

    for(int i=0;i<5;i++)
        if(dizi[i] > 0)
            toplam = toplam + dizi[i];

    return toplam;
}
int main()
{
   int sayilar[5] = {5,6,9,3,2};
   int sonuc = dizi_topla(sayilar);
   cout << sonuc;
}
```

3) **KODLAYICI**- Parametre olarak gönderilen sayının asal sayı olup olmadığını belirleyen fonksiyon yazınız. (asal ise 1 değilse 0 döndürsün.)

```
#include <iostream>
using namespace std;
int  asal_sayi_mi(int sayi)
{
    for(int i=2;i<sayi;i++)
        if( sayi% i == 0)
            return 0;

    return 1;
```

```
}
int main()
{
   int sonuc = asal_sayi_mi(23);
   cout << sonuc;
}
```

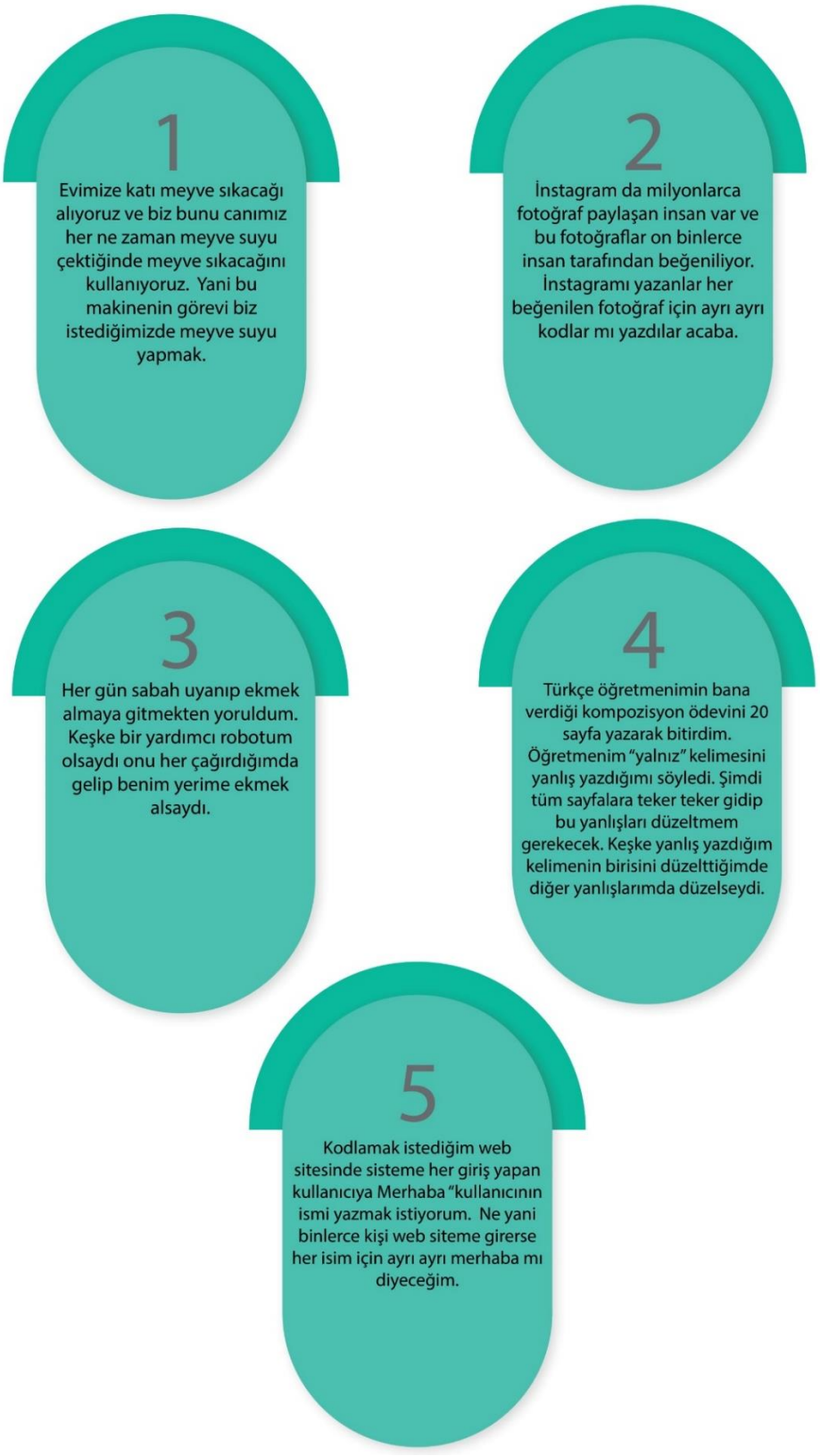
4) **ANALİZCİ**- Aşağıdaki programın ekran çıktısı nedir?

```
#include <iostream>
using namespace std;
int sayi=2;
void fonksiyon1()
{
    sayi = 5;
}
void fonksiyon2()
{
    int sayi = 7;
}
int main()
{
    fonksiyon1();
    fonksiyon2();
    cout << sayi;
}
```

Cevap:5

Hafta 7. Ders Materyalleri

L7. B1. 1 Fonksiyonların Özelliklerini Keşfediyorum!



L7. B2. 1 C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri

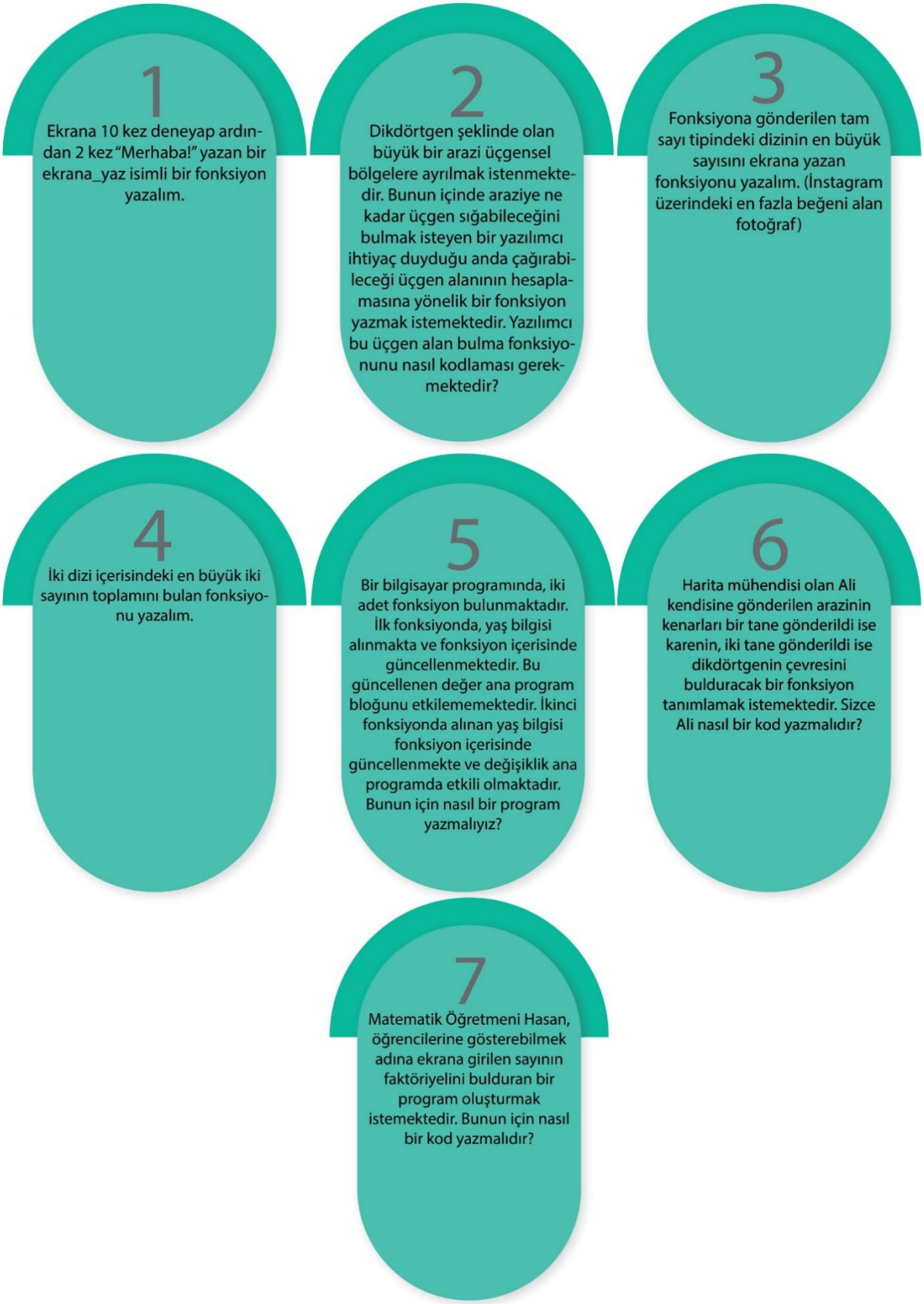
1) Instagram da işe başlayan bir yazılımcı, kullanıcıların fotoğraflara gönderdikleri yorumlara yönelik bir fonksiyon tanımlamak istiyor. Bu fonksiyonu siz olsaydınız nasıl tanımlardınız?

2) Dikdörtgen şeklinde olan büyük bir arazi üçgensel bölgelere ayrılmak istenmektedir. Bunun içinde araziye ne kadar üçgen sığabileceğini bulmak isteyen bir yazılımcı ihtiyaç duyduğu anda çağırabileceği üçgen alanının hesaplanmasına yönelik bir fonksiyon tanımlamak istemektedir. Yazılımcı bu alan fonksiyonunu nasıl tanımlamalıdır?

L7. B2. 2 Fonksiyonların Kullanım Afişi?

Resim 26. *Fonksiyon tanımlama afişi*

L7. B3. 1 Fonksiyonları Kullanarak Kodlama Yapma



L7. B3. 2 Fonksiyon Kodlama ile İlgili Destekleyici Bilgi Sunusu

Yukarıda verilen görevlerin her biri için destekleyici bilgi hazırlanmıştır. Bu destekleyici bilgilere ulaşmak için L7. B3. 2 adlı sunumu açınız. Sunumun çıktısı alınarak her öğrenciye bu çıktılar dağıtılabilir veya öğrencinin kendi bilgisayarına yüklenmesi sağlanabilir.

L7. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 8. Nesneler

Kazanımlar

- K1. C++ programlama dilinde nesne yönelimli programlama mantığını açıklar.
- K2. Nesne yönelimli programlamayı prosedürel programlamadan ayırt eder.
- K3. Nesne yönelimli programlamanın avantaj ve dezavantajlarını ayırt eder.
- K4. Nesne ve sınıf kavramlarını analiz eder.
- K5. Nesne yönelimli programlamada erişim belirteçlerini analiz eder.
- K6. Günlük hayatta karşılaştığı problemlerle ilgili fonksiyon oluşturma işlemini gerçekleştirir.
- K7. C++ programlamada sınıf ve nesne tanımlamasını farklı durumlarda uygular.

Amaç

Bu haftanın amacı, nesne yönelimli programlamanın temel felsefesini, prosedürel programlamadan farkını, avantaj ve dezavantajları ile nesne yönelimli programlamada sınıf, nesne ve fonksiyon tanımlama işlemlerini uygulamaktır.

Önerilen Ders Akışı

A. Öğrenme Görevleri
   A1. Aynısını Çiz (25 dk.)
      - ❑ Ders Arası (5 dk.)

   A2. Sınıfının Özelliklerini Tanı! (60 dk.)
      - ❑ Ders Arası (5 dk.)
      - ❑ Motivasyon Oyunu (10 dk.)

   A3. Afişini Yeniden Tasarla! (60 dk.)
      - ❑ Ders Arası (5 dk.)
      - ❑ Motivasyon Oyunu (10 dk.)

B. Kısmi Öğrenme Görevleri (60 dk.)

A. Öğrenme Görevleri

A1. Aynısını Çiz!

Süre: 25 dk.

Kazanımlar: K1. C++ programlama dilinde nesne yönelimli programlama mantığını açıklar.

K2. Nesne yönelimli programlamayı prosedürel programlamadan ayırt eder.

K3. Nesne yönelimli programlamanın avantaj ve dezavantajlarını ayırt eder.

Materyaller: Bir fanus ya da kura torbası

Hazırlık: Öğrenciler dörderli gruplar hâlinde otururlar. İçinde sınıf örneklerinin olduğu kura torbası hazırlanır.

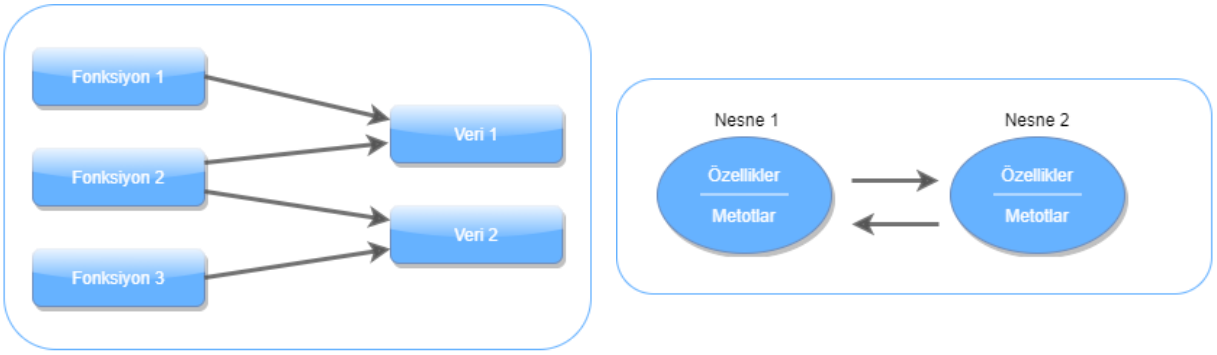
Uygulama: Eğitmen öğrencilere bu öğrenme görevinde bir oyun oynayacaklarını söyler. Her grup meyve, çanta, taşıt, okul gibi sınıf örneklerinin yazılı olduğu kura torbasından bir kâğıt seçer. Grup üyeleri, bu sınıf kartına ait birer nesne düşünür ve bu nesnenin çizimini arkadaşlarına göstermeden çizer. Gruptan biri çizimini diğer arkadaşlarına tahmin ettirmeye çalışır. Grup üyeleri, çizimi yapan arkadaşlarına bu nesnenin özelliklerine ilişkin sorular sorar. Bu sorular; "Rengi ne renk?", "Canlı mı cansız mı?", "Eşya mı?", "Yenilebilir mi?" tarzında nesne hakkında çıkarımda bulunabilecekleri türde olmalıdır. Grup üyeleri bu şekilde arkadaşlarının çizdiği nesneyi tahmin etmeye çalışır. Her bir grupta üyelerin çizimlerini tahmin etme süresi 1 dk. olarak belirtilir. Tüm grup üyelerinin çizimlerinin tahmin süresi bittikten sonra, grupların çizim yaptıkları nesneler ve doğru tahmin edilen nesne sayısı tahtaya yazılır. Eğitmen bu şekilde kazanan grubu belirler. Oyun sonunda eğitmen sınıf ve nesne arasındaki farkı özetler. Eğitmen, grupların sınıf ve nesnelerinden örnekler verir. Buradan yola çıkarak nesne yönelimli programlamanın önemi, avantaj ve dezavantajları hakkında bilgi verir.

Konu İçeriği: Nesne yönelimli programlama, gerçek dünya ortamına dayalı modeller oluşturmak için sınıfları ve nesneleri kullanan bir programlama dilidir. Diğer bir ifadeyle, nesne yönelimli programlama, programları nesnelerin sınıfları açısından tanımlayan bir yazılım tasarım ve programlama yöntemidir. Bir nesne yönelimli programlama uygulaması, belirli bir hizmeti veya bilgiyi talep etmek için çağrıldığında mesajları iletecek bir nesneler koleksiyonunu kullanabilir. Nesneler, verileri veri biçiminde iletebilir, alabilir veya bilgileri işleyebilir.

Prosedürel Programlama	Nesne Yönelimli Programlama
Program, fonksiyon adı verilen küçük parçalara bölünmüştür.	Program, nesneler adı verilen küçük parçalara ayrılır.
Yeni veri ve fonksiyon eklemek kolay değildir.	Yeni veri ve fonksiyon eklemek koladır.

Verileri gizlemek için uygun bir yol yoktur, bu nedenle daha az güvenlidir.	Daha güvenli olmak için veri gizleme sağlar.
Fonksiyon veriden daha önemlidir.	Veri, fonksiyondan daha önemlidir.
Gerçek olmayan dünyaya dayanır.	Gerçek dünyaya dayanır.
Örnekler: C, Fortran, Pascal, Basic vb.	Örnekler: C++, Java, Python, C# vb.

Prosedürel programlama, veriler üzerinde işlemler gerçekleştiren fonksiyonlar yazmakla ilgiliyken, nesne yönelimli programlama ise hem verileri (özellikler) hem de fonksiyonları içeren nesneler oluşturmakla ilgilidir. Aşağıdaki şekilde görüldüğü üzere prosedürel programlama dillerinde veri ve fonksiyonlar birbirinden ayrı iken, nesne yönelimli programlama, veri ve fonksiyonları birleştirerek aralarındaki görevleri gerçekleştirmek için düzenli iletişim sağlayan nesneler kümesi şeklindedir.



Resim 27. *Yordamsal ve nesne yönelimli programlama*

Nesne (Object) Kavramı

Nesne yönelimli programlamanın temel birimi olan, bazı özellikleri ve davranışları olan tanımlanabilir bir varlıktır. Yani hem veriler hem de veriler üzerinde çalışan fonksiyonlar nesne olarak adlandırılan bir birim olarak oluşturulur. Örneğin: bisiklet, sandalye, kitap, top ve cep telefonu gibi.

Resim 28. *Nesne örnekleri*

Nesne bir sınıf örneğidir. Sınıf tanımlandığında, bellek ayrılmaz, ancak örnek oluşturulduğunda (yani bir nesne oluşturulduğunda) bellek ayrılır.

Sınıf (Class) Kavramı

Bir sınıf tanımladığınızda, bir nesne için bir taslak tanımlarsınız. Sınıf adının ne anlama geldiği, yani sınıftaki bir nesnenin ne içereceği ve böyle bir nesne üzerinde hangi işlemlerin gerçekleştirilebileceği tanımlanır. Cep telefonu bir sınıf ise bundan üretilen farklı cep telefonu çeşitleri de nesneleri ifade eder.

Iphone Huawei Samsung

Resim 29. *Sınıf ve nesne kavramı*

Nesne Yönelimli Programlamanın Avantaj ve Dezavantajları

Nesne yönelimli programlamayı kullanmamızın bir nedeni, yeni nesneler mevcut olanlardan özellikleri miras alarak oluşturulurken mevcut kodu korumayı ve değiştirmeyi kolaylaştırmasıdır. Bu, geliştirme süresini önemli ölçüde azaltır ve programlamayı çok daha basit hâle getirir. Diğer bir neden, geliştirme kolaylığı ve diğer geliştiricilerin geliştirmeden sonra programı anlama becerisidir. İyi yorumlanmış nesneler ve sınıflar, geliştiriciye programın geliştiricisinin izlemeye çalıştığı süreci söyleyebilir. Ayrıca yeni geliştirici için programa eklemeleri çok daha kolay hâle getirebilir.

Nesne yönelimli programlamanın, prosedürel programlamaya göre birçok avantajı vardır:

- Daha hızlı ve uygulanması daha kolaydır.
- Programlar için net bir yapı sağlar.
- C++ kodunun tekrar edilmemesine yardımcı olur ve kodun bakımını, değiştirilmesini ve hata ayıklamasını kolaylaştırır.
- Daha az kod ve daha kısa geliştirme süresiyle tam yeniden kullanılabilir uygulamalar oluşturmayı mümkün kılar.

Nesne Yönelimli Programlamanın Temelleri

C++ programlamanın temel amacı, benzerleri arasında en güçlü ve en yaygın kullanılan programlama dillerinden biri olan C programlama diline nesne yönelimi eklemektir. Nesne yönelimli programlamanın özü, kod içerisinde belirli özelliklere ve yöntemlere sahip bir nesne oluşturmaktır. C++ modüllerini tasarlarken tüm dünyayı nesneler şeklinde tasarlamaya

çalışıyoruz. Örneğin bir araba, renk, kapı sayısı ve benzeri gibi belirli özelliklere sahip bir nesnedir. Ayrıca hızlanma, frenleme gibi belirli yöntemlere de sahiptir. Nesne yönelimli programlamanın temelini oluşturan altı ana kavram vardır. Bunlardan ilk ikisi yukarıda detaylı olarak açıklanmıştı. Ama tekrar olması açısından hepsini listelersek:

1) Nesne (Object): Nesne yönelimli programlamanın temel birimidir. Veri ve veriler üzerinde çalışan fonksiyonlar, nesne olarak adlandırılan bir birim olarak paketlenmiştir.

2) Sınıf (Class): Nesne yönelimli programlamaya yapı taşıdır. Sınıfın bir örneğini oluşturarak erişilebilen ve kullanılabilen kendi veri üyelerini ve fonksiyonlarını tutan kullanıcı tanımlı bir veri türüdür.

3) Soyutlama (Abstraction): Dış dünyaya yalnızca gerekli bilgilerin sağlanması ve arka plan ayrıntılarının gizlenmesi, yani gerekli bilgilerin programda ayrıntıları verilmeden temsil edilmesidir.

4) Kapsülleme (Encapsulation): Veri ve bilgilerin tek bir birim altında toplanması olarak tanımlanır. Verileri ve onları işleyen fonksiyonları birbirine bağlamak olarak tanımlanır.

5) Miras (Inheritance): Bir sınıf oluşturulurken başka bir sınıftın özellikleri ve karakteristiklerinin türetilerek kullanılmasıdır. Türetilmiş sınıf olarak adlandırılan yeni bir sınıf oluşturulur.

6) Polimorfizm (Polymorphism): Birden fazla form anlamına gelir. Bir üye fonksiyonun onu çağıran nesneye göre farklı davranabilmesidir.

A2. Sınıfının Özelliklerini Tanı!

Süre: 60 dk.

Kazanımlar: K4. Nesne ve sınıf kavramlarını ayırt eder.

K5. Nesne yönelimli programlamada erişim belirteçlerini analiz eder.

K6. Günlük hayatta karşılaştığı problemlerle ilgili fonksiyon oluşturma işlemini gerçekleştirir.

Materyaller: L8. A2. 1. Grup Afişleri

Hazırlık: Materyallerin 2 adet çıktısı alınır ve her gruba ikişer adet aynı afişten dağıtılır.

Uygulama: Öğrenciler beşerli gruplar hâlinde çalışmaktadır. Eğitmen her gruba grup afişlerinden birini verir. Öğrencilerin gruplar hâlinde bu afişleri incelemeleri ve sınıf, üye listesi, erişim belirteci, fonksiyon ve nesne oluşturma konularını aralarında tartışmaları istenir. Gruplara 5 dk. inceleme yaptıktan sonra, grupların afişlerini ve bu terimleri sırayla diğer arkadaşlarına açıklamaları istenir. Bunun için birinci gruptan birer üye diğer gruplara dağılır. Birinci grubun üyeleri kendi afişlerini dahil olduğu yeni gruptakilere 5 dk. süre ile açıklar ve kendi grubuna geri döner. Benzer şekilde ikinci 5 dk. başlatılır. İkinci grubun her bir üyesi diğer gruplara dağılır ve afişlerini açıklar. İkinci grubun üyeleri geri döndükten sonra, üçüncü grup üyeleri dağılır. Bu şekilde beş grubun da sırası tamamlanır. Eğitmen zaman planlaması için bir çan kullanabilir ya da "Grup 1 Dağılın!", "Grup 1 Geri Dönün" gibi emir ifadeleri ile süreci yönetebilir.

Eğitmene Öneriler: Eğitmen grup etkinliklerinin tamamlanmasını ardından kalan dakikalarda öğrencilere sınıf tanımlama, erişim belirteçleri, üye listesi, fonksiyonlar ve nesne tanımlama ile ilgili sorular yöneltir. Bu şekilde eksik ya da hatalı öğrenmeleri düzenlemeye çalışır.

Konu İçeriği: Sınıf Tanımlama

Sınıflar, nesne yönelimli programlamanın en önemli yapı elemanıdır. Bir sınıf, bir nesnenin özelliklerini ve fonksiyonlarını tanımlar. Cep telefonuna ait bir sınıf oluşturmak istenildiğinde bu sınıfa ait özellikleri ve fonksiyonları belirlememiz gerekmektedir. Özellikler sınıfın ayırt edici bilgilerinden oluşan marka, model ve imei no gibi verileri içermektedir. Fonksiyonlar ise sınıfın arama, mesaj gönderme ve internete bağlanma gibi fonksiyonlarını içermektedir. Sınıfın tasarımı tamamen bizim elimizde olup istediğimiz olası tüm yetenekleri sınıfa ekleyebiliriz. Bir sınıfı tanımlamak için aşağıdaki gibi genel bir sözdizimi kullanılmaktadır.

```
class SınıfAdı
{
        üyeListesi
};
```

Burada **class**, sınıf tanımına ait anahtar kelime, **üyeListesi**, sınıf üyelerinin listesi ve **SınıfAdı**, sınıfın adıdır. Sınıf adının büyük harfle başladığına dikkat edelim. Bu en yaygın kullanımdır ve kodunuzun okunabilir olması için önem arz eder. Sınıf bildiriminin fonksiyonlardaki gibi parantezle başlayıp bittiğine dikkat ediniz. Ayrıca kapanış parantezinden sonra noktalı virgül kullanıldığını unutmayınız. Noktalı virgülün unutulması derleyici tarafından yakalanacak bir hataya sebep olacaktır.

UYARI:

Her sınıf ismini büyük harfle başlatın. Bu kural sadece C++ 'da değil, aynı zamanda diğer tüm nesne yönelimli programlama dillerinde de kullanılır.

üyeListesi, üye bildirimlerini içeren listeden oluşur. Bunlar veri üyesi veya fonksiyon bildirimleri olabilir. Veri üyesi bildirimleri normal değişken bildirimleri ile aynıdır. Örneğin,

sırasıyla karakter, tam sayı ve kesirli sayı türlerinde veri üyeleri oluşturmak istersek aşağıdaki tanımlamaları yaparız.

```
int x;
float y;
char z;
```

Ancak, veri üyelerini bildirdiğiniz yerde ilk değer ataması gerçekleştiremezsiniz. Değişkenlere ilk değer atamasının hazırlanan bir fonksiyonda ya da sınıfın dışında gerçekleştirilmesi gerekir. Tanımlanan bu veri üyelerinin kapsamı, sınıftan oluşturulan nesnenin kapsamı ile aynıdır. Bununla birlikte, veri üyelerine sınıfın dışından her zaman erişilebilmesi mümkün değildir. Yukarıda verdiğimiz cep telefonu sınıfında markanın, modelin ve imei numarasının ne olduğu bu veri üyeleri tarafından saklanır. Bir sınıfın veri üyeleri, sınıf özelliklerini tanımlar ve açıklar. Bu nedenle, her bir cep telefonu nesnesinin markası, o nesnenin bir özelliğidir. Aşağıda cep telefonu sınıfına ait marka, model, imei numarası, renk ve fiyat veri üyelerinin sınıf içerisinde nasıl tanımlandığını görebilirsiniz.

```
class CepTelefonu
{
   char marka[30];
   char model[30];
   int imei_no;
   char renk[30];
   float fiyat;
};
```

Erişim Belirteçleri

Erişim belirteçleri, C++ sınıfınızın içerisinde yer alan veri üyelerine nereden erişilebileceğini kontrol etmenizi sağlar. Bu kontrol sayesinde veri üyeleri üzerinde yetkisiz olarak değişiklik yapılmasının önüne geçilmesi sağlanır. Erişim belirteci, sınıftaki veri üyelerine erişimi kontrol eden bir kelimedir. Bir erişim tanımlayıcısının sözdizimi aşağıdaki gibidir:

```
class SınıfAdı
{
        üyeListesi
erişimBelirteci:
        üyeListesi
erişimBelirteci:
        üyeListesi
};
```

Bir erişim belirticisi tanımlandıktan sonraki üye listesindeki tüm veriler için geçerlidir. Sınıf içerisinde başka bir erişim belirticisi tanımlanırsa bu tanımlamadan sonraki veri üyeleri bu erişim türüne ait olur. Sınıfın sonuna ulaşıncaya kadar sınıfın tüm üyelerini en son tanımlanan erişim belirteci etkiler. Bir sınıfın üç farklı erişim belirleyicisi bulunmaktadır:

1) public: Bu anahtar kelime ile tanımlanan tüm üyelere, sınıfın ulaşılabilir olduğu her yerden erişilebilir.

2) private: Bu anahtar kelime ile tanımlanan üyelere yalnızca aynı sınıfın diğer üye fonksiyonları tarafından erişilebilir.

3) protected: Bu anahtar kelime ile tanımlanan üyelere yalnızca aynı sınıfın diğer üye fonksiyonları ve bu sınıftan türetilen sınıfların üye fonksiyonları tarafından erişilebilir.

UYARI:

Varsayılan olarak, tüm sınıf üyelerinin erişimi "private" olarak tanımlıdır. Bu nedenle, sınıf bildiriminde erişim belirticisi olmadan görünen tüm üyelerin erişimi "private" olur.

```
class Ceptelefonu
{
    char marka[30];
    int yil;
public:
    char model[30];
    char renk[30];
private:
    float fiyat;
protected:
    int imei_no;
};
```

Yukarıda verilen sınıf tanımlamasında fiyat veri üyesi ile birlikte marka ve yıl veri üyelerinin de "private" olduğunu unutmayınız. Diğer taraftan model ve renk "public" olarak tanımlı iken imei numarası ise "protected" olarak tanımlanmıştır. Sınıf tanımlamalarınızda istediğiniz kadar erişim belirteciniz olabilir, ancak belirteçleri tek bir grup altında toplamak sınıfın anlaşılırlığını artıracaktır.

UYARI:

Sınıflarınızı projeye eklemeden önce çalıştığından emin olmak için test edin. Bir sınıfın çalıştığını gözlemledikten sonra, sınıflar arasındaki etkileşimin doğru şekilde çalıştığından emin olmanız yeterlidir.

Fonksiyon Oluşturma

Bir sınıfın üye fonksiyonu, diğer herhangi bir değişken gibi sınıf içinde tanıma sahip olan bir fonksiyondur. Üyesi olduğu sınıfın herhangi bir nesnesi üzerinde çalışır ve bu nesne için bir sınıfın tüm üyelerine erişim sağlayabilir. Fonksiyon tanımları olmadan bir sınıfın tanımı tam olarak gerçekleştirilmiş sayılmaz. Fonksiyon tanımı yapıldıktan sonra ancak sınıfın nesneleri kullanılabilir. Tanımlanan bu fonksiyonlara yalnızca sınıfın bir nesnesi aracılığıyla erişilebilir. Bir fonksiyon tanımladığınızda, kapsam çözümleme operatörü "::" aracılığıyla fonksiyon adından ayıracak şekilde sınıf adını da eklemeliyiz. Sınıf adını eklemezsek global fonksiyon tanımı çağrılmaya çalışılır.

UYARI:

Tanımladığınız her bir fonksiyonun yalnızca bir iş yaptığından emin olun. Fonksiyon içerisinde çok fazla kod satırınız var ise, çok fazla şey yapmaya çalışıyorsunuzdur.

Bir fonksiyonu iki şekilde tanımlayabilirsiniz. Birinci yol, sınıf bildirimi içinde bir fonksiyon bildirmek ve sonra onu dışarıda uygulamaktır. İkinci yol, fonksiyonu aynı zamanda sınıf bildirimi içinde bildirmek ve uygulamaktır. Genelde uygulamalarda birinci yol tercih edilir. Bir sınıf dışında gerçekleşen bir fonksiyon tanımının genel sözdizimi şu şekildedir:

```
dönüş_tipi SınıfAdı::fonksiyon_adi(parametre_listesi)
{
        fonksiyon gövdesi
}
```

Gördüğünüz gibi, bu sözdizimi fonksiyon tanımı sözdizimine çok benzemektedir. Tanımlanan fonksiyon içinde, sınıfa ait tüm üyelere adları kullanılarak erişilebilir. Sınıf üyeliği otomatik olarak sağlanır ve aynı sınıfa ait fonksiyonlar birbirini doğrudan çağırabilir. "private" üyelere erişim yalnızca aynı sınıfa ait fonksiyonlar ile mümkündür. Böylece "private" üyeler tamamen sınıf tarafından kontrol edilir. Bir sınıf tanımladığınızda, sınıfa ait veri üyeleri için bellekten otomatik olarak yer ayrılmaz. Bellek ayırmak için bir nesne tanımlanması gerekir. Belirli bir nesne için bir fonksiyon çağrıldığında, ilgili fonksiyon bu nesnenin verilerini işleyebilir. Artık Ceptelefonu sınıfının arama ve mesaj gönderme fonksiyonlarını uygulayabiliriz.

Nesne Oluşturma

Nesne oluşturmak için öncelikle nesne için şablon olacak sınıf tasarımınızı gerçekleştirin. Sınıf tanımlamanızı gerçekleştirdikten sonra nesne değişkeni için bir tanımlayıcı oluşturun ve hangi

sınıftan oluşacağını belirtin. Nesne oluşturmak için aşağıda verilen sözdizimlerinden birini kullanabilirsiniz:

SınıfAdı nesneDeğişkeni;

A3. Afişi Yeniden Tasarla!

Süre: 60 dk.

Kazanımlar: K7. C++ programlamada sınıf ve nesne tanımlamasını farklı durumlarda uygular.

Hazırlık: Öğrenciler dörderli gruplar hâlinde bilgisayar başındadır.

Uygulama: Eğitmen A2 etkinliğinde grupların incelediği afişleri bu etkinlikte de materyal olarak kullanacaklarını belirtir. İlk olarak öğrencilere bir sınıf ve bu sınıfa ait nesneleri düşünmelerini ister. Belirledikleri sınıf ve nesneleri afişteki gibi kodlayarak, kodlarını çalıştırıp test etmeleri istenir. Eğitmen gruplara çalışan kodları, grup afişindekine benzer şekilde yeni bir afişe dönüştürün der. Bu afişte de grup afişindeki gibi kod satırlarını açıklayan bilgiler olması gerektiğini belirtir. Buradaki amacınız afiş aracılığıyla diğer arkadaşlarınıza konuyu öğretmenizdir, artık eğitmen sizsiniz diye de ekler. Eğitmen bu şekilde öğrencilerden doğru çalışan kodlar ile grup afişlerini yeniden tasarımlarını ister.

Eğitmene Öneriler: Eğitmen öğrencilerine yeni afiş tasarlama aşamasında “canva.com” afiş tasarlama programını kullandırabilir. Diğer bir alternatif ise, Word ortamında afişi oluşturmaları beklenebilir.

B. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L8. B. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Denetleyici: Kardeş olan iki peyzaj ustası kare şeklinde olan bahçelerine peyzaj yapmak istiyor. Her ikisi de kendi bahçelerine hem çit gerecek hem de belli bir alana çiçek ekecektir. Çit ve çiçek alışverişi için kare şeklindeki iki bahçenin kenar bilgilerini ölçerek, bahçelerin alan ve çevresini hesaplayan bir program tasarlarlar. Ancak program hata vermektedir. Programı düzeltmek zorundalar çünkü hesaplayacakları daha pek çok bahçe var. Bu iki kardeşe programı baştan tasarlamaları için yardım ediniz.

İPUCU:

İki kardeşin kullandığı hatalı programda sınıf içerisinde kullanılacak kenar bilgileri değer atama yoluyla gerçekleşirken, fonksiyon tanımlama sınıf dışında yazılmıştır. Tasarlayacağınız kodda bu detayların bulunmasına dikkat ediniz.

```
#include <iostream>
using namespace std;
class Kare {
private:
	float kenar;
public:
```

```
    void deger_atama(float);
    float cevre() {
        return 4 * kenar;
    }
    float alan() {
        return kenar * kenar;
    }
};
void Kare::deger_atama (float k) {
    kenar = k;
}
int main () {
    Kare kare1, kare2;
    kare1.deger_atama (6.2);
    kare2.deger_atama (4.3);

    cout<<"Kare 1 Cevresi: "<<kare1.cevre()<<" Alani: "<<kare1.alan()<<"\n";
    cout<<"Kare 2 Cevresi: "<<kare2.cevre()<<" Alani: "<<kare2.alan()<<"\n";
}
```

Kodun Çıktısı:

```
Kare 1 Cevresi: 24.8 Alani: 38.44
Kare 2 Cevresi: 17.2 Alani: 18.49
```

Kodlayıcı: Ev sahipleri bahçelerine biri yetişkin diğeri çocuk için iki havuz yaptırmak istiyor. Bahçeyi inceleyen ustanın, havuzların yapılacağı alanı hesaplamaya ihtiyacı var. Ev sahiplerine istedikleri havuzların yarıçaplarını belirlemelerini istiyor. Ev sahiplerinden bu bilgiyi aldıktan sonra, daire şeklindeki iki havuzun alanını hesaplayan bir programa bilgileri giriyor. Ustanın kullandığı programın kodlarını tasarlayınız.

İPUCU:

Sınıf içerisinde tanımlanacak yarıçap bilgilerinin usta tarafından erişilebilir olmasına dikkat ediniz.

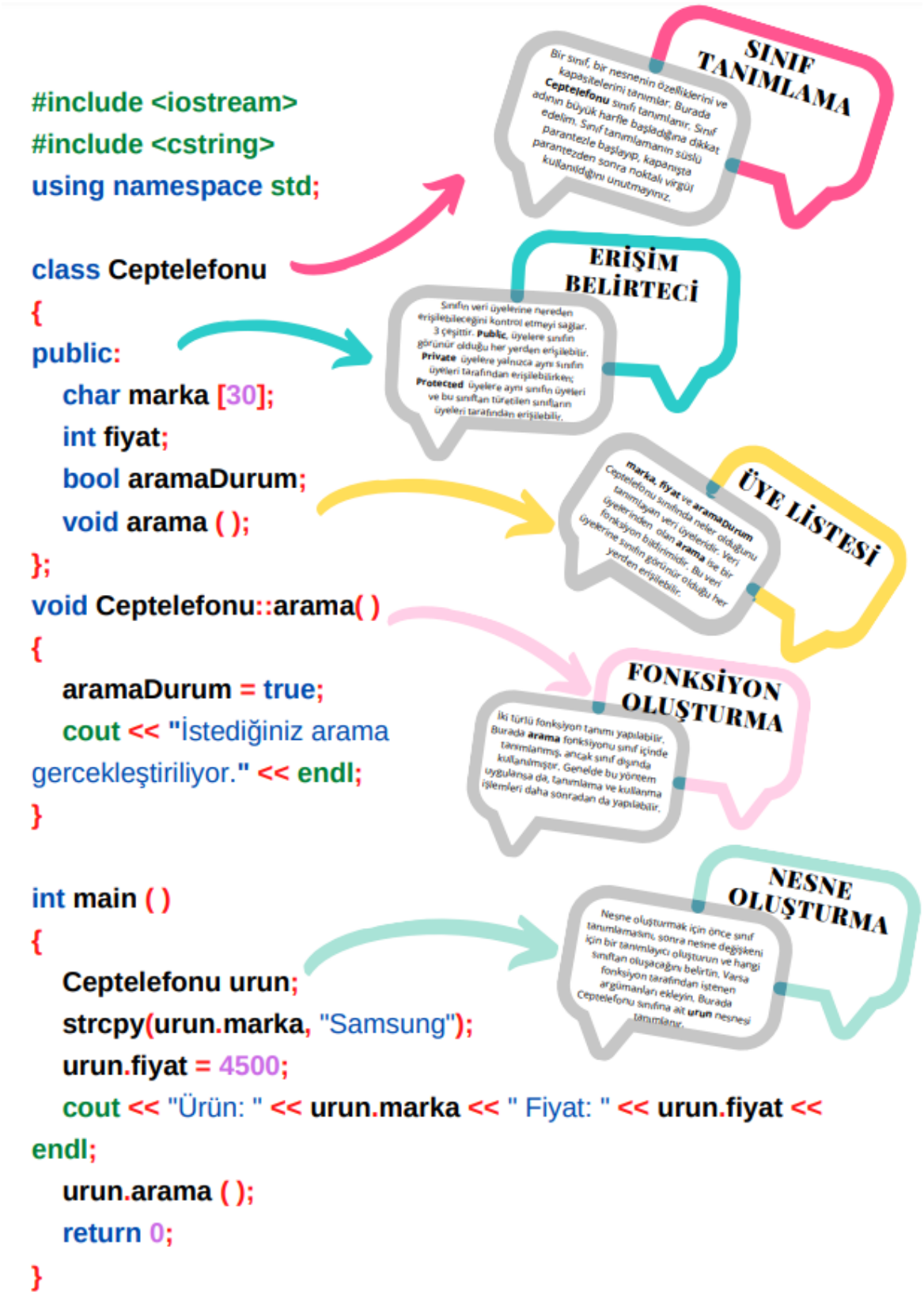
```
#include <iostream>
using namespace std;

class Daire {
```

```
public:
    float yari_cap;
    float alan_bul(float yari_cap) {
        return 3.14 * yari_cap * yari_cap;
    }
};

int main () {
    Daire daire1, daire2;
    cout << "1. dairenin yaricapini giriniz: " << endl;
    cin >> daire1.yari_cap;

    cout << "2. dairenin yaricapini giriniz: " << endl;
    cin >> daire2.yari_cap;

    cout<<"Daire 1 Alani: "<<daire1.alan_bul(daire1.yari_cap)<<"\n";
    cout<<"Daire 2 Alani: "<<daire2.alan_bul(daire2.yari_cap)<<"\n";
}
```

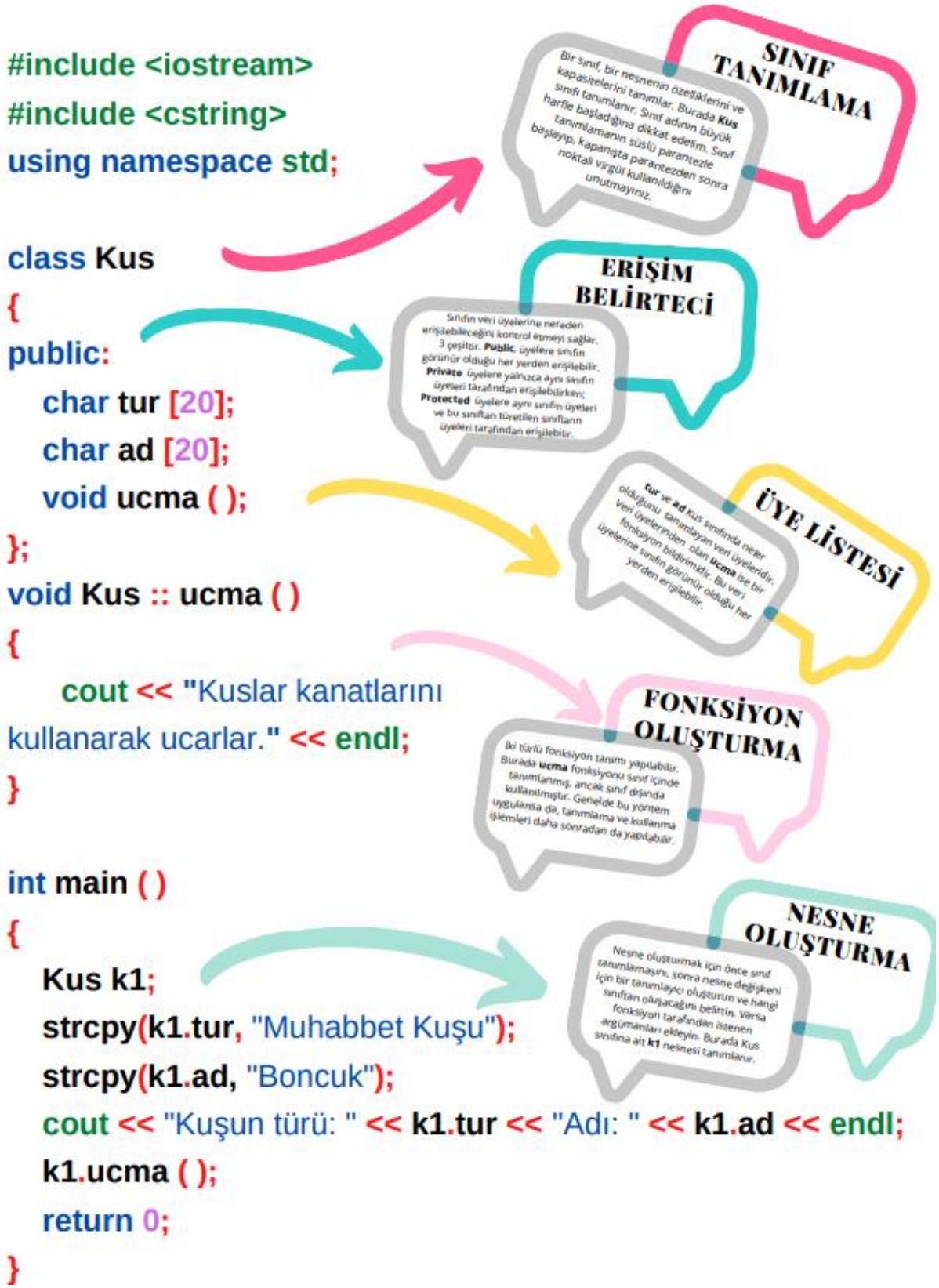
Kodun Çıktısı:

```
1. dairenin yaricapini giriniz:
4.2
2. dairenin yaricapini giriniz:
3.6
Daire 1 Alani: 55.3896
Daire 2 Alani: 40.6944
```

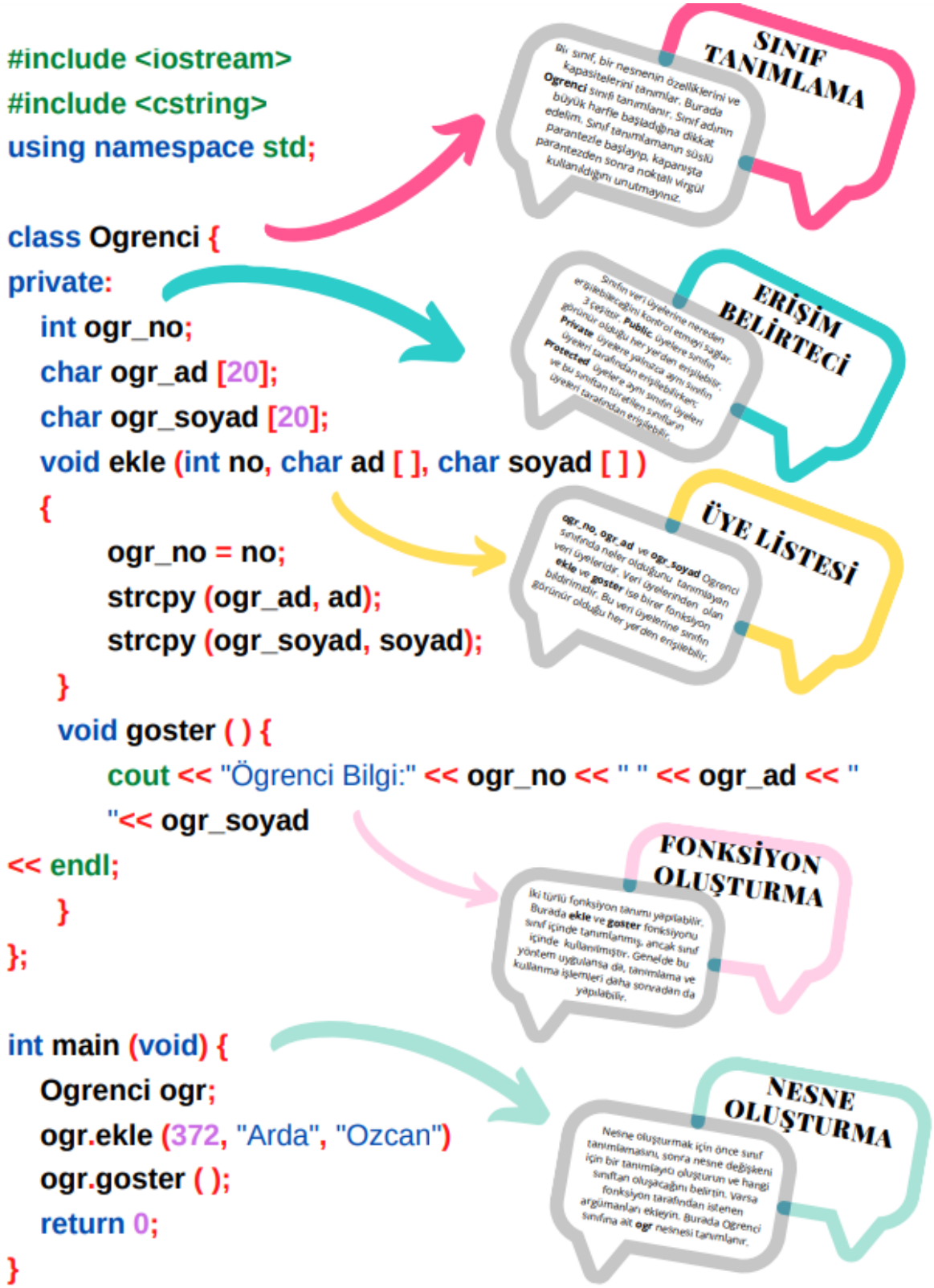
Hafta 8. Ders Materyalleri



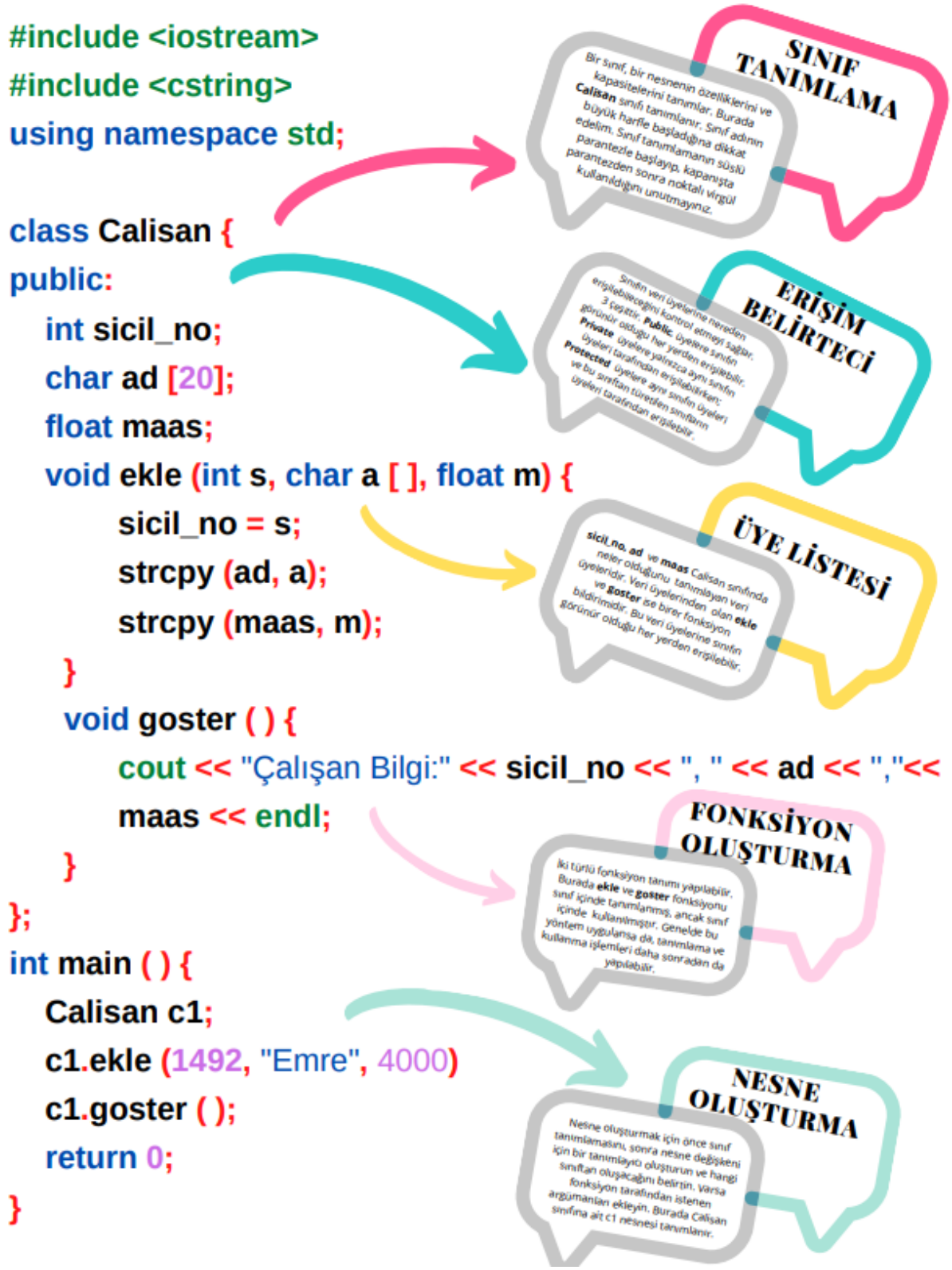
Resim 30. *Cep telefonu sınıfı afişi*



Resim 31. *Kuş sınıfı afişi*



Resim 32. *Öğrenci sınıfı afişi*



Resim 33. *Çalışan sınıfı afişi*

L8. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 9. Nesne Yönelimli Programlama

Kazanımlar

- K1. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.
- K2. Yapıcı fonksiyonları kullanarak program geliştirir.
- K3. Yıkıcı fonksiyonları kullanarak program geliştirir.
- K4. Kod içinde sınıf dosyaları ile proje geliştirir.

Amaç

Bu haftanın amacı yapıcı, yıkıcı ve sabit fonksiyonları kullanmak, sınıf dosyaları ile büyük boyutlu projeler hazırlamayı anlamaktır.

Önerilen Ders Akışı

A. Öğrenme Görevleri
   A1. Yarış Benimle! (40 dk.)
      ❏ Ders Arası (5 dk.)
   A2. Kod Satırlarını Tamamla! (60 dk.)
      ❏ Ders Arası (5 dk.)
      ❏ Motivasyon Oyunu (10 dk.)
   A3. Sınıfları Dosyala! (45 dk.)
      ❏ Ders Arası (5 dk.)
      ❏ Motivasyon Oyunu (10 dk.)
B. Kısmi Öğrenme Görevleri (60 dk.)

A. Öğrenme Görevleri

A1. Yarış Benimle!

Süre: 40 dk.

Kazanımlar: K1. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.

Materyaller: L9. A1. 1. Yarışma Kartları

L9. A1. 2. Kod Örneği

Hazırlık: Bir kura torbasında yarışma kartları bir adet çıktı alınır ve kesikli çizgilerden kesilerek kuraya hazırlanır. Kod örneği ÖYS üzerinden erişimine açılır ve tahtaya yansıtılır.

Uygulama: Eğitmen öğrencilerine bu öğrenme görevinde bir oyun oynayacaklarını söyler. Sınıfı iki gruba ayırır. Her gruba kod örnekleri verilir. Bu örneklerde hem yapıcı fonksiyon hem de yıkıcı fonksiyon kodlarının nasıl kullanıldığı bulunmaktadır. Oyunda gruplara sırayla bir kart çektirilir ve yarışma kartı gruba okunur. Bu kartlarda ise, yapıcı ya da yıkıcı fonksiyon arasındaki farkı anlatan bir özellik yazmaktadır. Grup çektikleri yarışma kartındaki özelliğin, ellerindeki kod örneğine bakarak doğru ya da yanlış bir ifade olup olmadığına karar verecektir. Doğru bilinen her bir karta 10 puan eklenir. Grup puanları sınıfta tahtaya yazılır. Bu şekilde en çok puan toplayan grup, diğer gruptan bir ödül alır. Eğitmen oyun sırasında yapıcı ve yıkıcı fonksiyonları detaylandırır. Oyun sonunda ise kısaca sabit fonksiyonlar hakkında da özetleyici bilgi geçer.

Eğitmene Öneriler: Eğitmen bu oyunu Web 2.0 teknolojilerinden yararlanarak hazırlayabilir. Oyun sonunda grup ödülüne öğrencilerle karar verilebilir ya da basit bir şekilde kazanan grubu alkışlamak gibi takdir hareketleri uygulanabilir. Her bir kartta eğitmen konu içeriğine bağlı olarak açıklamalarda bulunabilir.

Konu İçeriği: Yapıcı ve Yıkıcı Fonksiyonlar

Tanımladığınız sınıf içerisinde yapıcı (constructor) ve yıkıcı (deconstructor) olmak üzere iki özel fonksiyon türü olabilir. Her ikisi de isteğe bağlıdır ve diğer fonksiyonları sağlayamayacağı özel fonksiyonlar sağlarlar. Yapıcı fonksiyonlar sınıftan bir nesne oluşturulduğu zaman otomatik olarak çağrılır ve genellikle veri üyeleri için başlangıç değerlerini ayarlamak için kullanılır. Örneğin cep telefonu sınıfı için düşünecek olursak, yapıcı fonksiyon yıl veri üyesinin değerini sıfır yapabilir. Her zaman sınıfla aynı isme sahiptir ve int, void vb. gibi bir dönüş değerine sahip olamaz.

Yıkıcı fonksiyonlar ise, bir nesne yok edildiğinde otomatik olarak çağrılır ve gerekli tüm temizleme görevlerini gerçekleştirir. Yıkıcı fonksiyonlar, her zaman sınıfla aynı adla adlandırılır, ancak başında yaklaşık işareti (~) bulunur. Yıkıcı fonksiyonlar da bir dönüş değerine sahip olamaz. Hem yapıcı hem de yıkıcı fonksiyonlar diğer fonksiyonlar gibi

tanımlanır ve uygulanır. Sınıf içinde eş zamanlı olarak tanımlamaları yapılacaksa aşağıdaki sözdizimi kullanılır.

```
class SınıfAdı
{
SınıfAdi (parametre listesi){
        yapıcı fonksiyon gövdesi
}
~SınıfAdi (){
        yıkıcı fonksiyon gövdesi
}
};
```

Yapıcı ve yıkıcı fonksiyonların tanımlamaları daha sonra yapılacaksa aşağıdaki sözdizimi kullanılır.

```
class SınıfAdı
{
        SınıfAdi (parametre listesi);
        ~SınıfAdi ();
};

SınıfAdı::SınıfAdi (parametre listesi){
        yapıcı fonksiyon gövdesi
}
SınıfAdı::~SınıfAdi (){
        yıkıcı fonksiyon gövdesi
}
```

Yapıcı metodun parametre alabileceğine dikkat ediniz. Parametre alabilen bir yapıcı fonksiyon oluşturursanız, sınıfın kullanımı sırasında nesne oluştururken bu parametreler için değerler

sağlanmalıdır. Diğer taraftan, yıkıcı fonksiyonların argümanları olamaz. Otomatik olarak çağrıldığından, kullanıcının bağımsız değişken sağlama şansı yoktur.

Sabit Fonksiyonlar

Veri üyelerinin değerinin sabit olduğu fonksiyonlar oluşturmanız mümkündür. Sabit fonksiyonlarda veri üyelerinin değerlerinin değiştirilmesi imkânsızdır. Bunu yapmaya çalışmak sözdizimi hatasına neden olur. Sabit bir fonksiyon bildirmek için, const anahtar sözcüğünü parametre listesinden sonra aşağıdaki sözdiziminde gördüğünüz gibi ekleriz:

```
dönüş_tipi fonksiyon_adi(parametre_listesi) const;
```

Sınıf tanımlamasının dışında sabit bir fonksiyon oluşturmak isterseniz, const anahtar sözcüğünün hem fonksiyon prototipinde hem de tanımına yerleştirilmesi gerekir.

A2. Kod Satırlarını Tamamla!

Süre: 60 dk.

Kazanımlar: K2. Yapıcı fonksiyonları kullanarak program geliştirir.

K3. Yıkıcı fonksiyonları kullanarak program geliştirir.

Materyaller: L9. A2. 1. Grup Görev Kartları

Hazırlık: Materyalin bir adet çıktısı alınır.

Uygulama: Öğrenciler beşerli gruplar hâlinde çalışmaktadır. Bu etkinlikte eğitmen istasyon tekniği kullanmaktadır. Bunun için sınıfta da dört farklı grup masası oluşturulur. Her masada bir görev kartı bulunmaktadır. Gruplar 5 dk. içinde grup görev kartını bilgisayarda kodlayarak tamamlamaya çalışır. Süre bitiminde grupların saat yönünde bir sonraki grup masasına geçmesi istenir. Diğer grubun kaldığı noktadan grup masasındaki görev kartı tamamlanmaya çalışılır. Bu şekilde tüm gruplar her bir grup masasına geçişini tamamladıktan sonra dönme işlemi bitirilir. Dönme işleminin sonunda eğitmen sırayla ilk grubun görevini ve masada oluşan çözümü arkadaşlarına açıklamalarını ister. Varsa hatalı noktalar giderilir. Bu şekilde tüm grup masalarına hak verilir ve tüm kodlardaki görevin amacı eğitmen tarafından açıklanır.

Eğitmene Öneriler: Eğitmen grup yanıtlarını sadece bir arkadaşın aktarması için gruplara bir moderatör belirlemelerini isteyebilir. Bu şekilde bir ya da iki arkadaşın bilgisayar başında diğer arkadaşların da katkısıyla kodu yazmaları hızlandırılmış olur. Grup oluşturma aşamasında bu moderatörlerin seçilmesi sağlanmalıdır. Diğer bir nokta ise, tüm grupların son kod satırlarını padlet.com gibi bir dijital tartışma panosuna göndermeleri istenebilir. Bu şekilde grupların yaptıklarının tüm öğrenciler tarafından erişilebilir olması sağlanır.

Konu İçeriği: Grup görevlerinin amaçlarını ve yanıtlarını aşağıda bulabilirsiniz.

Grup 1: Basit bir Araba sınıfı oluşturarak, birden fazla nesne tanımlaması yapmak.

```
#include <iostream>
#include <cstring>
using namespace std;
// Bazi niteliklere sahip bir Araba sinifi olusturun
class Araba {
public:
    char marka[30];
    char model[30];
    int yil;
    int fiyat;
};
int main() {
    // Ilk Araba nesnesini olusturun
    Araba araba1;
    strcpy(araba1.marka, "Suzuki");
    strcpy(araba1.model, "Vitara");
    araba1.yil = 2016;
    araba1.fiyat = 225000;
    // Ikinci Araba nesnesini olusturun
    Araba araba2;
    strcpy(araba2.marka, "Volkswagen");
    strcpy(araba2.model, "T-Roc");
    araba2.yil = 2020;
    araba2.fiyat = 360000;
    // Nesnelerin ozelliklerini yazdirin
    cout << "Araba 1: " << araba1.marka << ", " << araba1.model << ", " << araba1.yil
<< ", " << araba1.fiyat << " \n";
    cout << "Araba 2: " << araba2.marka << ", " << araba2.model << ", " << araba2.yil
<< ", " << araba2.fiyat << " \n";
    return 0;
}
```

Kodun Çıktısı:

```
Araba 1: Suzuki, Vitara, 2016, 225000
Araba 2: Volkswagen, T-Roc, 2020, 360000
```

Grup 2: Araba sınıfına bir fonksiyon ekleme ve nesne tanımlama.

```
#include <iostream>
#include <cstring>
using namespace std;

class Araba {
public:
    char marka[30];
    char model[30];
    int hiz(int max_hiz);
};

int Araba::hiz(int max_hiz) {
  return max_hiz;
}

int main() {
    Araba araba1; // Araba nesnesini olusturun
    strcpy(araba1.marka, "Opel");
    strcpy(araba1.model, "Astra");
    cout << "Araba: " << araba1.marka << " " << araba1.model << " \n";
    cout << "Hiz: " << araba1.hiz(190); // Metodu parametre ile cagirin
    return 0;
}
```

Kodun Çıktısı:

```
Araba: Opel Astra
Hiz: 190
```

Grup 3: Araba sınıfına sınıf içi bir yapıcı fonksiyon ekleme ve nesne tanımlama.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    //Parametreli sinif ici yapıcı fonksiyon
    Araba(string x_marka, string x_model, int x_yil, int x_fiyat) {
        marka = x_marka;
        model = x_model;
        yil = x_yil;
        fiyat = x_fiyat;
    }
};

int main() {
    // Yapıcı metodu farkli degerlerle cagirarak Araba nesneleri olusturma
    Araba araba1("Suzuki", "Vitara", 2016, 225000);
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);

    // Degerleri yazdirma
    cout << araba1.marka << " " << araba1.model << " " << araba1.yil << " " <<
araba1.fiyat << "\n";
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil << " " <<
araba2.fiyat << "\n";
    return 0;
}
```

Kodun Çıktısı:

```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

Grup 4: Araba sınıfına sınıf dışı bir yapıcı fonksiyon ekleme ve nesne tanımlama.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    // Yapici fonksiyon bildirimi
    Araba(string x_marka, string x_model, int x_yil, int x_fiyat);
};
//Sinif disinda yapıcı fonksiyon tanimlama
Araba::Araba(string x_marka, string x_model, int x_yil, int x_fiyat) {
    marka = x_marka;
    model = x_model;
    yil = x_yil;
    fiyat = x_fiyat;
}
int main() {
    // Yapıcı metodu farkli degerlerle cagirarak Araba nesneleri olusturma
    Araba araba1("Suzuki", "Vitara", 2016, 225000);
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);
    // Degerleri yazdirma
    cout << araba1.marka << " " << araba1.model << " " << araba1.yil << " " <<
araba1.fiyat << "\n";
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil << " " <<
araba2.fiyat << "\n";
    return 0;
}
```

Kodun Çıktısı:

```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

A3. Sınıflarını Dosyala!

Süre: 45 dk.

Kazanımlar: K4. Kod içinde sınıf dosyaları ile proje geliştirir.

Materyaller: L9. A3. 1. Destekleyici Kod Bilgisi

Hazırlık: Öğrenciler ikişerli gruplar hâlinde bilgisayar başındadır.

Uygulama: Eğitmen öğrencilere bir cep telefonu sınıfının, sınıf dosyaları olarak kullanıldığı destekleyici kod bilgisi verir. Eğitmen destekleyici kod üzerinde onlara farklı gelen ya da daha önce görmedikleri kod satırlarının hangileri olduğunu sorar. Burada #include "Ceptelefonu.h" kod satırına odaklanmaları sağlanır. Devamında öğrencilere aşağıdaki görev verilir.

> *Arkadaşlar şimdi bir daire sınıfı tasarlayıp oluşturacağınız iki adet dairenin çevresini ve alanını hesaplayan kod satırlarını yazınız. Bunun için 10 dk süremiz var. Destekleyici kod örneğini referans alabilirsiniz.*

Eğitmen çiftler tarafından verilen görev için yazılan kodları, karşılarındaki çiftlerin yazdığı kod ile değiştirmelerini ister ve aşağıdaki talimatı verir.

Şimdi elinize verilen, arkadaşlarınızın yazdığı kodların çalışıp çalışmadığını test edin. Kodlar çalışmıyor ise çalışır duruma getirip, destekleyici kod bilgisindekine benzer şekilde kodları tablolaştırın. 5 dk. süremiz var.

Etkinlik sonunda eğitmen yeni yazılan kodlar üzerinden sınıf dosyalarına neden ihtiyacımız olduğunu düşünmelerini ve öğrencilerin bunu açıklamasını ister. Öğrencilere söz hakkı verilir ve geri bildirimlerle eğitmen eşliğinde konu tekrar edilir.

Eğitmene Öneriler: Sınıf Dosyaları Oluşturma

Hazırladığınız programlar çok fazla sayıda ve uzun kod satırlarına sahip sınıflar ve fonksiyonlar içerebilir. Böyle durumlarda kodların tek bir dosya üzerinde yönetimi çok zor olur ve belli bir aşamadan sonra yönetilemez hâle gelir. Ayrıca, hazırladığınız sınıflarınızı başka programlarda yeniden kullanmak isteyebilirsiniz. Bu durumda, bunları her bir proje dosyasına tekrar tekrar kopyalayıp eklemeniz gerekir. Bu süreç de oldukça zor olacaktır.

Oluşturmuş olduğunuz sınıfları ayrı bir dosyada tutmanız çok faydalı olacaktır. Bunun için hazırlamış olduğunuz sınıf yapısını SinifAdi.h isimli ve uzantılı bir dosya içerisine kaydedebilirsiniz. Normalde C++ dosyaları için .cpp uzantısı kullanmaktayız, fakat .h uzantılı dosyalar da C++ için geçerli bir dosya uzantısıdır. Fakat biz bu uzantılara sahip dosyalarda sınıflarımızı ve varsa kütüphanelerimizi tanımlayabiliriz. Sınıf tanımlaması, kullanılan kod dosyasından ayrı bir dosyaya kaydedildiği için, #include ifadesi ile programa dahil edilmelidir. #include yönergesinin sözdizimi şu şekildedir:

```
#include "dosya_adı"
```

Destekleyici Kod Bilgisi: Cep telefonu sınıfını sınıf dosyaları kullanarak tanımlama.

ornek.cpp

```
#include <iostream>

#include "Ceptelefonu.h"

using namespace std;


int main()

{

    Ceptelefonu urun1(4500);

    Ceptelefonu urun2 = Ceptelefonu(3750);

    cout << "Urun 1 baslangic fiyati: " <<
urun1.fiyat << endl;

    cout << "Urun 2 baslangic fiyati: " <<
urun2.fiyat << endl;

    return 0;

}
```

Ceptelefonu.h

```
class Ceptelefonu

{

public:

    char marka[30];

    char model[30];

    int fiyat;

    bool aramaDurum;

    Ceptelefonu(int x_fiyat){

        fiyat = x_fiyat;

    }

    ~Ceptelefonu(){

        cout << "Nesne yok edildi." <<
endl;

    }

    void arama();

};

void Ceptelefonu::arama()

{

    aramaDurum = true;

}
```

Kodun Çıktısı:

```
Urun 1 baslangic fiyati: 4500

Urun 2 baslangic fiyati: 3750
```

Görev kodu ve yanıtı: Daire sınıfı kullanarak oluşturacağınız iki adet dairenin çevresini ve alanını hesaplayınız.

Alancevre.cpp | Daire.h

```cpp
#include <iostream>

#include "Daire.h"

using namespace std;


int main () {

    Daire daire1, daire2;

    daire1.deger_atama (4.2,3.14);

    daire2.deger_atama (2.1,3.14);


    cout << "Daire 1 Cevresi: " << daire1.cevre() << "\n";

    cout << "Daire 1 Alani: " << daire1.alan ()<< "\n\n";

    cout << "Daire 2 Cevresi: " << daire2.cevre() << "\n";

    cout << "Daire 2 Alani: " << daire2.alan() << "\n";

    return 0;

}
```

```cpp
class Daire {

    float yaricap;

    float pi_sayisi;

public:

    void deger_atama(float ,float);

    float cevre() {

        return 2 * pi_sayisi * yaricap;

    }

    float alan() {

        return pi_sayisi * yaricap * yaricap;

    }

};


void Daire::deger_atama (float x_yaricap, float x_pi_sayisi) {

    yaricap = x_yaricap;

    pi_sayisi = x_pi_sayisi;

}
```

Kodun Çıktısı:

```
Daire 1 Cevresi: 26.376

Daire 1 Alani: 55.3896


Daire 2 Cevresi: 13.188

Daire 2 Alani: 13.8474
```

B. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L9. B. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Tasarlayıcı: Bir basketbol takımının koçu, oyuncularının ad-soyad, forma numarası ve attığı basket bilgilerini tutmak için bir program hazırlamak ister. Koç bunun için örnek bir program yazar. Programda örnek olarak Basketbolcu sınıf yapısı ve bu sınıfa ait 2 basketbolcu nesnesi bulunmaktadır. Koçun yazdığı örnek kodu tasarlayınız.

```
#include <iostream>
#include <string>
using namespace std;
class Basketbolcu
{
public:
	string ad_soyad;
	int forma_no;
	int basket_sayisi;
	Basketbolcu(string x_ad_soyad, int x_forma_no, int x_basket_sayisi){
	ad_soyad = x_ad_soyad;
	forma_no = x_forma_no;
	basket_sayisi = x_basket_sayisi;
	}
};
int main()
{
	Basketbolcu b1("Mirsad Turkcan", 5 , 21);
```

```
    Basketbolcu b2("Semih Erden", 9, 15);
    cout << b1.ad_soyad << " " << b1.forma_no << " " << b1.basket_sayisi << "\n";
    cout << b2.ad_soyad << " " << b2.forma_no << " " << b2.basket_sayisi << "\n";
    return 0;
}
```

Kodun Çıktısı:

```
Mirsad Turkcan 5 21
Semih Erden 9 15
```

Kodlayıcı: Grafik programlarında kullanmak üzere nokta nesnelerini tanımlamak için bir Nokta sınıfı oluşturalım. Noktalar iki boyutlu düzlemde yer alacağından özellik olarak x ve y koordinatları olmak üzere iki adet koordinat bilgisine sahiptir. Programınızda noktaların sahip olması gereken yetenekler (davranışlar) ise şunlar olmalıdır:

- Noktalar, düzlemde herhangi bir yere konumlanabilmeli: git fonksiyonu
- Noktalar bulundukları koordinatları ekranda gösterebilmeli: goster fonksiyonu
- Noktalar, sıfır (0,0) koordinatında olup olmadıkları sorusunu yanıtlayabilmeli: sifir_mi fonksiyonu

```
#include <iostream>
using namespace std;
class Nokta{
    int x,y;
    public:
    void git(int, int);
    void goster();
    void sifir_mi();
};
void Nokta::git(int yeni_x, int yeni_y)
{
    x = yeni_x;
    y = yeni_y;
}
void Nokta::goster()
{
    cout << "X noktasi: " << x << ", Y noktasi: " << y << endl;
}
```

```
void Nokta::sifir_mi()
{
    if ((x == 0) && (y == 0))
        cout << "n1 su anda sifir noktasindadir." << endl;
    else
        cout << "n1 su anda sifir noktasinda degildir." << endl;
}
int main() {
    Nokta n1,n2;
    n1.git(78,34);
    n1.goster();
    n1.git(61,35);
    n1.goster();
    n1.sifir_mi();
    n2.git(0,0);
    n2.sifir_mi();
    return 0;
}
```

Kodun Çıktısı:

X noktasi: 78, Y noktasi: 34

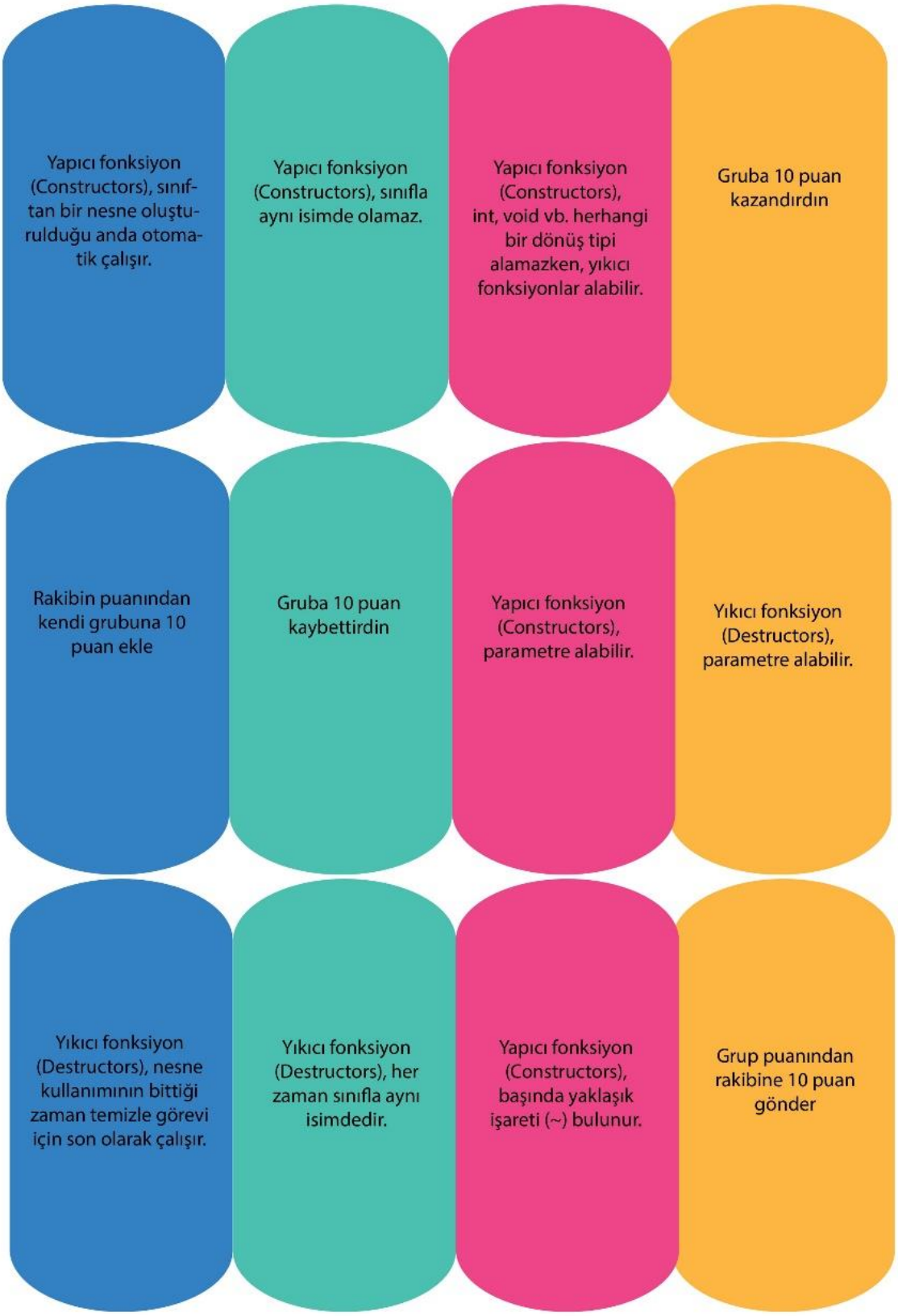
X noktasi: 61, Y noktasi: 35

n1 su anda sifir noktasinda degildir.

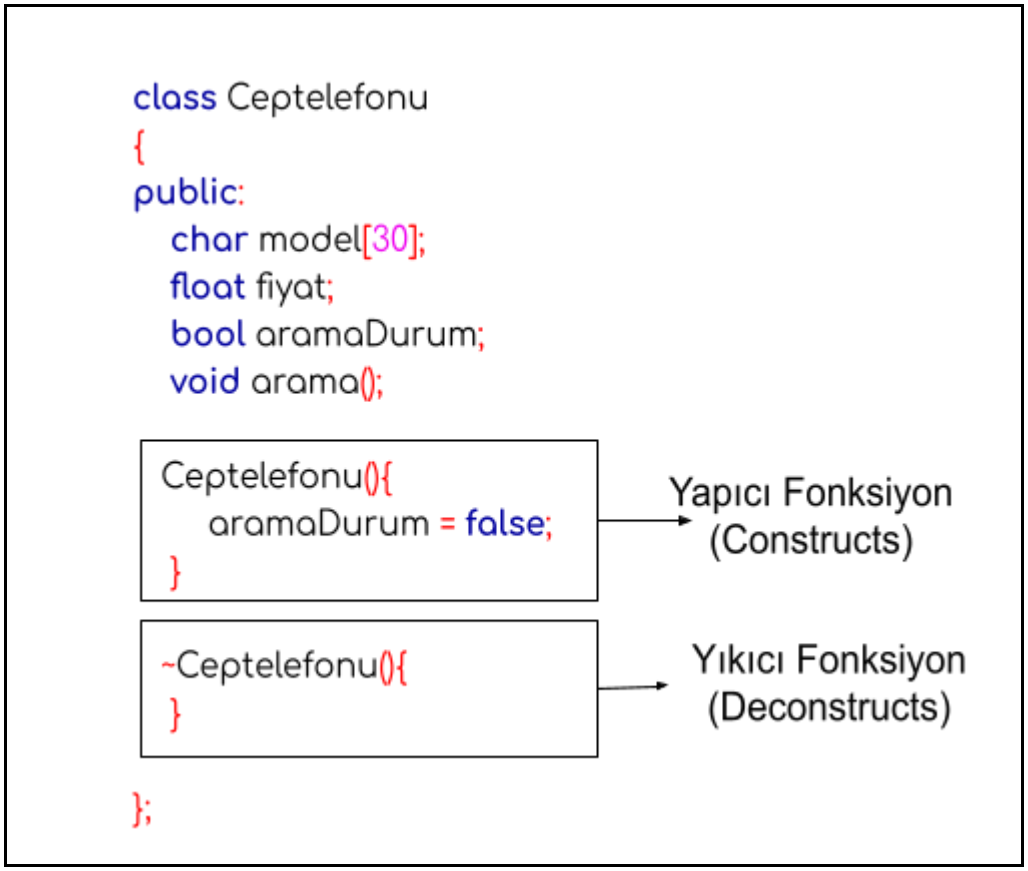
n1 su anda sifir noktasindadir.

Hafta 9. Ders Materyalleri

L9. A1. 1. Yarışma Kartları



L9. A1. 2. Kod Örneği



L9. A2. 1. Grup Görev Kartları

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    char marka[30];
    char model[30];
    int yil;
    int fiyat;
};
int main() {
    Araba araba1;
    strcpy(araba1.marka, "Suzuki");
    araba1.fiyat = 225000;
    strcpy(araba2.model, "T-Roc");
    araba2.yil = 2020;
    cout << "Araba 1: " << araba1.marka << ", " << araba1.model << ", " << araba1.yil
<< ", " << araba1.fiyat << " \n";
    return 0;
}
```

Kodun Çıktısı:

Araba 1: Suzuki, Vitara, 2016, 225000

Araba 2: Volkswagen, T-Roc, 2020, 360000

Görev: Grup 1

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    char marka[30];
    char model[30];
    int hiz(int max_hiz);
};
int Araba::
int main() {
    strcpy(araba1.model, "Astra");
    cout << "Araba: " << araba1.marka << " " << araba1.model << " \n";
    cout << "Hiz: " << araba1.hiz(190); // Fonksiyonu parametre ile cagirin
    return 0;
}
```

Kodun Çıktısı:

```
Araba: Opel Astra
Hiz: 190
```

Görev: Grup 2

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    Araba(string) {
    }
};
int main() {
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);
    cout << araba1.marka << " " << araba1.model << " " << araba1.yil
<< " " << araba1.fiyat << "\n";
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil
<< " " << araba2.fiyat << "\n";
    return 0;
}
```

Kodun Çıktısı:

```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

Görev: Grup 3

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    Araba(string x_marka, string x_model, int x_yil, int x_fiyat);
};
{
 }
int main() {
    Araba araba1("Suzuki", "Vitara", 2016, 225000);
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil
<< " " << araba2.fiyat << "\n";
    return 0;
}
```

Kodun Çıktısı:

```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

Görev: Grup 4

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

L9. A3. 1. Destekleyici kod bilgisi

Örnek Kod: Cep telefonu sınıfını, sınıf dosyaları kullanarak tanımlama.

ornek.cpp

```cpp
#include <iostream>
#include "Ceptelefonu.h"
using namespace std;

int main()
{
    Ceptelefonu urun1(4500);
    Ceptelefonu urun2 = Ceptelefonu(3750);
    cout << "Urun 1 baslangic fiyati: " << urun1.fiyat << endl;
    cout << "Urun 2 baslangic fiyati: " << urun2.fiyat << endl;
    return 0;
}
```

Ceptelefonu.h

```cpp
class Ceptelefonu
{
public:
    char marka[30];
    char model[30];
    int fiyat;
    bool aramaDurum;
    Ceptelefonu(int x_fiyat){
        fiyat = x_fiyat;
    }
    ~Ceptelefonu(){
    }
    void arama();
};
void Ceptelefonu::arama()
{
    aramaDurum = true;
}
```

L9. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 10. Nesne Yönelimli Programlamanın Prensipleri

Kazanımlar

- K1. Veri soyutlama, kapsülleme, kalıtım ve polimorfizm tekniklerini açıklar.
- K2. Kalıtım tekniğini farklı problemler içinde uygular.
- K3. Verilen programda kullanılan aşırı yükleme (overloading) tekniğini analiz eder.
- K4. Verilen programdaki geçersiz kılma/çiğneme (overriding) tekniğini yeniden tasarlar.

Amaç

Bu haftanın amacı, C++ nesne yönelimli programlamanın prensipleri olan veri soyutlama, veri kapsülleme, kalıtım, polimorfizm, aşırı yükleme ve geçersiz kılma kavramlarını öğrenmektir.

Önerilen Ders Akışı

A. Öğrenme Görevleri
   A1. Adam Asmaca (40 dk.)
      ❑ Ders Arası (5 dk.)
   A2. Kalıtımı Sürdür (45 dk.)
      ❑ Ders Arası (5 dk.)
      ❑ Motivasyon Oyunu (10 dk.)
   A3. Aşırı Yüklenenler (30 dk.)
   A4. Geçersiz Olanlar (35 dk.)
      ❑ Ders Arası (10 dk.)
B. Kısmi Öğrenme Görevleri (60 dk.)

B. Öğrenme Görevleri

A1. Adam Asmaca!

Süre: 40 dk.

Kazanımlar: K1. Veri soyutlama, kapsülleme, kalıtım ve polimorfizm kavramlarını açıklar.

Materyaller: L10. A1. Adam Asmaca Oyunu

Hazırlık: Materyal öğrencilerin tümünün görebileceği şekilde ekrana yansıtılır ve ÖYS üzerinden erişime açılır.

Uygulama: Eğitmen bu öğrenme görevine başlamadan önce nesne yönelimli programlamanın temelini oluşturan kavramları bugün inceleyeceklerini belirtir. Bu kelimeleri direkt vermeden önce tahminde bulunabilecekleri bir oyun oynayacaklarını açıklar. Sınıf iki gruba ayrılır. Daha sonra öğrencilere adam asmaca oyunu oynatılır. Eğitmen tahtaya kelimenin tanımı, kelime ile ilgili bir örnek ve görselinden oluşan ipuçlarını herkesin görebileceği şekilde tahtaya yansıtarak gruplara sırayla sorar. İpuçlarına bağlı olarak öğrencilerden kelimeyi tahmin etmelerini ister. Gerektiğinde bir harf alabilecekleri belirtilir. Öğrencilerin beş farklı harf alma hakkı vardır. İlk grup kelimeyi bilemezse, diğer gruba da harf almaksızın bir kere tahmin hakkı verilir. Kelime ortaya çıktıkça eğitmen ilgili kelime hakkındaki konuyu aşağıdaki gibi özetlemeye çalışır.

Eğitmene Öneriler: Eğitmen adam asmaca oyununu dijital ortamda tasarlayıp ve tüm öğrencilerin görebileceği şekilde oyunu tahtada başlatabilir. Bunun için ücretsiz ve basit şekilde oyunun geliştirilebileceği örnek uygulamalardan birisi https://learningapps.org/' dir. Eğitmen materyalde yer alan ipuçları ve bilinmesi gereken kelimeleri learningapps sitesinden ilgili template seçerek adam asmaca oyununu tasarlayabilir. Oyunun tam ekran paylaşım linki ise öğrencilerle Moodle üzerinden paylaşılabilir.

Konu İçeriği:

Nesne yönelimli programlamanın özü, kodda belirli özelliklere ve yöntemlere sahip bir nesne oluşturmaktır. Nesneleri kullanarak, kodun diğer bölümlerinin bu fonksiyon dışında bu verilere erişememesi için verileri ve üzerinde çalışan fonksiyonları birleştirmektir. C++ modülleri tasarlarken, tüm dünyayı nesne şeklinde görmeye çalışıyoruz. Örneğin araba; renk, kapı sayısı, hız limiti gibi belirli özelliklere sahip bir nesnedir. Farklı isimler ve markalara sahip birçok araba olabilir, ancak hepsi bazı ortak özellikleri paylaşmaktadır. Ayrıca, hızlanma ve yavaşlama gibi belirli yöntemlere de sahiptir. Nesne yönelimli programlamanın temelini oluşturan kavramlar şunlardır:

Resim 34. *Nesne yönelimli programlamanın temelini oluşturan kavramlar*

1. Veri Soyutlama

Veri soyutlama, dış dünyaya sadece gerekli bilgileri sağlama ve arka plan ayrıntılarını gizleme, yani ayrıntıları sunmadan programdaki gerekli bilgileri temsil etme anlamına gelir. Bir arabanın nasıl sürüleceğini ve nasıl yakıt ekleneceğini biliyor olabiliriz, fakat motorun nasıl çalıştığı hakkında bir fikrimiz var mı? Hayır ve ayrıca bilmemize gerek de yok çünkü bu detay soyutlanmış durumda.

Araba örneğinden devam edersek, araba kullanan adam sadece gaza basmanın arabanın hızını artıracağını veya fren yapmanın arabayı durduracağını bilir, ancak gaza bastığında hızın gerçekte nasıl arttığını ya da arabanın iç mekanizması (gaz, fren vb. çalışma prensibini) bilmez. Başka bir örnek vermek gerekirse; mahalledeki pastaneden en sevdiğiniz kurabiyelerden satın alırken kurabiyenin yapım aşamasında kullanılan unun veya şekerin markasını ya da ne kadar kabartma tozu kullanıldığı gibi bilgileri düşünmüyorsunuz. Önemli olan kurabiyenin lezzetidir ki işin asıl amacı da budur yani beğenilmesidir.

Araba (veri soyutlamasız)

Araba (veri soyutlamalı)

Resim 35. *Veri soyutlama araba örneği*

Veri soyutlama, veri üyelerini gizleme ve sınıfın kullanıcısının verileri bozmaması için bir arayüzün arkasında bir sınıfın uygulanması sürecidir. Buradaki fikir, verilerin bir sınıf için

uygulamanın içinde gizli olmasıdır. Veriler doğrudan değil, "public" fonksiyonlar aracılığıyla manipüle edilir. Genel kural, tüm değişkenler ve üyeler için mümkün olan en küçük kapsamı kullanmaktır. Ayrıca, olabildiğince az global değişken kullanın. Bunu yaparken, yalnızca verilere erişimi olması gereken kod bu verileri değiştirebilir. Başka bir kodun özel bir nesne içindeki verileri kullanması veya değiştirmesi gerekiyorsa, o nesnenin üye fonksiyonları bu tür erişime izin veren yordamları sağlayabilir.

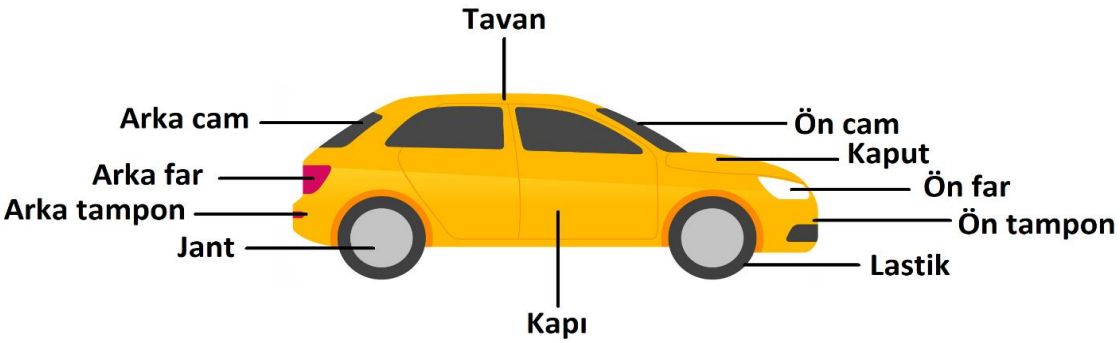
Örneğin, bir veritabanı sistemi, verilerin nasıl saklandığına, oluşturulduğuna ve muhafaza edildiğine ilişkin belirli ayrıntıları gizler. Benzer şekilde, C ++ sınıfları, bu yöntemler ve veriler hakkında dahili ayrıntılar vermeden dış dünyaya farklı yöntemler sağlar.

2. Kapsülleme

Verileri ve bu veriler üzerinde çalışan fonksiyonları aynı yere yerleştirmektir. Nesne yönelimli programlama, size verileri ve ilgili fonksiyonları aynı nesneye yerleştirmek için çerçeve sağlar. Bir sınıfın değişkenleri veya verileri başka herhangi bir sınıftan gizlenir ve yalnızca bildirildikleri kendi sınıfının herhangi bir üye fonksiyonu aracılığıyla erişilebilir. Kapsülleme fikri sınıfları birbirinden ayrı tutmak ve birbirleriyle sıkı sıkıya bağlı olmalarını önlemek için kullanılır. Oluşturduğunuz her sınıf, belirli bir şeyi veya kavramı temsil etmelidir. Birden çok sınıf, nesnelerin veya kavramların kombinasyonlarını temsil etmek için daha sonra bir araya gelir.

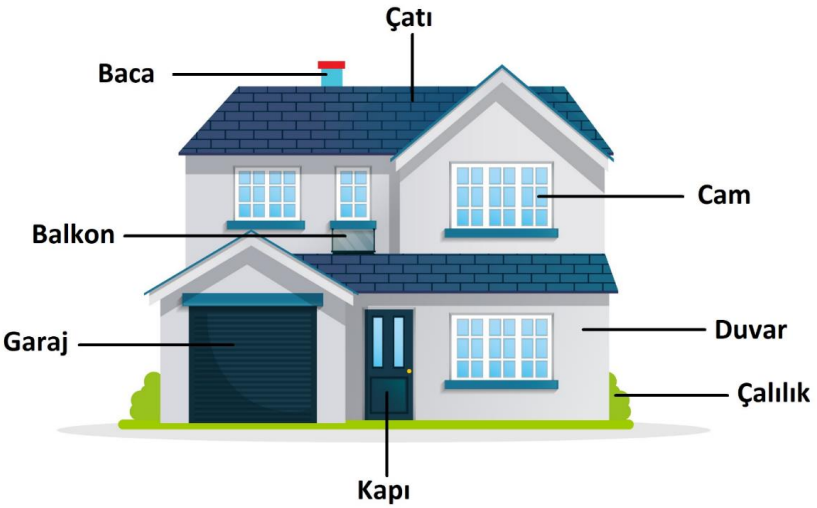
Araba örneğine tekrar bakalım. Bir arabanın hidrolik direksiyonu, dahili olarak birbirine bağlanmış çok sayıda bileşene sahip karmaşık bir sistemdir ve bu sistemdeki bileşenler aracı istenen yönde döndürmek için senkronize olarak çalışır. Motorun direksiyon simidine verdiği gücü bile kontrol eder. Ancak dış dünyaya yani bize ulaşan sadece bir arayüz mevcuttur ve karmaşıklığın geri kalanı gizlidir. Ayrıca, direksiyon ünitesi kendi içinde eksiksiz ve bağımsızdır. Başka bir mekanizmanın çalışmasını etkilemez.

Aşağıdaki gibi bir araba üretmek istediğinizi düşünün. Arabanın bitmiş hâlini oluşturmadan önce daha küçük alt parçaları oluşturmalısınız. Her küçük parça küçük bir fonksiyonu vardır, ancak hepsi bir araba üretmek için birleşir. Aşağıdaki görselde bir arabanın farklı parçaları gösterilmektedir. Her bölüm benzersiz bir amaca hizmet etmektedir.



Resim 36. *Veri kapsülleme araba örneği*

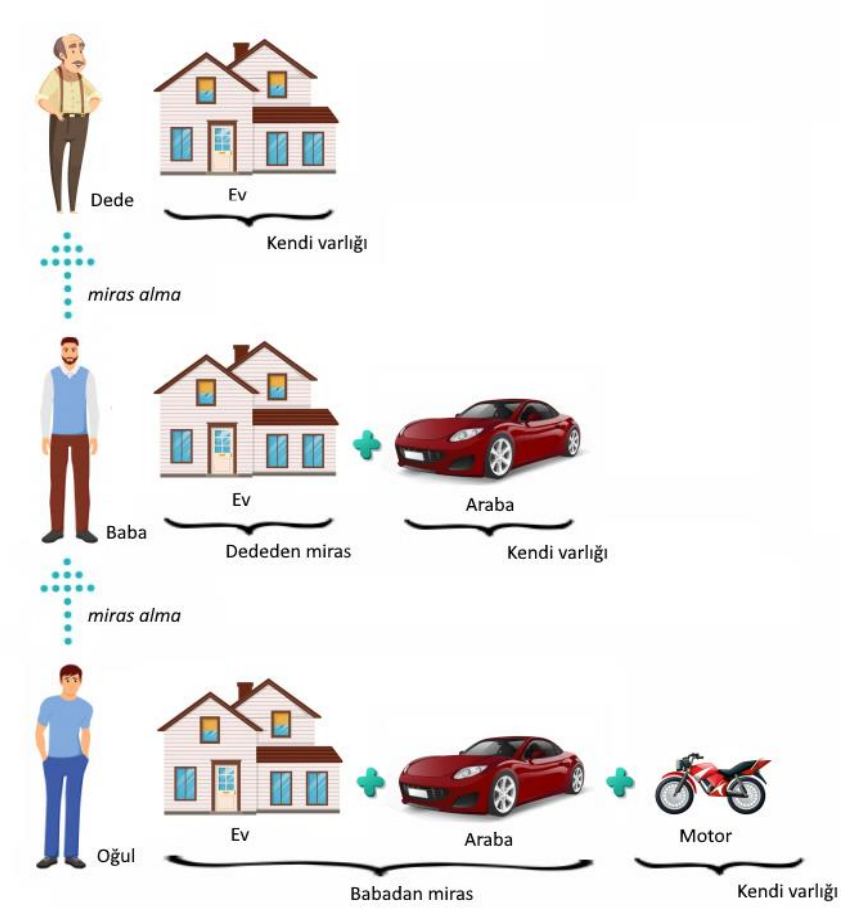
Aşağıdaki görselde ise bir evin farklı parçaları gösterilmektedir. Aynı şekilde burada da her bölüm benzersiz bir amaca hizmet etmektedir.



Resim 37. *Veri kapsülleme ev örneği*

3. Kalıtım

Kalıtım, bir nesnenin başka bir nesnenin belirli ya da tüm özelliklerini edinme mekanizmasıdır. Üst sınıfın özelliklerinin alt sınıfa aktarılmasını sağlayarak türetilmiş sınıflar oluşturur. Kodun yeniden kullanılabilirliğini sağladığı ve kod boyutunu azaltmaya yardımcı olduğu için nesne yönelimli programlamada çok önemli bir kavramdır.



Resim 38. *Kalıtım örneği*

Üyeleri temel bir sınıftan devralma yeteneği, temel sınıfta tanımlanan belirli ortak özellikleri paylaşan türetilmiş sınıfların oluşturulmasına izin verir.

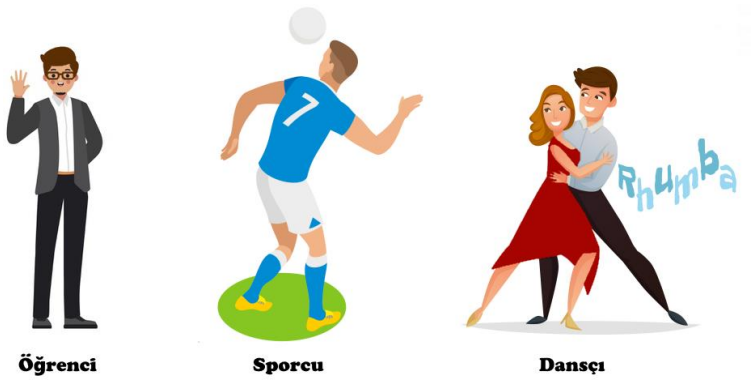
UYARI:

Sınıf örneklerini ve türetilmiş sınıfları birbiri ile karıştırmayın. Örnek, bir sınıfın kopyasıdır, türetilmiş sınıf ise türetildiği temel sınıfın özelliklerini miras alan yeni bir sınıftır.

Örneğin otomobiller, kamyonlar ve motosikletler dahil olmak üzere motorlu taşıtlarla ilgili bir uygulamanın gerektiğini varsayalım. Program, her türlü aracı temsil etmek için "taşıt" adlı bir sınıf kullanabilir. Arabalar, kamyonlar ve motosikletler taşıt türleri olduğundan, "taşıt" sınıfının alt sınıfları ile temsil edilecektir. "taşıt" sınıfı tüm taşıtlar için ortak olan plaka ve sahibi gibi genel bilgilere sahip olabilir. Bunlar tüm taşıtlarda ortak olan bilgilerdir. Ancak her bir taşıta özel bir değişken de eklenebilir. Araba sınıfı kaç kapılı bilgisini, kamyon sınıfı tekerlek sayısını ve motosiklet sınıfı da motosiklet sepeti olup olmadığı bilgisini alabilir.

4. Polimorfizm (Çok Biçimlilik)

Bir işleci veya fonksiyonu farklı şekillerde kullanma yeteneğine, başka bir deyişle işleçlere veya fonksiyonlara farklı anlam veya özellikler verme yeteneğine polimorfizm denir. Poli birçok anlamına gelirken morf ise form anlamındadır. Yani bir nesnenin aynı sürece farklı şekillerde yanıt verebilme yeteneğidir. Tek bir fonksiyon veya kullanımdan farklı şekillerde fonksiyon gören bir operatöre polimorfizm denir. Daha iyi anlamak için aşağıdaki gerçek hayat örneğine bakalım. Bir kişi sınıfta kendini öğrenci, sahada bir sporcu ve dans kulübünde bir dansçı olarak temsil ediyor.



Resim 39. *Polimorfizm kişi örneği*

Aynı araba örneğine bakalım. Bir araçta vites iletim sistemi vardır. Dört ön vites ve bir arka vites vardır. Motor hızlandığında, hangi vitese geçildiğine bağlı olarak, araca farklı miktarda güç ve hareket verilir. Eylem vites değiştirme olarak aynıdır, ancak vites tipine bağlı olarak eylem farklı davranır. İşte bu durumda polimorfizm mantığı devreye girmiş olur. Polimorfizm

mantığını aşağıdaki örneği kullanılarak da açıklayabiliriz. Aşağıdaki verilen nesnelerin doğasında *ses_cikar()* yöntemi vardır, ancak her canlı bunu farklı şekilde gerçekleştirir.

Resim 40. *Polimorfizm ses çıkarma örneği*

A2. Kalıtımı Sürdür!

Süre: 45 dk.

Kazanımlar: K2. Kalıtım tekniğini farklı problemler içinde uygular.

Materyaller: L10. A2. Kalıtım Örnekleri

Hazırlık: Materyalden iki çıktı alınır ve kod ve görev birlikte olacak şekilde dört grup materyali olarak kesilir.

Uygulama: Sınıfta dört grup oluşturulur. Uygulama iki etaplıdır. İlk etapta iki gruba birinci görev, diğer iki gruba ise ikinci görev dağıtılır. 10 dk. süre ile görevi tamamlamaya çalışırlar. Süre sonunda görev 1'i alan gruplar yaptıklarını birbiriyle takas eder. Aynı şekilde görev 2'yi alanlar da yaptıklarını kendi içlerinde takas eder. Bu şekilde gruplar birbirlerinin yaptıklarını düzenler ve akran öğrenimi gerçekleştirilir. Takas süreci sonrası düzenlemeler için öğrencilere 5 dk. verilir ve ilk etap tamamlanır. İkinci etapta, ilk etabın süreci izlenir. Tek fark ilk etapta görev 1'i alanlara, görev 2 verilirken, görev 2 alan gruplara ise görev 1 dağıtılır. Eğitmen burada grup süreleri için bir çan ya da zil gibi materyal kullanabilir. Öğrencilerin İkinci etapta görev 1 gruplarının, görev 2'dekilerle ya da tam tersi olacak şekilde grup yanıtlarını paylaşmamaları gereklidir. İki etapta tamamlandıktan sonra görevler sınıfça tartışılır ve eğitmen tarafından özetlenir.

Konu İçeriği: Çokgenler üzerinde kalıtım örneğindeki eksik kodun tamamlanmış hâli aşağıdadır.

```
#include <iostream>
using namespace std;
```

```
class Cokgen
{
protected:
    int genislik, yukseklik;
public:
    void degerAta(int g, int y) {
        genislik = g;
        yukseklik = y;
    }
};

class Dikdortgen: public Cokgen
{
public:
    int alan() {
        return (genislik * yukseklik);
    }
};

class Ucgen : public Cokgen
{
public:
    int alan() {
        return ((genislik * yukseklik)/2);
    }
};

int main()
{
    Dikdortgen dik;
    Ucgen ucg;
    dik.degerAta(4, 5);
    ucg.degerAta(4, 5);
    cout << "Dikdorgen Alani: " << dik.alan() << endl;
    cout << "Ucgen Alani: " << ucg.alan() << endl;
return 0 ;
}
```

Kodun Çıktısı:

```
Dikdorgen Alani: 20
Ucgen Alani: 10
```

Hayvan sınıfı üzerinde uygulanan kalıtım örneğindeki eksik kodun tamamlanmış hâli aşağıdadır.

```
#include <iostream>
using namespace std;

class Hayvan {
public:
    void yeme() {
        cout << "Yemek yiyebilirim.." << endl;
    }

    void uyuma() {
        cout << "Uyuyabilirim.." << endl;
    }
};

class Kopek : public Hayvan {
public:
    void havlama() {
        cout << "Havlayabilirim.. Hav hav!!" << endl;
    }
};

int main() {
    Kopek kopek1;
    kopek1.yeme();
    kopek1.uyuma();
    kopek1.havlama();
    return 0;
}
```

Ekran Çıktısı:

Yemek yiyebilirim..

Uyuyabilirim..

Havlayabilirim.. Hav hav!!

A3. Aşırı Yüklenenler

Süre: 30 dk.

Kazanımlar: K3. Verilen programda kullanılan aşırı yükleme (overloading) tekniğini analiz eder.

Materyal: L10. A3. Aşırı Yüklenenler

Hazırlık: Materyalden iki kopya çıktı alınır ve kod ve görev birlikte olacak şekilde dört grup materyali olarak kesilir.

Uygulama: Eğitmen uygulama öncesi polimorfizm kavramı hakkında hatırlatma yapar. İki tür polimorfizm olduğunu aşağıdaki gibi öğrencilere açıklamada bulunur.

> *Polimorfizm, nesnenin farklı koşullarda farklı davranmasına izin veren poliformizm kavramını hatırlayınız. Bu ve bir sonraki uygulamada polimorfizmi daha yakından tanıyacağız. C++'da iki tür polimorfizm vardır:*
>
> *1) Fonksiyon aşırı yükleme (Derleme zamanı polimorfizmi)*
>
> *2) Fonksiyon geçersiz kılma (Çalışma Zamanı polimorfizmi)*
>
> *İlk olarak fonksiyon aşırı yükleme türünü inceleyeceğiz.*

Yukarıdaki gibi açıklama yapıldıktan sonra eğitmen sınıfı dörde ayırır. İlk iki gruba görev 1, diğer iki gruba görev 2 verilir ve kendi içlerinde görevleri tartışmaları beklenir. Eğitmen bunun için gruplara 5 dk. verir. Eğitmen süre sonunda görev 1 ve görev 2'yi alan iki grubun birleşerek, görev 1-görev2 grup üyeleri şeklinde ikişerli yeni gruplar oluşturmalarını ister. Yeni çiftler hâlinde birleşen öğrenciler görev 1 ve görev 2 grubundaki tartışmalarını birbirlerine açıklarlar. Bu şekilde akran öğrenimini destekleyen 5 dk. daha geçer. Son olarak eğitmen eşliğinde görevler sınıfça tartışılır.

Konu İçeriği: Aşırı Yükleme (Overloading)

Aynı isimde iki ya da daha fazla fonksiyonun farklı tür ve sayıda parametre kullanılarak yeniden tanımlanmasına *fonksiyon aşırı yükleme* denir. Fonksiyon aşırı yüklemesinin avantajı, aynı fonksiyon için farklı isimler kullanmayı gerektirmediği için programın okunabilirliğinin artmasıdır.

Görev 1: Farklı parametre sayısı kullanarak fonksiyon aşırı yükleme.

```
#include <iostream>
using namespace std;
class Topla
{
public:
    int ekle(int a,int b){
        return a + b;
    }
    int ekle(int a, int b, int c)
    {
        return a + b + c;
    }
};
int main(void) {
    Topla f;
    cout << f.ekle(21, 13) << endl;
    cout << f.ekle(21, 13, 30) << endl;
   return 0;
}
```

Kodun Çıktısı:

```
34
64
```

Görev 2: Farklı parametre türü kullanarak fonksiyon aşırı yükleme.

```
#include <iostream>
using namespace std;

class asiriYukleme
{
public:
    void yaz(int x) {
        cout << "Integer deger: " << x << endl;
    }
```

```
    void yaz(double  y) {
        cout << "Double deger: " << y << endl;
    }
    void yaz(char z) {
        cout << "Char deger: " << z << endl;
    }
};
int main() {
    asiriYukleme ornek;
    ornek.yaz(45);
    ornek.yaz(34.78);
    ornek.yaz('A');
   return 0;
}
```

Kodun Çıktısı:

```
Integer deger: 45
Double deger: 34.78
Char deger: A
```

A4. Geçersiz Olanlar

Süre: 35 dk.

Kazanımlar: K4. Verilen programdaki geçersiz kılma/çiğneme (overriding) tekniğini yeniden tasarlar ve anlar.

Materyal: L10. A4. Destekleyici Örnek

Hazırlık: Materyalden ÖYS üzerinden öğrenci erişimine açılır.

Uygulama: Eğitmen uygulama öncesi aşağıdaki gibi öğrencilere açıklama yapar.

> *Bir önceki uygulamada Polimorfizmin türleri olduğunu gördük. Bunlardan ilki olan fonksiyon aşırı yükleme kavramını yakından inceledik. Şimdi ikinci poliformizm türü olan fonksiyon geçersiz kılma türünü bu uygulama ile inceleyeceğiz.*

Yukarıdaki açıklamayı yaptıktan sonra, eğitmen öğrencilere materyali verir. Öğrenciler dörderli gruplar hâlinde çalışacaktır. Eğitmen grupların Moodle üzerinden erişime açılan materyali incelemelerini ve 5 dk. süre ile aralarında geçersiz kılma kavramını tartışmalarını ister. Materyalde fonksiyon geçersiz kılma üzerine bir kod örneği ve bu örneğe ilişkin geçersiz kılma tekniği anlatılmıştır. Eğitmen daha sonra öğrencilere örnekteki kod satırlarını kendi geçersiz kılma örnekleriyle revize etmelerini ister. Bu şekilde gruplar geçersiz kılma tekniğini

uyguladıkları yeni bir örnek tasarlamış olacaktır. Bunun için eğitmen gruplardan bir temel sınıf ile o sınıftan türetilmiş bir başka sınıf altında geçersiz kılınacak bir fonksiyon düşünerek örnek kodu bilgisayarda uygulamalarını ister. Gruplar yeni örneklerini oluşturduktan sonra sınıfta eğitmen eşliğinde geçersiz kılma tekniği özetlenir.

Eğitmene Öneriler: Eğitmen öğrencilere temel sınıf ve türetilmiş sınıf hakkında spesifik bir örnek verebilir. Öğretmen temel sınıfı altındaki brans fonksiyonunu türetilmiş Matematik öğretmeni sınıfı ile geçersiz kılma gibi.

Konu İçeriği: Geçersiz Kılma/Çiğneme (Overriding)

Temel sınıftaki bir fonksiyonu geçersiz kılmak için o fonksiyonu türetilmiş bir sınıfta tekrar aynı isim, parametre ve dönüş türüne sahip olacak şekilde kullanırız. Bu şekilde, örnekte Temel sınıfı altındaki yaz() fonksiyonunu etkili bir şekilde gizlenmiş ya da geçersiz kılınmıştır. Bununla birlikte Temel sınıfı altındaki yaz() fonksiyonunu çağırabilmek için :: kapsam çözümleme operatörü kullanılabilir. Temel sınıf fonksiyonlarını geçersiz kılma tekniği, aşırı yüklenmiş fonksiyonları istemeden gizlemekten kaçınmak için dikkatle kullanılmalıdır. Türetilmiş bir sınıftaki tek bir geçersiz kılma fonksiyonu ile temel sınıfta aynı isme sahip tüm aşırı yüklenmiş fonksiyonlar gizlenecektir.

```
#include <iostream>
using namespace std;
class Temel
{
public:
    void yaz() {
        cout << "Temel Fonksiyon" << endl;
    }
};
class Turetilmis : public Temel
{
public:
    void yaz() {
        cout << "Turetilmis Fonksiyon" << endl;
    }
};
int main()
{
    Turetilmis turet1, turet2;
    turet1.yaz();
    turet2.Temel::yaz();
    return 0;
}
```

Kodun Çıktısı:

```
Turetilmis Fonksiyon

Temel Fonksiyon
```

Ogretmen temel sınıfından türetilen *Matematik_Ogretmeni* sınıfı altında yer alan *brans()* fonksiyonu ile oluşturulmuş geçersiz kılma tekniği:

```
#include <iostream>
using namespace std;
class Ogretmen
{
public:
    void brans(){
        cout << "Sinif Ogretmeni" << endl;
    }
};
class Ogretmen : public MatematikOgretmeni
{
public:
    void brans(){
       cout << "Matematik Ogretmeni" << endl;
    }
};
int main(void)
{
   MatematikOgretmeni o1, o2;
   o1.brans();
   o2.Ogretmen::brans();
   return 0;
}
```

Kodun Çıktısı:

```
Matematik Öğretmeni
Sınıf Öğretmeni
```

C. Kısmi Öğrenme Görevleri

Süre: 60 dk.

Materyal: L10. B. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilere ÖYS ortamında süreli ödev olarak açılır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Bir görevi doğru yapan öğrencilere, o göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi Öğrenme Görevleri Yanıtlar: Görevler, öğrenciye verilecek beceri rozetleri ile isimlendirilmiştir. Her bir rozetin altında ilgili görevin yanıtları verilmektedir. Eğitmen bu yanıtları uygulama süresinin sonunda GitHub üzerinden öğrencilerle paylaşır.

Tasarlayıcı: Bir meyve sınıfı oluşturup içerisinde ekrana “Ben meyve sinifiyim” diyen bir meyve sınıfından türetilen Elma ve Muz sınıflarını oluşturunuz. Türetilmiş bu sınıfların içerisinde meyvelerin renklerini ekrana yazdırıp, tüm sınıflar için birer nesne tanımlayınız.

```
#include <iostream>
using namespace std;
class Meyve {
public:
    Meyve(){
        cout<<"Ben meyveyim."<<endl;
  }
};
class Elma: public Meyve {
public:
    Elma(){
        cout<<"Ben kirmizi renkliyim!!"<<endl;
    }
};
class Muz: public Meyve {
public:
```

```
    Muz(){
        cout<<"Ben sari renkliyim!!"<<endl;
    }
};
int main() {
    Elma e;
    Muz m;
}
```

Ekran Çıktısı:

Ben meyve sinifiyim.

Ben kirmizi renkliyim!!

Ben meyve sinifiyim.

Ben sari renkliyim!!

Denetleyici: Verilen kod satırlarını inceleyerek aşırı yüklenen fonksiyonun hangisi olduğunu ve kodun ekran çıktısını tahmin ediniz.

```
#include <iostream>
using namespace std;
class Dikdortgen{
public:
    void alanYaz1(int x, int y){
        cout << x * y << endl;
    }
    void alanYaz2(int x){
        cout << x * x << endl;
    }
    void alanYaz1(int x, double y){
        cout << x * y << endl;
    }
    void alanYaz3(double x){
        cout << x * x << endl;
    }
};
int main(){
    Dikdortgen dikt;
```

```
    dikt.alanYaz1(3,5);
    dikt.alanYaz1(4,2.1);
    dikt.alanYaz2(8);
    dikt.alanYaz3(6.7);
    return 0;
}
```

CEVAP: Farklı parametre türünde aşırı yüklenen fonksiyon alanYaz1'dir. Kodun ekran çıktısı ise aşağıdaki gibidir:

```
15
8.4
64
44.89
```

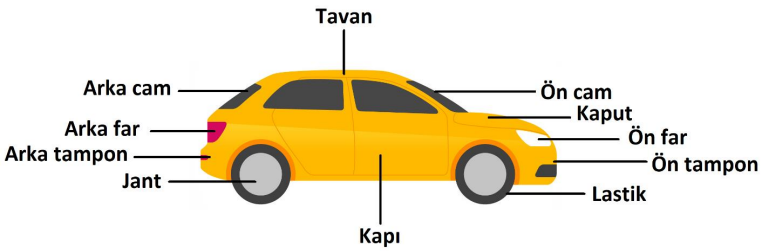
Kodlayıcı: Hayvan sınıfından türetilen bir Kopek sınıfı oluşturunuz. Bu sınıflara ait iki farklı nesne tanımlayarak, yer() isimli fonksiyonu geçersiz kılan (overriding) kod satırlarını oluşturunuz.

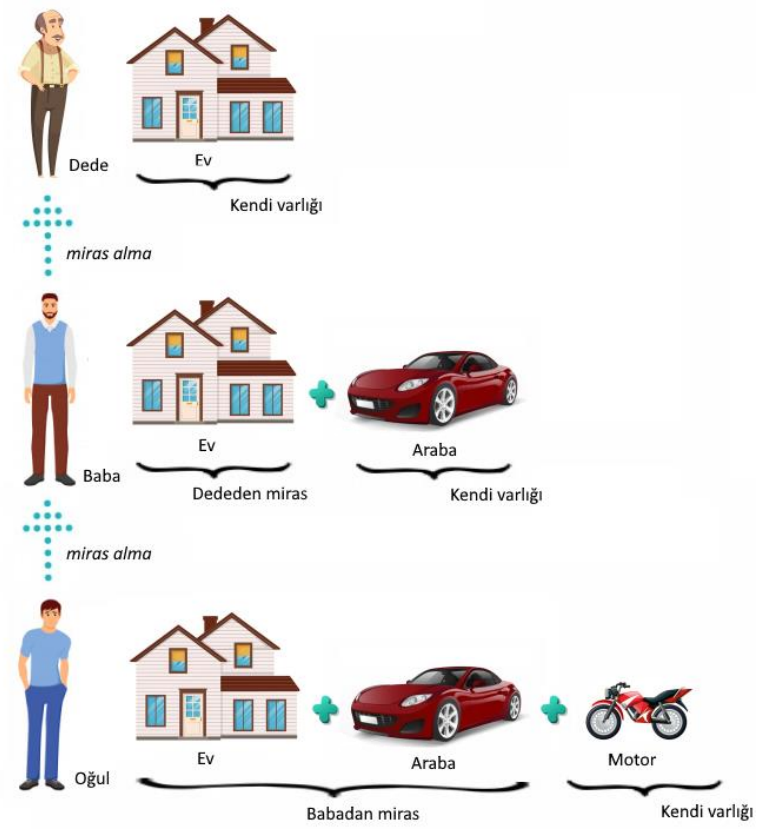
```
#include <iostream>
using namespace std;
class Hayvan
{
public:
    void yer(){
        cout << "Yer..." << endl;
    }
};
class Kopek: public Hayvan
{
public:
    void yer(){
       cout << "Ekmek yer..." << endl;
    }
};
int main(void)
{
   Kopek k1, k2;
```

```
    k1.yer();
    k2.Hayvan::yer();
    return 0;
}
```

Hafta 10. Ders Materyalleri

L10. A1. Adam Asmaca Oyunu

İpucu		Kelime
1.	  Dış dünyaya sadece gerekli bilgileri sağlama ve arka plan ayrıntılarını gizleme, yani ayrıntıları sunmadan programdaki gerekli bilgileri temsil etme anlamına gelir. Buradaki fikir, verilerin bir sınıf için uygulamanın içinde gizli olmasıdır. Örneğin bir arabanın nasıl sürüleceğini ve nasıl yakıt ekleneceğini biliyor olabiliriz, fakat bu bilgiler için motorun nasıl çalıştığını bilmek zorunda değiliz.	... ...
		

2.	Sınıfları birbirinden ayrı tutmak ve birbirleriyle sıkı sıkıya bağlı olmalarını önlemek için kullanılır. Oluşturduğunuz her sınıf, belirli bir şeyi veya kavramı temsil etmelidir. Birden çok sınıf, nesnelerin veya kavramların kombinasyonlarını temsil etmek için daha sonra bir araya gelir. Örneğin, yukarıdaki gibi bir arabanın bitmiş hâlini oluşturmadan önce daha küçük alt parçaları oluşturmalısınız. Her küçük parça küçük bir işlevi vardır, ancak hepsi bir araba üretmek için birleşir. Aşağıdaki görselde bir arabanın farklı parçaları gösterilmektedir. Her bölüm benzersiz bir amaca hizmet etmektedir.	… … … … … … … … … …
3.		… … … … … … …

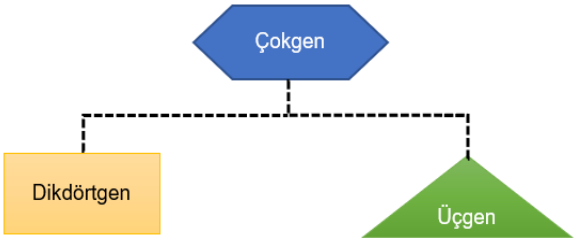
	Bir nesnenin, başka bir nesnenin belirli ya da tüm özelliklerini edinme mekanizmasıdır. Başka bir ifadeyle veri üyelerini temel bir sınıftan devralma yeteneğidir. Üst sınıfın özelliklerinin alt sınıfa aktarılmasını sağlayarak türetilmiş sınıflar oluşturur. Kodun yeniden kullanılabilirliğini sağlar ve kod boyutunu azaltmaya yardımcıdır.	
4.	 **Öğrenci**  **Sporcu**  **Dansçı** Bir nesnenin farklı koşullarda farklı davranmasına izin verme durumudur. Başka bir ifadeyle tek bir işlev veya kullanımdan farklı şekillerde işlev gören bir operatördür. Örneğin bir kişi sınıfta kendini öğrenci, sahada bir sporcu ve dans kulübünde bir dansçı olarak temsil edebilir ya da bir adam sınıfta öğretmen, evde baba, markette müşteri gibi davranır. Burada tek bir kişi duruma göre farklı davranmaktadır.	… … … … … … … … … … … …

L10. A2. Kalıtım Örnekleri

Kod

```
#include <iostream>
using namespace std;

class Cokgen
{
protected:
    int genislik, yukseklik;
public:
    void degerAta(int g, int y) {
        genislik = g;
        yukseklik = y;
    }
};

class Dikdortgen: public Cokgen
{
public:
    int alan() {
        return (genislik * yukseklik);
    }
};

int main()
{
    Dikdortgen dik;
    dik.degerAta(4, 5);
    cout << "Dikdortgen Alani: " << dik.alan() << endl;
return 0 ;
}
```

Görev 1



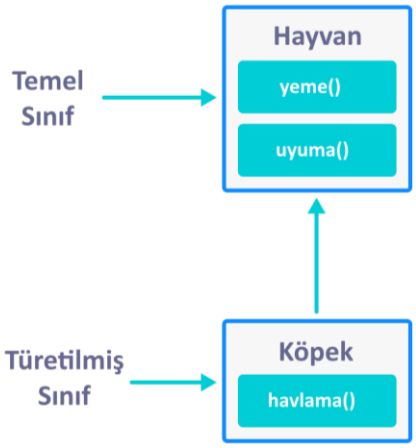
Yanda "Çokgen" temel sınıfının ortak genişlik ve yükseklik özellikleri tanımlanmıştır. Buna göre genişlik ve yükseklik özelliklerini kalıtım yoluyla Çokgen sınıfından alan "Dikdörtgen" ve "Üçgen" sınıfını türeten kod satırlarından eksik olanları tamamlayınız. Türetilen sınıfların alan hesaplarını ekran çıktısı olarak yazdırınız.

Kodun Çıktısı:

```
Dikdortgen Alani: 20
Ucgen Alani: 10
```

Kod	Görev 2

```
#include <iostream>
using namespace std;

class Hayvan {
public:
    void yeme() {
        cout << "Yemek yiyebilirim.." << endl;
    }

};

class Kopek : public Hayvan {
    void havlama() {
        cout << "Havlayabilirim.. Hav hav!!" << endl;
    }
};

int main() {
    Kopek kopek1;
    kopek1.havlama();
    return 0;
}
```



Görsele ilişkin yanda verilen kod üzerinde Köpek sınıfı "Hayvan" sınıfından türetilmiştir. Bu nedenle hayvan sınıfının üyelerine köpek sınıfı tarafından erişilebilir. Buna göre aşağıda verilen ekran çıktısını görmek için yandaki kod üzerinde eksik olan fonksiyonu ve gerekli satırları tamamlayınız.

Kodun Çıktısı:

```
Yemek yiyebilirim..
Uyuyabilirim..
Havlayabilirim.. Hav hav!!
```

L10. A3. Aşırı Yüklenenler

Kod	Görev 1
``#include <iostream>```	Aynı isimde iki ya da daha fazla fonksiyonun farklı tür ve sayıda parametre kullanılarak yeniden tanımlanmasına *fonksiyon aşırı yükleme* denir.
``using namespace std;```	
``class Topla```	
``{```	
``public:```	
``int ekle(int a,int b){```	
``return a + b;```	
``}```	
``int ekle(int a, int b, int c)```	
``{```	
``return a + b + c;```	
``}```	
``};```	
``int main(void) {```	
``Topla f;```	
``cout << f.ekle(21, 13) << endl;```	
``cout << f.ekle(21, 13, 30) << endl;```	
``return 0;```	
``}```	

Yukarıdaki tanıma göre yanda verilen kodu bilgisayarda yazarak ekran çıktısını bulunuz. Bu kod satırlarına göre aşırı yükleme yapılan fonksiyon hangisidir? Bu fonksiyonun özelliği nedir? Tartışınız.

Kod	Görev 2
`` #include <iostream> using namespace std; class asiriYukleme { public: void yaz(int x) { cout << "Integer deger: " << x << endl; } void yaz(double y) { cout << "Double deger: " << y << endl; } void yaz(char z) { cout << "Char deger: " << z << endl; } }; int main() { asiriYukleme ornek; ornek.yaz(45); ornek.yaz(34.78); ornek.yaz('A'); return 0; } ```	Aynı isimde iki ya da daha fazla fonksiyonun farklı tür ve sayıda parametre kullanılarak yeniden tanımlanmasına *fonksiyon aşırı yükleme* denir. Yukarıdaki tanıma göre yanda verilen kodu bilgisayarda yazarak ekran çıktısını bulunuz. Bu kod satırlarına göre aşırı yükleme yapılan fonksiyon hangisidir? Bu fonksiyonun özelliği nedir? Tartışınız.

L10. A4. Destekleyici Örnek

Kod	Bilgi
1. #include <iostream> 2. using namespace std; 3. class Temel 4. { 5. public: 6. void yaz() { 7. cout << "Temel Fonksiyon" << endl; 8. } 9. }; 10. class Turetilmis : public Temel 11. { 12. public: 13. void yaz() { 14. cout << "Turetilmis Fonksiyon" << endl; 15. } 16. }; 17. int main() 18. { 19. Turetilmis turet1, turet2; 20. turet1.yaz(); 21. turet2.Temel::yaz(); 22. return 0; 23. } *Kodun Çıktısı:* Turetilmis Fonksiyon Temel Fonksiyon	Temel sınıftaki bir fonksiyonun, türetilmiş sınıfta aynı isim, parametre ve dönüş türüne sahip olacak şekilde tekrar kullanılmasına fonksiyonu geçersiz kılma denir. Yandaki örnekte **Temel** sınıfı altındaki **yaz()** fonksiyonu tanımlıdır. Bu fonksiyon 13. satırda **Turetilmis** sınıf altında da aynı şekilde tekrar kullanılmış yani temel sınıftaki geçersiz kılınmıştır. Bu nedenle kodun çıktısında 20. satırdaki turet1.yaz() komutu ile "Temel" sınıfındaki fonksiyon yerine, "Turetilmis" sınıfındaki fonksiyon çağırılır. Bununla birlikte Temel sınıfı altındaki yaz() fonksiyonunu çağırabilmek için 21. satırdaki gibi :: kapsam çözümleme operatörü kullanılabilir.

L10. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 11. C++ Programlama Dilinde Kütüphane Kullanımı ve Dosyalama İşlemleri

Kazanımlar

- K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.
- K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.
- K3. Dosyalama işleminin gerekliliğini açıklar.
- K4. Dosya kütüphanesi kullanarak program geliştirir.
- K5. Dosya okuma işlemlerini içeren program tasarlar.
- K6. Dosya yazma işlemlerini içeren program tasarlar.

Amaç

Bu haftanın amacı öğrencilerin C++ programlama dili içerisinde bulunan kütüphaneleri kullanma ve dosyalama işlemleri yapabilmesini sağlamaktır.

Önerilen Ders Akışı

A. Giriş (10 dk.)
B. Öğrenme Görevleri
   B1. C++ Programlama Dilinde Yerleşik Kütüphaneleri Keşfediyorum (20 dk.)
   B2. Eksik Kodları Dolduruyorum (20 dk.)
   - ❑ Ders Arası (10 dk.)

   B3. Kütüphanelerdeki Bazı Fonksiyonları Kullanarak Kodluyorum (30 dk.)
   B4. Neden Dosyalama İşlemleri Yaparız? (30 dk.)
   - ❑ Ders Arası (10 dk.)

   B5. C++ Dilinde Dosyalama İşlemleri Yapıyorum (20 dk.)
   B6. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum (30 dk.)
   - ❑ Ders Arası (10 dk.)

C. Kısmi Öğrenme Görevleri (50 dk.)

A. Giriş:

Süre: 10 dk.

Uygulama: Sınıfta bulunan her öğrenci kendine bir oyun arkadaşı seçer. Herkes seçtiği oyun arkadaşı ile taş, kâğıt ve makas oyununu karşılıklı oynar, oyunu kaybeden kazandığı kişinin arkasına geçerek onun takipçisi olur. Kazananlar sürekli bir diğer kazananla karşılaşarak aynı şekilde taş, kâğıt ve makas oyununu tekrar eder, kaybeden her kişi kazananın arkasına geçmektedir ve kazananın takipçisi olmaktadır. En uzun kuyruğa ulaşan bir başka ifadeyle hiç kaybetmeyen kişi oyunu kazanır.

Eğitmene Öneriler: Oyundaki amaç kazananı belirlemekten çok grup dinamiğini canlı tutmaktır. Oyunu kaybeden öğrenci, kazananın arkasına geçmektedir ve kaybettiği arkadaşı bir sonraki diğer kazananla yarıştığında arkasında bulunduğu için aslında farkında olmadan kaybettiği arkadaşına destek vermektedir.

B. Öğrenme Görevleri

B1. C++ Programlama Dilinde Yerleşik Kütüphaneleri Keşfediyorum

Süre: 20 dk.

Kazanımlar: K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: L11.B1.1 C++ Programlama Dilinde Kütüphaneler

L11.B1.2 Karakter Kütüphanesi

L11.B1.3 Metin Kütüphanesi

Hazırlık: Birinci materyal ÖYS üzerinden erişime açılır. Diğer iki materyalin ise 10'ar tane çıktısı alınır.

Uygulama: Eğitmen öncelikle her bir grup dörderli olacak şekilde sınıfı beş gruba böler. Materyalleri dağıtır ve ÖYS üzerinden erişilecek materyali açtırır. Bu materyallerden "L11.B1.1 C++ Programlama Dilinde Kütüphaneler" öğrenciye ve eğitimcilere kütüphanelerin ne olduğuna yönelik derse giriş yapılması için hazırlanmıştır. Diğer iki materyalde ise noktalı boş olan yerleri öğrenci gruplarının doldurarak kütüphane ve o kütüphanedeki kullanılabilecek hazır fonksiyonları öğrencilerin gerek materyalle gerek akranlarıyla etkileşime girerek öğrenmesi amaçlanmaktadır. Eğitmenlerin bu etkinlikteki görevi öğrenci gruplarını sırayla gezerek ihtiyaç duydukları anda geri bildirimler vererek onları desteklemektir.

Eğitmene Öneriler: Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir. Cevaplar kırmızı renkte verilmiştir.

Fonksiyon	İşlevi	Örnek Kullanım	Sonuç
isalpha(c)	c karakteri eğer bir harf ise geriye true değilse false döndürür.	isalpha('2')	F
isdigit(c)	c karakteri eğer bir rakam ise geriye true değilse false döndürür.	isdigit('2')	T
isalnum(c)	c karakteri eğer bir rakam veya harf ise geriye true değilse false döndürür.	isalnum('*')	F
islower(c)	c karakteri eğer bir küçük harf ise geriye true değilse false döndürür.	islower ('d')	T
isupper(c)	c karakteri eğer bir büyük harf ise geriye true değilse false döndürür.	isupper('H')	T
tolower(c)	c karakterini küçük harfe çevirir.	tolower('E')	e
toupper(c)	c karakterini büyük harfe çevirir.	toupper('g')	G
strlen (s1)	s1 katarının uzunluğunu döndürür.	s1 = "Merhaba" strlen (s1)	7
strcpy (s1, s2)	s2 katarını s1 katarına kopyalar.	s1= "Merhaba" s2= "Dunya" strcpy (s1, s2)	Dunya
strcat (s1, s2)	s2 katarını s1 katarının sonuna ekler.	s1= "Merhaba" s2= "Dunya" strcat (s1, s2)	MerhabaDunya

strcmp (s1, s2)	s1 ve s2 aynı ise 0 değerini döndürür; Eğer alfabetik olarak s1, s2 metninden önce geliyorsa -1, sonra geliyorsa 1 değerini döndürür.	s1= "Merhaba" s2= "Dünya" strcmp (s1, s2)	M harfi D harfinden sonra geldiği için 1 değerini döndürür.

B2. Eksik Kodları Dolduruyorum

Süre: 20 dk.

Kazanımlar: K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: L11. B2. 1 C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan materyali ÖYS üzerinden paylaşır. Code Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Öğrenciler ikişerli gruplar hâlindedir. Eğitmenler "L11.B2.1 C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma" adlı materyali bilgisayar başında oturan öğrencilerden ÖYS'de açmalarını ve boşluk bulunan yerleri tahmin etmelerini ister. Eğitmenlerden biri sınıfta öğrenci gruplarını dolaşarak onların ihtiyaç duydukları anda işlemsel bilgiyi verir. Eğitmenlerden bir diğeri ise öğrencilerin görebilecekleri şekilde bilgisayarda kodları yazar. Öğrenci ihtiyaç duyduğu anda yazılan kodlara bakabilir.

Eğitmene Öneriler: Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir.

Görev 1: 'a' karakteri ile ilgili örnek kullanım ve ekran çıktısı aşağıdaki gibidir.

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char c='a';

    if(isalpha(c))
        cout << "Bu bir harftir."<<endl;
    else
```

```
        cout << "Bu bir harf degildir."<<endl;

    if(isdigit(c))
        cout << "Bu bir rakamdir."<<endl;
    else
        cout << "Bu bir rakam degildir."<<endl;

    if(islower(c))
        cout << "Bu bir kucuk harftir."<<endl;

    if(isupper(c))
        cout << "Bu bir buyuk harftir."<<endl;
}
```

Görev 2: Katarlar üzerinde farklı fonksiyonların kullanımı.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str1[] = "Bugun hava cok guzel.";
    char str2[] = "Piknige gidelim.";
    char str3[50];
    int  uzunluk;

    cout << "str1 katari: " << str1 << endl;
    uzunluk = strlen(str1);
    cout << "str1 katari uzunlugu: " << uzunluk << endl << endl;

    cout << "str2 katari: " << str2 << endl;
    uzunluk = strlen(str2);
    cout << "str2 katari uzunlugu: " << uzunluk << endl << endl;

    strcpy(str3, str1);
    cout << "str3 katari: " << str3 << endl << endl;

    strcat(str1, str2);
    cout << "s1 katari: " << str1 << endl;
```

```
    uzunluk = strlen(str1);
    cout << "s1 katari uzunlugu: " << uzunluk << endl << endl;

    if(strcmp(str1,str3)==0)
        cout << "Iki katar birbirine esittir" << endl;
    else
        cout << "Iki katar birbirine esit değildir." << endl;
}
```

B3. Kütüphanelerdeki Bazı Fonksiyonları Kullanarak Kodluyorum

Süre: 30 dk.

Kazanımlar: K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: L11.B3.1. Kütüphaneleri ve Fonksiyonları Kullanma: Görevler

Hazırlık: Materyal ÖYS üzerinden erişime açılır.

Uygulama: Eğitmenler C++ programlama dilindeki bazı kütüphane ve hazır fonksiyonları kullanıp uygulaması için hazırlanan görev etkinliğinde en az bir tanesini öğrencilerin seçmesini sağlayarak, öğrencilerin bu derste kodlama yapmasını ister. Diğer görevler ise öğrencilere kodlanması için ev görevi olarak verilebilir. Öğrenciler kodlama esnasında verilen görevler için ilk derste verilen katar ve karakter kütüphaneleri ilgili materyalleri destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmenlerden biri sınıfı dolaşarak, biri de kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Serkan yazdığı programda isim kısmına kullanıcının rakam girmesi durumunda programının "isminizde rakam olmaz" hatasını vermesini istiyor. Bunun için "isalpha" kodunda kullanan Serkan sizce nasıl bir kod yazmıştır?

Cevap 1:

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    char ad[50];
```

```
    cout << "Adinizi giriniz:";
    cin >> ad;

    for(int i=0; i<strlen(ad);i++)
    {
        if(!isalpha(ad[i]))
        {
            cout << "Isminizde rakam olamaz!";
            return 0;
        }
    }
    cout << "Merhaba " << ad;
    return 0;
}
```

Görev 2: Ahmet yazacağı bir programın şifresinin otomatik olarak rastgele belirlenecek sekiz rakamdan oluşmasını istemektedir. Bunun için nasıl bir kod yazmalıdır?

Cevap 2:

```
#include <iostream>
#include <cstring>
#include <ctime>

using namespace std;

int main()
{
    srand(time(0));

    int karakterSayisi = 8;
    char sifre[9];

  for(int i=0;i<karakterSayisi;i++)
    {
        char karakter;
        do
```

```
        {
            karakter = rand() % 255;
        }
        while(!isalnum(karakter));

        sifre[i] = karakter;
    }
    sifre[karakterSayisi] = '\0';
    cout << "Sifreniz: "<< sifre;
    return 0;
}
```

B4. Neden Dosyalama İşlemleri Yaparız?

Süre: 30 dk.

Kazanımlar: K3. Dosyalama işleminin gerekliliğini açıklar.

K4. Dosya kütüphanesi kullanarak program geliştirir.

Materyaller: L11.B4.1 Dosyalama İşlemleri Görev Yaprağı

L11.B4.2 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 1

L11.B4.3 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 2

L11.B4.4 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 3

L11.B4.5 Dosyalama İşlemleri

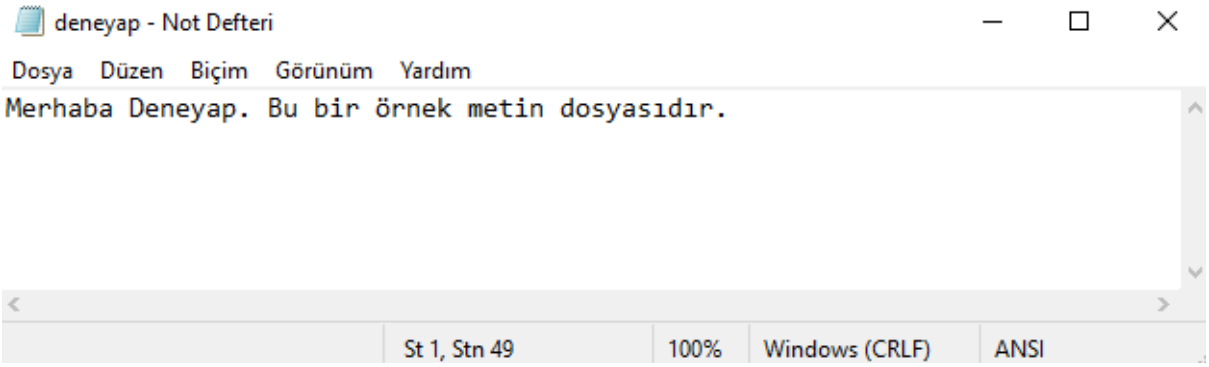
Hazırlık: Eğitmen birinci materyal hariç diğer tüm materyalleri ÖYS üzerinden erişime açar ve öğrencilerin göreceği şekilde projeksiyon ile yansıtır. Bu afiş öğrencilere destekleyici bilgi sağlamak için kullanılacaktır. Eğitmen dersin bu bölümü için hazırlanan materyal olan "L11.B4.1. Dosyalama İşlemleri Görev Yaprağı" adlı görev kâğıdını her gruba bir tane verecek şekilde 5 adet hâlinde çıktı alarak derse gelir.

Uygulama: Eğitmen öncelikle her bir öğrenciye "L11.B4.1. Dosyalama İşlemleri Görev Yaprağı" adlı görev kâğıdını dağıtır. Öğrencilerden bu ders için hazırlanan afişteki destekleyici bilgileri kullanarak bu görev kâğıdındaki boş yerleri doldurması istenir. Eğitmenlerin bu etkinlikteki görevi, öğrenci gruplarını sırayla gezerek onların ihtiyaç duydukları anda geri bildirimler, işlemsel bilgileri vererek öğrencileri desteklemesidir.

Eğitmene Öneriler: Afişlerde geçen bilgiler aşağıdaki gibi özetlenebilir. Öğrencilere verilen görevlerin cevaplarına ise bu bölümün sonunda ulaşılabilir.

DOSYALAMA

Programlama dilleri için dosya, verilerin kalıcı olarak saklanması için kullanılır. Dosya, program yardımıyla veya kullanıcılar tarafından oluşturularak depolama biriminde tutulur. Not defteri ile kolayca oluşturabiliriz.



Program verilerinin aksine, dosyadaki veriler bilgisayar kapansa bile silinmez. Bu sebeple ihtiyaç duyulan önemli bilgiler veya kullanıcılardan alınan bilgiler dosyalar yardımıyla tutulur. Genellikle "txt" uzantılı metin dosyaları kullanılır. "txt" uzantılı dosyalar hem kullanıcılar tarafından hemde program tarafından kolayca oluşturulabilir, okunabilir ve üzerine yazılabilir.

Var olan dosyalar üzerinde yapılacak metinsel işlemler okuma veya yazmadır. Dosya işlemleri ise, yeni dosya oluşturma ve mevcut dosyanın silinmesidir. Tüm bu işlemler C++ programlama dili ile hızlıca yapılmaktadır. Dosya işlemleri için C++ içerisinde üç temel sınıf bulunmaktadır.

ifstream	Okuma amaçlı açılacak dosya işlemleri için kullanılır.
ofstream	Yazma amaçlı açılacak dosya işlemleri için kullanılır.
fstream	Hem okuma hem de yazma amaçlı dosya işlemleri için kullanılır.

Dosyalar üzerinde yapılacak temel işlemler ve fonksiyonları ise aşağıdaki gibidir.

Dosyayı Aç	open()
Dosyadan veri oku	read()

Dosyaya veri yaz	write()
Dosyayı kapat	close()

Dosyayı açma sırasında eğer dosya mevcut değil ise, varsayılan olarak boş bir dosya oluşturulacaktır. Dosyayı farklı modlarda açabiliriz. Bu modlar;

Açıklama	mod
Normal dosya okuma modudur. Dosya en baştan okunmaya başlanır. Bu mod ifstream için varsayılan moddur.	ios::in
Normal dosya yazma modudur. Dosyaya en baştan yazılmaya başlanır. Bu mod ofstream için varsayılan moddur.	ios::out
Dosya yazma modudur. Dosyaya yazım işleminde, veriler dosyanın son karakterinden sonra eklenir.	ios::app
Dosya açıldığında içindeki tüm veriler silinir.	ios::trunc
Sadece dosya mevcut ise dosya açılacaktır. Eğer yoksa dosya oluşturulmayacaktır.	ios::nocreate

Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir.

Dosya Açma Kipi	Dosya açılma kontrolü kodu	Dosya mevcutsa ne olur?	Dosya yoksa ne olur?
ifstream sınıfının varsayılan modu okuma için olan ios::in modudur. ofstream sınıfının varsayılan modu yazma için olan ios::out modudur. fstream ise varsayılan modu ios::in ve ios::out modlarıdır.	Bir dosyanın açılıp/açılmadığı "is_open()" fonksiyonu ile kontrol edilir. Eğer hatalı bir durum olduysa "0" değeri döndürecektir. Bu şekilde dosyanın düzgün biçimde açıldığı kontrol edilebilir.	Dosya mevcut ise açma moduna göre içerik silinebilir. ios::out modunda dosya içeriği silinecektir. ios::app modunda ise dosya içeriği korunacaktır.	Eğer dosya mevcut değil ise belirtilen isimde yeni dosya oluşturulacaktır. ios::nocreate modunda ise yeni dosya oluşturulmayacak sadece dosya mevcut ise açılacaktır.
Örnek Kod: fstream dosya;	Örnek Kod: if(!dosya.is_open())		Örnek Kod: dosya.open("deneyap.txt", ios::nocreate) if(!dosya.is_open());

dosya.open("deneyap.txt" , ios::in | ios::out);	cout << "Dosya acilamadi!";		cout << "Dosya mevcut değil!";

B5. C++ Dilinde Dosyalama İşlemleri Yapıyorum

Süre: 20 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: L11. B4. 5. Dosyalama İşlemleri

L11. B5. 1. Kodlar Arasındaki Farkı Bulma

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan "L11. B4. 5. Dosyalama İşlemleri" adlı afişi ÖYS üzerinden erişime açar ve öğrencilerin göreceği şekilde projeksiyon ile yansıtır. Bu afiş öğrencilere destekleyici bilgi sağlamak için kullanılacaktır. Eğitmen dersin bu bölümü için hazırlanan diğer materyal olan "L11. B5. 1. Kodlar Arasındaki Farkı Bulma" adlı görev kâğıdını öğrencilerle her gruba bir tane olacak şekilde 5 adet çıktı alınır.

Uygulama: Eğitmenler sınıfı dörder kişilik beş gruba ayırır. Her grubun görebileceği şekilde bu ders için hazırlanan afişi sınıfın belirli bölgelerine asar ve afişi projeksiyon üzerinden yansıtır. Her gruba "L11. B5.1 Kodlar Arasındaki Farkı Bulma" adlı materyali dağıtmadan önce aşağıdaki gibi derse ister bilgisayar ister tahta üzerinden destekleyici bilgiyi sunarak başlar.

Destekleyici Bilgi:

C++ programlama dili üzerinde dosya işlemleri <fstream> kütüphanesi aracılığıyla yapılmaktadır. ifstream ve ofstream sınıfları bu amaçlar için kullanılmaktadır. Dosya okuma işlemleri için ifstream, dosyaya yazma işlemleri için ofstream kullanılır.

Boş dosya oluşturma:

ilk olarak kütüphaneyi projemize dahil ettik. Ardından dosya isimli bir nesne oluşturduk. Parantez içerisine de dosyanın ismini verdik. Eğer dosya yok ise projenin bulunduğu klasörde boş bir metin dosyası oluşturulmuş oldu.

```
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
```

```
    ofstream dosya("deneyap.txt");
}
```

İstersek yapıcı fonksiyonu kullanmadan dosya.open() fonksiyonu ile de dosyayı açabiliriz. Varolan dosyanın üstüne bilgi ekleyebiliriz bunun için dosya isminden sonra ikinci parametre olarak mod bilgisini (ios::app) vermemiz gerekmektedir.

Dosya içerisine yazı yazmak için;

```
   dosya << "Merhaba Deneyap!";
```

Satırını ekleyelim. Böylece dosyamızın içerisine “Merhaba Deneyap!” yazmış olduk. Son olarak dosyamızı kullanmayı bitirmek için;

```
dosya.close();
```

yazarak dosyamızı kapatıyoruz. Dosyamızı kapatarak geçiçi hafızayı da temizlemiş oluyoruz.

Bu bilgiler ışığında dağıttığım materyal üzerindeki kod farklılıklarını grup arkadaşlarımızla inceleyip, daha sonra hep beraber üzerinde tartışalım.

Eğitmene Öneriler: Eğitmenlere bu derse yönelik öneriler yukarıda (uygulama başlığı altında) verilmiştir.

B6. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

Süre: 30 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: L11.B4.2 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 1

L11.B4.3 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 2

L11.B4.4 C++ Programlama Dilinde Dosyalama İşlemleri Afişi 3

L11.B4.5 Dosyalama İşlemleri

L11. B6.1 Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

L11. B6. 2 Dosyalama İşlemleri ile İlgili Destekleyici Bilgi Sunusu

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan afişleri ÖYS üzerinden erişime açar. Bu afişler ve "**L11.B6.2** Dosyalama İşlemleri ile İlgili Destekleyici Bilgi Sunusu" öğrencilere destekleyici bilgi sağlamak için kullanılacaktır. Eğitmen dersin bu bölümü için hazırlanan diğer materyal olan "**L11. B6.1 Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum"** adlı görev kâğıdını her iki öğrenciye bir kopya olacak şekilde 10 adet çıktı alır. Her bilgisayarda 2 öğrenci oturması sağlanarak verilen görevleri iki kişilik grupların yapması sağlanır. Code Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Eğitmenler C++ programlama dilindeki dosyalama işlemleri için hazırlanan görev kâğıdındaki tüm etkinlikleri her öğrenci grubunun yapmasını ister. Öğrenciler kodlama esnasında verilen görevler için bir önceki derste hazırlanan dosyalama işlemleri ile ilgili afişleri ve "**L11.B6.2 Dosyalama İşlemleri ile İlgili Destekleyici Bilgi Sunusu**" destekleyici bilgi olarak kullanır. Sunu öğrencilere paylaşılır ve ÖYS üzerinden istedikleri görev sayfasındaki destekleyici bilgiyi görmesi sağlanır. Öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmenlerden biri sınıfı dolaşarak, biri de kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

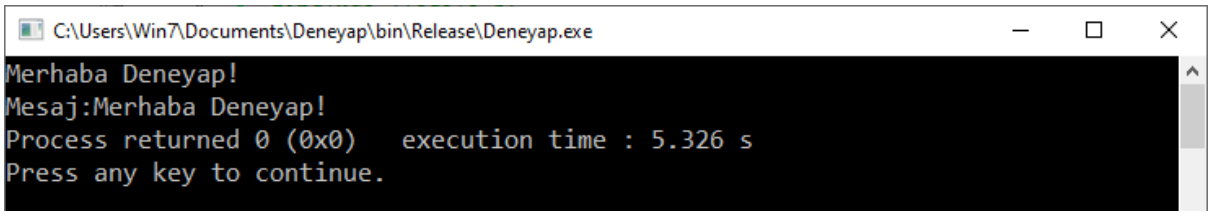
Dosyalama işlemleri ile ilgili verilen görevlerin cevapları aşağıdaki gibidir:

Görev 1: Klavyeden girilen "Merhaba Deneyap!" adlı cümleyi direkt string nesnesine aktarıp sonucu ekrana yazdıran kodu yazalım.

Cevap 1:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{

    string cumle = "Merhaba Deneyap!";
    cout << "Mesaj:" << cumle;
}
```

Klavyeden okuduğumuz cümleleri doğrudan string nesnesine aktarabiliriz. Bunun için aşağıdaki örneğe bakalım.



Resim 41. *Ekran çıktısı*

```
#include <iostream>
#include <fstream>
#include <string>
#include <locale.h>
using namespace std;
int main()
{
    string cumle;
    getline(cin,cumle);
    cout <<"Mesaj:" << cumle;
}
```

Cin komutu kullanıldığında hafızada tutulan önceki girişlerin temizlenmediği durumlarda sorun oluşabilmektedir. Bu yüzden cin.ignore() fonksiyonu kullanılarak giriş hafızası temizlenebilir.

Görev 2: C++ Programlama dili ile dosyalama kullanarak kendinize bir günlük oluşturun.

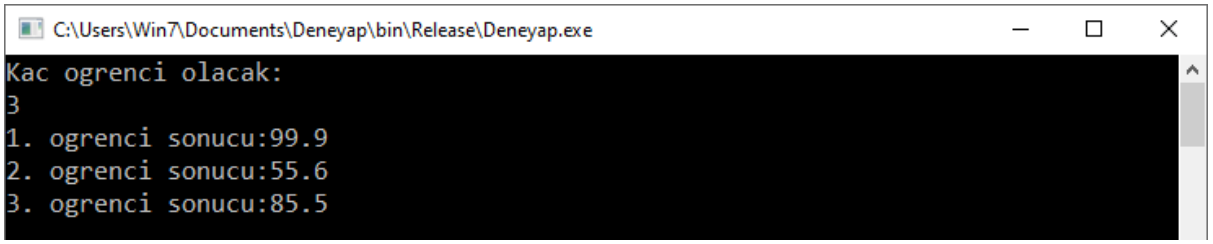
Cevap 2:

```
#include <iostream>
#include <fstream>
#include <string>
#include <locale.h>
using namespace std;
int main()
{
    setlocale(LC_ALL,"");
    int islem;

    cout << "1- Günlük Oku." << endl;
    cout << "2- Günlük Yaz." << endl;
    cin >> islem;

    if(islem==1)
    {
        ifstream dosya("gunluk.txt");
        if(!dosya.is_open())
        {
            cout << "Dosya Okunamadi!";
            return 0;
        }
        string satir;
        while(getline(dosya,satir))
            cout << satir <<endl;
        dosya.close();

    }
    else if(islem==2)
    {
        ofstream dosya("gunluk.txt",ios::app);
        if(!dosya.is_open())
        {
            cout << "Dosya Okunamadi!";
            return 0;
        }
```

```
        string satir;
        cout << "Ne yazmak istersin:\n";
        cin.ignore();
        getline(cin, satir);
        cout << "Kaydediliyor...";
        dosya << satir <<endl;
        dosya.close();
    }
}
```

Görev 3: Klavyeden girilen öğrenci sayısı kadar sınav notlarını klavyeden okuyup dosyaya yazdıran programı oluşturunuz.

Cevap 3:



Resim 42. Ekran çıktısı

```
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
    int ogrenciSayisi;
    float sonuc;

    ofstream dosya("sinav.txt");
    if(!dosya.is_open())
    {
        cout << "Dosya Okunamadi!";
        return 0;
    }
    cout << "Kac ogrenci olacak:" << endl;
    cin >> ogrenciSayisi;
```

```
    for(int i=0;i<ogrenciSayisi;i++)
    {

        cout << i+1 <<". ogrenci sonucu:";
        cin >> sonuc;
        dosya << sonuc << endl;
    }
    dosya.close();
}
```

C. Kısmi Öğrenme Görevleri

Süre: 50 dk.

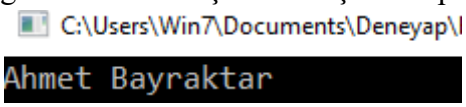
Materyal: L11. C.1. Kısmi Öğrenme Görevleri Afişi

Hazırlık: Kısmi öğrenme görevleri afişi öğrencilerin kolay ulaşabileceği noktalara asılır ya da öğrencilerin Öğrenme Yönetim Sistemi ortamında materyal olarak erişmeleri sağlanır.

Uygulama: Öğrenciler afişteki görevleri istedikleri sırada ve sayıda kendi tercihlerine bağlı olarak verilen süre içinde uygulamaya başlar. Her bir görevi tamamlayan öğrencilere, göreve ilişkin beceri rozeti verilir. Öğrenciler görev seçimlerini ve tamamladıkları görevleri eğitmene bildirmelidir. Eğitmenler ihtiyaç duydukları zaman öğrencilere görevi tamamlama aşamasında anlık geri bildirimlerde bulunur. Öğrenciler, ÖYS üzerinden süreli ödev olarak açılan kısmi öğrenme görevlerinden kendi tercihlerine göre istedikleri sayıda görev tamamlayıp eğitmene iletir. Süre bitiminde eğitmen görevleri ve yanıtlarını ÖYS ya da GitHub üzerinden öğrencilere gönderir.

Kısmi öğrenme görevinin ismi beceri rozetini tanımlamaktadır. Beceri rozetine ait görevlerin yanıtları ise aşağıda verilmektedir.

1) **Kodlayıcı** - "Ahmet BAYRaKtAR" şeklinde girilen ad soyad için, soyadının sadece ilk harfini büyük harfe diğer harflerini küçük harfe çeviren programı yazalım.



Resim 43. *Ekran çıktısı*

Cevap 1:

```
#include <iostream>
#include <cstring>

using namespace std;
int main()
{
```

```
    char mesaj[] = "Ahmet BAYRaKtAR";
    int kelimeSayisi = 0;
    bool bosluk = false;
    for(int i=0; i<strlen(mesaj);i++)
    {
        if(bosluk)
        {
            mesaj[i] = tolower(mesaj[i]);
        }
        if(mesaj[i] == ' ')
        {
            bosluk = true;
            mesaj[i+1] = toupper(mesaj[i+1]);
            i++;
        }

    }
    cout << mesaj ;
    return 0;
}
```

2) **Kodlayıcı** - Parametre olarak gönderilen katar içerisinde parametre olarak gönderilen karakteri sayan fonksiyonu yazınız.

Cevap 2:

```
#include <iostream>
#include <cstring>

using namespace std;
int karakter_say(char metin[], char karakter)
{
    int sayac = 0;
    for(int i=0;i<strlen(metin);i++)
    {
        if(tolower(metin[i])==karakter)
            sayac++;
    }
    return sayac;
```

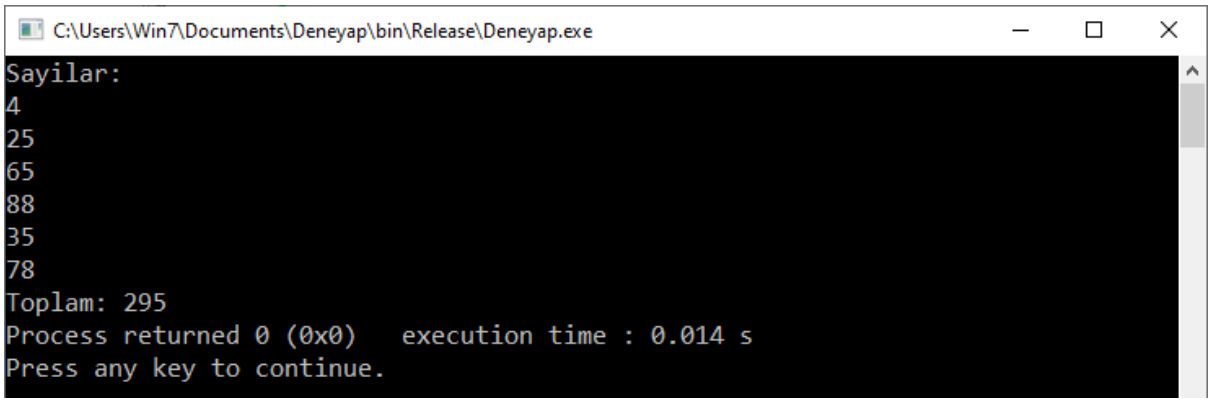
```
}
int main()
{
    char mesaj[] = "Bugun hava cok guzel. Azicik disari cikalim.";
    char karakter = tolower('B');

    cout << "Bu cumlede "<< karakter_say(mesaj,karakter) <<" adet "<< karakter
<<" vardir.";
    return 0;
}
```

3) Kodlayıcı - Dosyadan okuduğunuz sayıların toplamını ekrana yazınız.

Cevap 3:

```
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <string>
using namespace std;
int main()
{
    int toplam = 0;
    ifstream dosya("sayilar.txt");
    string satir;
    if(dosya.is_open())
    {
        cout << "Sayilar:"<<endl;

        while(getline(dosya,satir))
        {
            toplam += atoi(satir.c_str());
        cout << satir << endl;
        }
        cout << "Toplam: "<< toplam;
    }
    else
    {
        cout << "dosya Okunamadi!";
```

```
        }
    }
```



Resim 44. *Ekran çıktısı*

Hafta 11. Ders Materyalleri

L11.B1.1 C++ Programlama Dilinde Kütüphaneler

UYARI:

C++ programlama dili içerisinde geliştiricilerin kullanımına sunulmuş birçok fonksiyonun kütüphaneler içerisinde yer aldığını biliyor muydunuz?

Örneğin: Bu haftaya kadar kullandığımız ***cout*** ve ***cin*** fonksiyonları ***iostream*** isimli kütüphane içerisinde bulunmaktadır.

Şimdi gelin bu derste C++ programlama dilinde kullanabileceğimiz diğer kütüphaneleri ve bu kütüphanelerdeki bazı fonksiyonları öğrenelim.

L11.B1.2 Karakter Kütüphanesi

Tek bir karakter için hazırlanmış fonksiyonlar cctype kütüphanesi içerisinde bulunur. Standart kütüphane olarak projemizde bulunmaktadır.

“

#include<cctype>

satırı ile projemize dahil ederiz.

”

Lütfen şimdi aşağıda yer alan noktalı yerleri grup arkadaşlarımızla dolduralım.

Fonksiyon	İşlevi	Örnek Kullanım	Sonuç
isalpha(c)	c karakteri eğer bir harf ise geriye true değilse false döndürür.	isalpha(11.4)	…..
isdigit(c)	c karakteri eğer bir rakam ise geriye true değilse false döndürür.	isdigit(11.4)	…..
isalnum(c)	c karakteri eğer bir rakam veya harf ise geriye true değilse false döndürür.	isalnum(/*)	…..
islower(c)	c karakteri eğer bir küçük harf ise geriye true değilse false döndürür.	islower (d)	…..
isupper(c)	c karakteri eğer bir büyük harf ise geriye true değilse false döndürür.	isupper(H)	…..
tolower(c)	c karakterini küçük harfe çevirir.	tolower(E)	…..
toupper(c)	c karakterini büyük harfe çevirir.	toupper(g)	…..

L11.B1.3 Metin Kütüphanesi

C++ programlama dilinde metinler üzerinde işlem yapan kullanıma hazır fonksiyonları içerisinde barındıran kütüphanenin adı **cstring** kütüphanesidir.

“

#include<cstring>

satırı ile projemize dahil ederiz.

”

Lütfen şimdi aşağıda yer alan noktalı yerleri grup arkadaşlarımızla dolduralım.

Fonksiyon	İşlevi	Örnek Kullanım	Sonuç
strlen (s1)	s1 katarının uzunluğunu döndürür.	s1 = “Merhaba” strlen (s1)	……..
strcpy (s1, s2)	s2 katarını s1 katarına kopyalar.	s1= “Merhaba” s2= “Dünya” strcpy (s1, s2)	……..
strcat (s1, s2)	s2 katarını s1 katarının sonuna ekler.	s1= “Merhaba” s2= “Dünya” strcat (s1, s2)	……..
strcmp (s1, s2)	s1 ve s2 aynı ise 0 değerini döndürür; s1 < s2 ise 0'dan küçük değer döndürür; s1 > s2 ise 0'dan büyük değer döndürür.	s1= “Merhaba” s2= “Dünya” strcmp (s1, s2)	……..

L11.B2.1 C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

Kod	Kodun Çıktısı:
1)…….. 2)…….. `using namespace std;` `int main()` `{` `char c='a';` `if(3.....(c))` `cout << "Bu bir harftir."<<endl;` `else` `cout << "Bu bir harf degildir."<<endl;` `if(4.....(c))` `cout << "Bu bir rakamdir."<<endl;` `else` `cout << "Bu bir rakam degildir."<<endl;` `if(5.....(c))` `cout << "Bu bir kucuk harftir."<<endl;` `if(6.....(c))` `cout << "Bu bir buyuk harftir."<<endl;` `return 0;` `}`	Bu bir harftir. Bu bir rakam değildir. Bu bir küçük harftir.
`#include <iostream>` 1)…… `using namespace std;` `int main()` `{`	:

```
    char str1[] = "Bugun hava cok guzel.";

    char str2[] = "Piknige gidelim.";

    char str3[50];

    int  uzunluk;


    cout << "str1 katari: " << str1 << endl;

    uzunluk = 2.....(str1);

    cout << "str1 katari uzunlugu: " << uzunluk << endl <<
endl;


    cout << "str2 katari: " << str2 << endl;

    uzunluk = strlen(str2);

    cout << "str2 katari uzunlugu: " << uzunluk << endl <<
endl;


    3.....(str3, str1);

    cout << "str3 katari: " << str3 << endl << endl;


    4.....(str1, str2);

    cout << "s1 katari: " << str1 << endl;


    uzunluk = 5.....(str1);

    cout << "s1 katari uzunlugu: " << uzunluk << endl <<
endl;


    if(6.....(str1,str3)==0)

        cout << "Iki katar birbirine esittir" << endl;

    else

        cout << "Iki katar birbirine esit değildir." <<
endl;

    return 0;

}
```

```
str1 katari: Bugun hava cok guzel.

str1 katari uzunlugu: 21


str2 katari: Piknige gidelim.

str2 katari uzunlugu: 16


str3 katari: Bugun hava cok guzel.


s1 katari: Bugun hava cok guzel.Piknige gidelim.

s1 katari uzunlugu: 37


Iki katar birbirine esit değildir.
```

L11.B3.1 C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

Görev 1

Serkan yazdığı programda isim kısmına kullanıcının rakam girmesi durumunda programının "isminiz-de rakam olmaz" hatasını vermesini istiyor. Bunun için "isalpha" kodunda kullanan Serkan sizce nasıl bir kod yazmıştır?

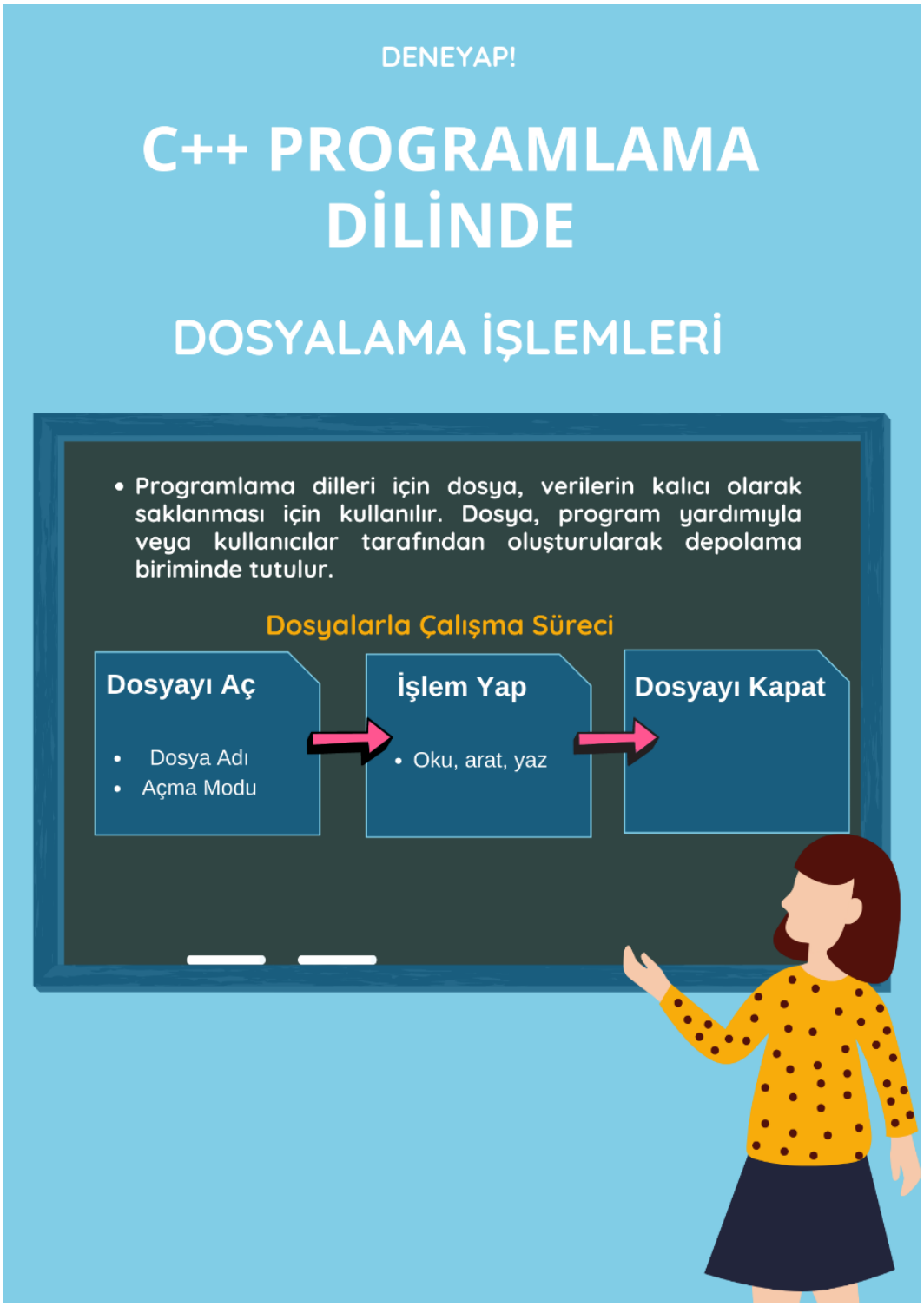
Görev 2

Ahmet yazacağı bir programın şifresinin otomatik olarak rastgele belirlenecek sekiz rakamdan oluşmasını istemektedir. Bunun için nasıl bir kod yazmalıdır?

L11.B4.1 Dosyalama İşlemleri Görev Yaprağı

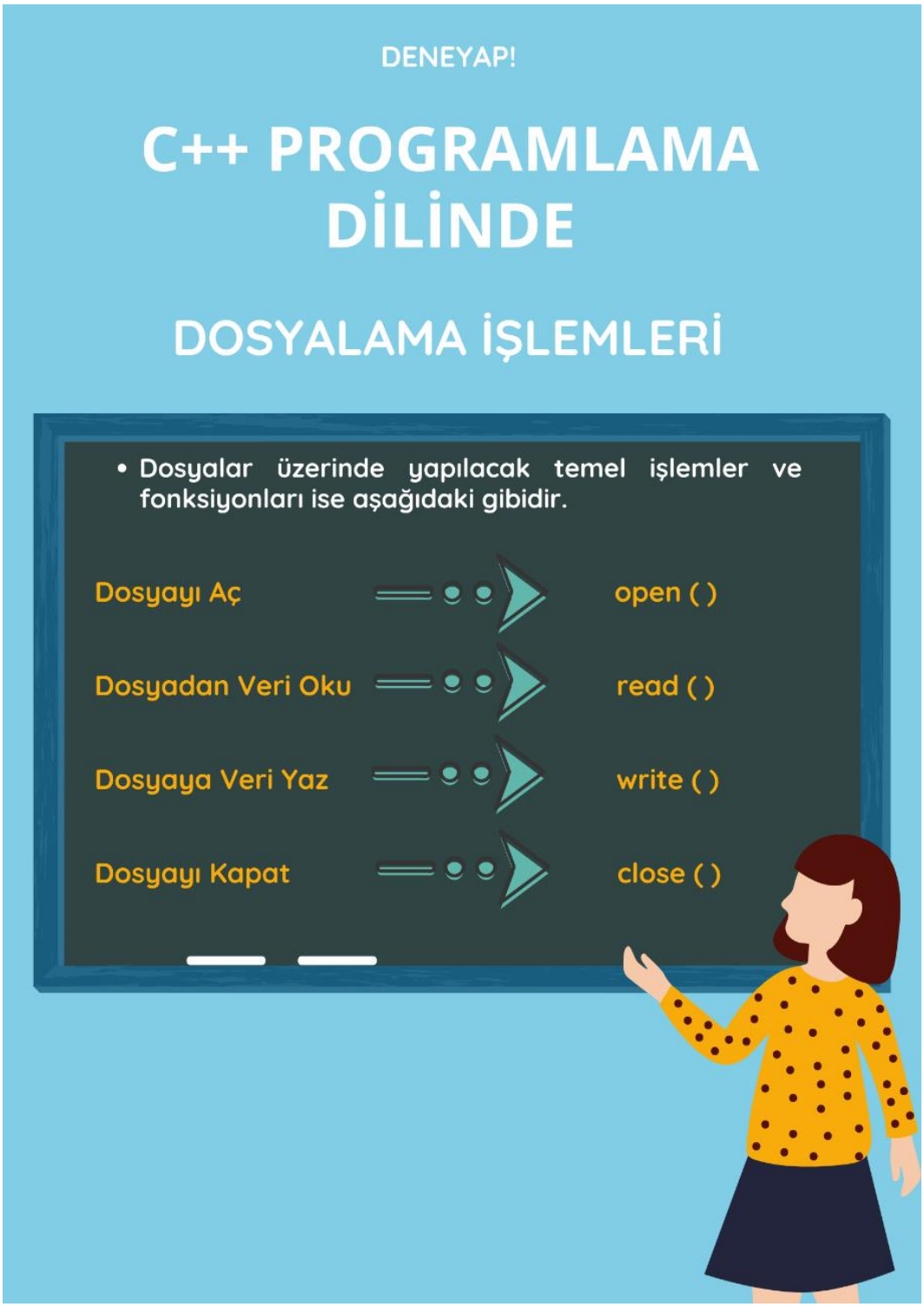
Dosya Açma Kipi	Dosya açılma kontrolü kodu	Dosya mevcutsa ne olur ?		Dosya yoksa ne olur?
ifstream sınıfının varsayılan modu okuma için olan …….. modudur. ofstream sınıfının varsayılan modu yazma için olan …… modudur. fstream ise varsayılan modu …… ve ……. modlarıdır.	Bir dosyanın açılıp/açılmadığı …….. fonksiyonu ile kontrol edilir. Eğer hatalı bir durum olduysa “0” değeri döndürecektir. Bu şekilde dosyanın düzgün biçimde açıldığı kontrol edilebilir.	Dosya mevcut ise açma moduna göre içerik silinebilir. …… modunda dosya içeriği silinecektir. ……. modunda ise dosya içeriği korunacaktır.		Eğer dosya mevcut değil ise belirtilen isimde yeni dosya oluşturulacaktır. …….. modunda ise yeni dosya oluşturulmayacak sadece dosya mevcut ise açılacaktır.
Örnek Kod: fstream dosya; dosya.open(“deneyap.txt”, ios::in | ios::out)	Örnek Kod: if(!dosya.is_open()) cout << “Dosya acilamadi!”;			Örnek Kod: dosya.open(“deneyap.txt”, ios::nocreate) if(!dosya.is_open()) cout << “Dosya mevcut değil!”;

L11.B4.2 C++ Programlama Dilinde Dosyalama İşlemleri Yapıyorum Afişi 1



Resim 45. *Dosyalama işlemleri afişi 1*

L11.B4.3 C++ Programlama Dilinde Dosyalama İşlemleri Yapıyorum Afişi 2

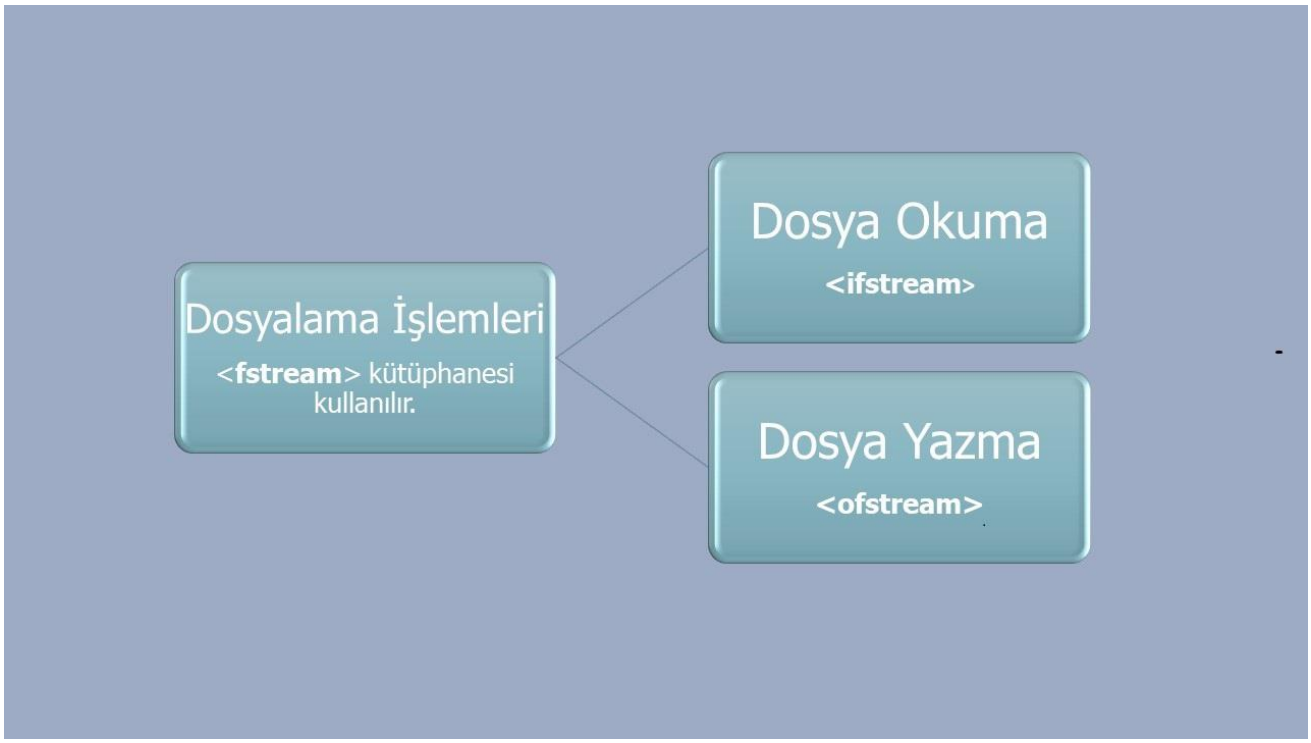


Resim 46. *Dosyalama işlemleri afişi 2*

L11.B4.4 C++ Programlama Dilinde Dosyalama İşlemleri Yapıyorum Afişi 3

Resim 47. *Dosyalama işlemleri afişi 3*

L11.B4.5 Dosyalama İşlemleri



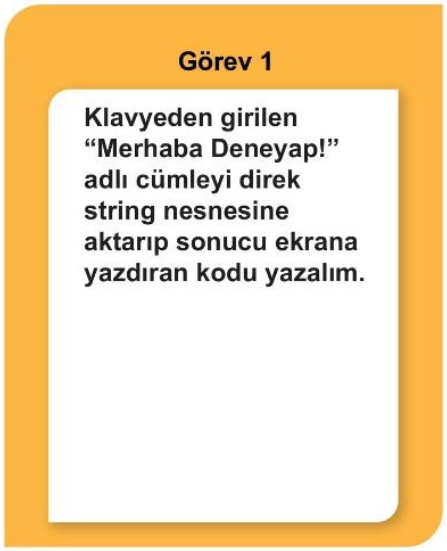
Resim 48. *Dosyalama işlemleri*

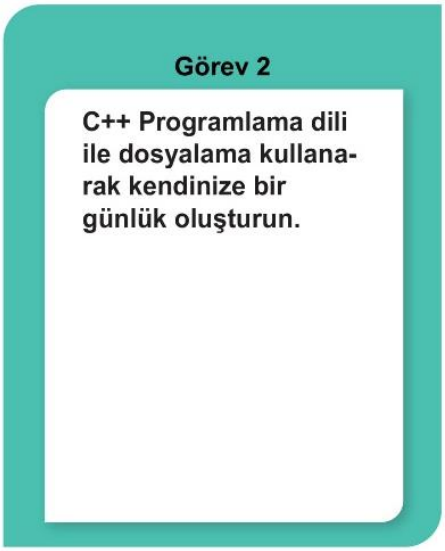
L11. B5.1 Kodlar Arasındaki Farkı Bulma

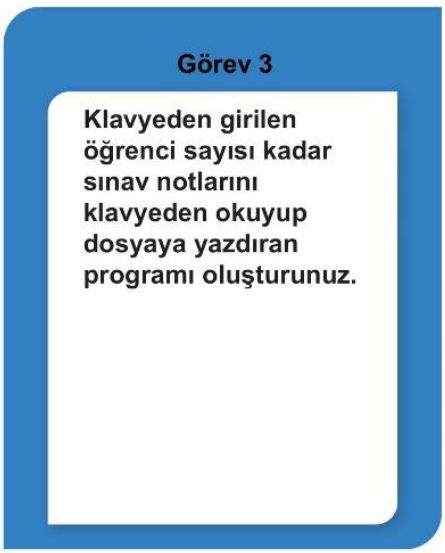
1. İşlem	2. İşlem
`#include <iostream>` `#include <fstream>` `using namespace std;` `int main()` `{` `ofstream dosya;` `dosya.open("deneyap.txt");` `dosya << "Merhaba Deneyap!" << endl;` `dosya.close();` `}`	`#include <iostream>` `#include <fstream>` `using namespace std;` `int main()` `{` `ofstream dosya;` `dosya.open("deneyap.txt",ios::app);` `dosya << "Merhaba Deneyap!" << endl;` `dosya.close();` `}`

!!! Yukarıda verilen kodları dikkatlice gözlemleyerek aralarında ne gibi bir fark olduğunu kodları bilgisayarda yazarak bulmaya çalışalım.

L11. B6.1 Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum







L11. C. 1. Kısmi Öğrenme Görevleri Afişi

Kısmi öğrenme görevleri afişine ulaşmak için buraya tıklayınız.

Hafta 12. Proje Yarışması

Amaç

Bu haftanın amacı öğrencilerin yazılım teknolojileri dersi kapsamında öğrendiği bilgileri kullanarak yarışma içerisinde verilen soruları cevaplandırmasıdır.

Yarışma Öncesi Öğrenci Gruplarının Belirlenmesi

Yarışma için öğrenci gruplarının oluşturulmasında öğrencilerin eğitim boyunca kazandıkları beceri rozetleri dikkate alınmalıdır. Buradaki amacımız 4 farklı beceri rozetine farklı sayıda sahip olan öğrencilerin gruplara dengeli olarak dağıtılmasıdır.

Rozetler:

1. Analizci Rozeti (AR): Verilen problem için üretilen çözümlerin uygunluğunu kontrol eder ve varsa mantık hatalarının giderilmesini sağlar.

2. Tasarlayıcı Rozeti (TR): Probleme uygun çözümlerin uygulamaya geçebilmesi için kodlanmasını sağlar.

3. Kodlayıcı Rozeti (KR): Verilen probleme uygun çözümün nasıl olabileceği ile ilgili ön hazırlıkları yaparak gerekli algoritma ve akış diyagramlarının hazırlanmasını sağlar.

4. Denetleyici Rozeti (DR): Verilen problem için üretilen kodlamaların uygunluğunu kontrol eder ve varsa derleyici hatalarının giderilmesini sağlar.

Grupların belirlenmesi için aşağıdaki 5 adım izlenmelidir.

1. Listedeki tüm öğrenciler Kodlayıcı Rozeti (KR) sayılarına göre en büyükten en küçüğe doğru sıralanır ve ilk beş öğrencinin ismi KR1, KR2, KR3, KR4, KR5 olarak etiketlenir.

2. Listedeki kalan öğrenciler Tasarlayıcı Rozeti (TR) sayılarına göre en büyükten en küçüğe doğru sıralanır ve ilk beş öğrencinin ismi TR1, TR2, TR3, TR4, TR5 olarak etiketlenir.

3. Listedeki kalan öğrenciler Analizci Rozeti (AR) sayılarına göre en büyükten en küçüğe doğru sıralanır ve ilk beş öğrencinin ismi AR1, AR2, AR3, AR4, AR5 olarak etiketlenir.

4. Listedeki kalan öğrenciler Denetleyici Rozeti (DR) sayılarına göre en büyükten en küçüğe doğru sıralanır ve ilk beş öğrencinin ismi DR1, DR2, DR3, DR4, DR5 olarak etiketlenir.

5. Öğrenciler verilen etiket bilgileri dikkate alınarak aşağıdaki tabloya yerleştirilir.

Tablo 26. *Öğrenci grupları oluşturulması*

Grup 1	Grup 2	Grup 3	Grup 4	Grup 5
KR1	KR2	KR3	KR4	KR5
TR5	TR4	TR3	TR2	TR1

Grup 1	Grup 2	Grup 3	Grup 4	Grup 5
AR2	AR1	AR3	AR5	AR4
DR4	DR5	DR3	DR1	DR2

Yarışmadaki Görevlerin Değerlendirilmesi

Jüri değerlendirme formu için aşağıdaki linke tıklayınız:

https://forms.gle/eW2GtFtdsztxsyhT7

- Her bir problem için verilmesi gereken süre 45 dakikadır. 4 problem için toplam süre ise 180 dakika olmaktadır.
- Problem çözümleri ekip tarafından daha kısa sürede teslim edilebilir.
- Problem çözümleri için kullanılan süre kayıt altına alınmalıdır.
- Problem çözümleri teslim edildikten sonra çözüm üzerinde bir değişiklik yapılamaz.
- Ekipler yeni problemi çözerken jüri teslim edilen problem çözümünü cevap kâğıdına uygun olarak değerlendirerek 0-100 arası puanı oluşturur.
- Dört problemin çözümünü tamamlayan gruplar istediği taktirde zamanları kalmışsa (henüz 4 soruda 180 dakika kullanılmamışsa) bu kalan zamanda Bonus problemi de çözebilirler.
- Bonus problemi için verilmesi gereken süre 30 dakikadır.
- Her bir ipucu kullanımında ekip 10 puan kaybedecektir.

Puan hesaplaması aşağıdaki gibi yapılacaktır.

$$GP = P1 + P2 + P3 + P4 + PB – (İP * 10) + TS – (S1 + S2 + S3 + S4 + SB)$$

GP: Genel Puan

P1 = Problem 1 Çözümünden Alınan Puan (Min:0 Max:100)

P2 = Problem 2 Çözümünden Alınan Puan (Min:0 Max:100)

P3 = Problem 3 Çözümünden Alınan Puan (Min:0 Max:100)

P4 = Problem 4 Çözümünden Alınan Puan (Min:0 Max:100)

PB = Bonus Probleminin Çözümünden Alınan Puan (Min:0 Max:50)

İP = Tüm Sorularda Kullanılan Toplam İpucu Sayısı (Min:0 Max:80)

TS = 180 (Toplam Süre)

S1 = Problem 1 Çözümünde Harcanan Süre (Min:0 Max:45)

S2 = Problem 2 Çözümünde Harcanan Süre (Min:0 Max:45)

S3 = Problem 3 Çözümünde Harcanan Süre (Min:0 Max:45)

S4 = Problem 4 Çözümünde Harcanan Süre (Min:0 Max:45)

SB = Bonus Probleminin Çözümünde Harcanan Süre (Min:0 Max:30)

HAZİNE AVI

Maceracı arkadaşlar hazine arayışındadır. Bir gün çok eski bir harita bulurlar. Haritada etrafı sularla kaplı dört küçük hazine adasından bahsediliyor. Ancak bu adalarda hazine avlamak o kadar da kolay değil. Maceracıların hazineye ulaşabilmeleri için adalara giriş şifrelerini bulmaları gerekiyor. Adalar arasında ilerledikçe çeşitli problemlerle karşılaşırlar. Maceracılar bu problemleri çözdükçe adalara giriş şifresini de çözmüş olacaklar. Maceracıların her soruda gerekli kod görevlerini yaparak bir sonraki adıma geçmeleri gerekmektedir. Bununla birlikte herhangi bir sorunun koduna ilişkin çözüm tamamlanmasa da adalarda ilerleme için gerekli işlemler elle tamamlanabilir. Fakat bu durumda ilgili sorudan herhangi bir puan alınmayacaktır. Elle çözüm yapılabilmesinin amacı maceracıların hazine avını yarıda bırakmasını engellemektir.


HAZİNE AVI
Maceracı arkadaşlar hazine arayışındadır. Bir gün çok eski bir harita bulurlar. Haritada etrafı sularla kaplı dört küçük hazine adalarından bahsediliyor. Ancak bu adalarda hazine avlamak o kadar da kolay değil. Maceracılar hazineye ulaşabilmeleri için adalara giriş şifrelerini bulmaları gerekiyor. Adalar arasında ilerledikçe çeşitli problemlerle karşılaşırlar. Maceracılar bu problemleri çözdükçe adalara giriş şifresini de çözmüş olacaklar.

TEKNE TAMİRİ

Maceracıların haritadaki ilk küçük adaya ulaşmaları için bir tekne bulmaları gerekiyor. Maceracıların haritadaki ilk durağı Kaşık Adası. Kaşık Adası'na ulaşmaları için bir tekne bulmaları gerekiyor. Ellerindeki tek teknenin ise tadilata ihtiyacı var.

Tadilat için 10 tane kısa uzunlukta ince tahta, 7 tane orta uzunlukta kalın tahta, 4 tane orta uzunlukta ince tahta ve 7 tane uzun kalın tahta kullanacaklar. Bunun için ödemeleri gereken ücreti tahta fiyat listesinden hesaplamaları lazım. İşte maceralarımız ilk problemleriyle karşılaşmıştır. Bunun için bir akış diyagramı oluşturmaya karar verirler. Bu akış diyagramında tahtaların adedi, boyu ve kalınlığına ilişkin bilgilerin kullanıcı tarafından girilmesi isteniyor.

Maceracılar akış diyagramının sonunda ihtiyaç listesine ödenecek toplam değeri hesaplayınca, Kaşık Adası'na giriş şifresini çözmüş olacaklar. Bu görevde maceracılara yardım ediniz.

Tablo 27. *Tekne tamiri tablosu*

Tahta Boyu	Tahta Kalınlığı	Tahta Fiyatı	İhtiyaç Listesi
1- Kısa	1- Kalın	42 TL	10 Adet Kısa ve İnce Tahta 7 Adet Orta ve Kalın Tahta 4 Adet Orta ve İnce Tahta 7 Adet Uzun ve Kalın Tahta
	2- İnce	34 TL	
2- Orta	1- Kalın	83 TL	
	2- İnce	61 TL	
3- Uzun	1- Kalın	93 TL	
	2- İnce	79 TL	

1. İpucu: Problemin çözümü için döngü yapısı kullanabilirsiniz.

2. İpucu: İhtiyaç listesi farklı sayıda olabilir. Bu yüzden tüm durumları kontrol etmelisiniz.


TEKNE TAMİRİ
Maceracıların haritadaki ilk durağı Kaşık adası. Kaşık adasına ulaşmaları için bir tekne bulmaları gerekiyor. Ellerindeki tek teknenin ise tadilata ihtiyacı var.
Tadilat için 10 tane kısa uzunlukta ince tahta, 7 tane orta uzunlukta kalın, 4 tane orta uzunlukta ince ve 7 tane uzun kalın tahta kullanacaklar. Bunun için ödemeleri gereken ücreti tahta fiyat listesinden hesaplamaları lazım. İşte maceralarımız ilk problemleriyle karşılaşmıştır. Bunun için bir akış diyagramı oluşturmaya karar verirler. Bu akış diyagramında tahtanın adedi, boyu ve kalınlığına ilişkin bilgilerin kullanıcı tarafından girilmesi isteniyor.
Maceracılarımız akış diyagramının sonunda ihtiyaç listesine ödenecek toplam değeri hesaplayınca, Kaşık adasına giriş şifresini çözmüş olacaklar.
BU GÖREVDE MACERACILARA YARDIM EDİNİZ.
TAHTA FİYAT LİSTESİ
BOY KALINLIK FİYATI
KISA KALIN 42 YTL
KISA İNCE 34 YTL
ORTA KALIN 83 YTL
ORTA İNCE 61 YTL
UZUN KALIN 93 YTL
UZUN İNCE 79 YTL

XOX OYNA

Kaşık Adası'na hoş geldiniz.

İkinci sırada Balıkçı Adası vardır. Balıkçı Adası'nın şifresini kazanmak için maceracılarımız ada yerlilerine XOX oyunu yazmak için yardım etmelidir.

XOX oyunu iki kişi ile oynanan 3x3 bir tahta oyunudur. Bir kişi tahtaya 'X' koyduğunda rakibi 'O' koymaktadır. Oyunun amacı 3 adet 'X' ve 'O' harflerini yan yana, üst üste veya çapraz olarak yerleştirmektir. Maceracılar XOX tahtasının durumuna göre bir oyunun bitip bitmediğini kontrol eden programı fonksiyon kullanarak yazarlar. Oyunun bitmesi durumunda; eğer oyunu X oyuncusu kazandı ise ekrana XXX, oyunu O oyuncusu kazandı ise ekrana OOO, oyun bitmediğinde ise ekrana XOX yazdırılması gerekmektedir.

Balıkçı Adası'nın şifresi için maceracılar yazdıkları programda aşağıdaki XOX tahtasını test eder. Karşılaştıkları program çıktısı Balıkçı Adası'nın şifresidir.

Anahtar: Örnek olarak aşağıdaki tahta için test edildiğinde çıktı ne olacaktır?

X	O	X
O	X	O
O	X	X

1. İpucu: Tahtayı char tahta[3][3] şeklinde tanımlayabilirsiniz.

2. İpucu: Satır ve sütun kontrolü yaparken 1 döngü ve 1 if koşulu yeterli olacaktır.


XOX OYNA
Kaşık adasına hoşgeldiniz.
İkinci sırada Balıkçı adası vardır. Balıkçı adasının şifresini kazanmak için maceracılarımız ada yerlilerine XOX oyunu yazmak için yardım etmelidir. XOX oyunu iki kişi ile oynanan 3x3 bir tahta oyunudur. Bir kişi tahtaya 'X' koyduğunda rakibi 'O' koymaktadır. Oyunun amacı 3 adet 'X' ve 'O' harflerini yan yana, üst üste veya çapraz olarak yerleştirmektir. Maceracılar XOX tahtasının durumuna göre bir oyunun bitip bitmediğini kontrol eden programı fonksiyon kullanarak yazarlar. Oyun bitmesi durumunda eğer X oyuncusu kazandı ise ekrana XXX, eğer O oyuncusu kazandı ise OOO, bitmediğinde ise XOX yazdırılması gerekmektedir.
Balıkçı adasının şifresi için maceracılar yazdıkları programda aşağıdaki XOX tahtasını test eder. Karşılaştıkları program çıktısı Balıkçı adasının şifresidir.
BU GÖREVDE MACERACILARA YARDIM EDİNİZ.
XOX TAHTASI
X O X
O X O
O X X

BALIKÇI ADASI

Balıkçı Adası'na hoş geldiniz. Üçüncü adada ekipler yemek için bir mola vermiştir. Menüde tabloda detayları verilen 3 farklı balık türü bulunmaktadır. Balık sınıfı oluşturarak balık türü ve ağırlık bilgilerini sınıf üyeleri olarak belirleyin. Balık sınıfından farklı türlerin sınıfını türetme yoluyla oluşturunuz.

Eğer seçilen balık Levrek ise TL fiyatı kilogram olarak ağırlığının 40 katı, Palamut ise 25 katı, Çupra ise 30 katı olmaktadır. Her sınıf içerisinde yazılacak bir "hesapla" fonksiyonu ile ana fonksiyondan gönderilen adet bilgisine göre ilgili balık için ödenmesi gereken ücret hesaplanmak istenmektedir. Daha sonra tüm balıklar için ödenmesi gereken hesap bulunarak oluşturulacak bir "hesap.txt" dosyasında kaydedilmek isteniyor. Dosyaya yazılacak bu değer bir sonraki soruya geçiş şifreniz olacaktır. Toplam 6 adet levrek, 4 adet palamut ve 8 adet çupra yenildiğine göre ödenmesi gereken toplam hesap ne kadar olacaktır?

Tablo 28. *Balık bilgisi tablosu*

Balık Türü	Fiyat
Levrek	450 gr
Palamut	600 gr
Çupra	500 gr

1. İpucu: Balık fiyatı hesaplamada örneğin Levrek için; adet * (agirlik * 40) hesaplaması yapılmalıdır. Buradaki agirlik değişkeni float olarak tanımlanmalı ve 40 ile çarpımı parantez içerisine alınmalıdır. Aksi takdirde elde edeceğiniz sonuç int-float çarpımı nedeniyle farklı olacaktır.

2. İpucu: Türetilen fonksiyonlarda yazacağınız hesapla fonksiyonu geriye int değer döndüren int parametre alan bir fonksiyon olarak tasarlanmalıdır.


BALIK YEME
Balıkçı adasına hoşgeldiniz.
Üçüncü adada ekipler yemek için bir mola vermiştir. Menüde tabloda detayları verilen 3 farklı balık türü bulunmaktadır. Balık sınıfı oluşturarak balık türü ve ağırlık bilgilerini sınıf üyeleri olarak belirleyin. Balık sınıfından farklı türlerin sınıfını türetme yoluyla oluşturunuz. Eğer seçilen balık Levrek ise TL fiyatı kilogram olarak ağırlığının 40 katı, Palamut ise 25 katı, Çupra ise 30 katı olmaktadır. Her sınıf içerisinde yazılacak bir "hesapla" fonksiyonu ile ana fonksiyondan gönderilen adet bilgisine göre ilgili balık için ödenmesi gereken ücret hesaplanmak istenmektedir. Daha sonra tüm balıklar için ödenmesi gereken hesap bulunarak oluşturulacak bir "hesap.txt" dosyasında kaydedilmek isteniyor. Dosyaya yazılacak bu değer bir sonraki soruya geçiş şifreniz olacaktır.
Toplam 6 adet levrek, 4 adet palamut ve 8 adet çupra yenildiğine göre toplam ödenmesi gereken hesap ne kadar olacaktır?
BU GÖREVDE MACERACILARA YARDIM EDİNİZ.
BALIK LİSTESİ
BALIK AĞIRLIK
LEVREK 450 GR
PALAMUT 600 GR
ÇUPRA 500 GR

HAZİNE ADASI

Hazine Adası'na hoş geldiniz.

Maceracılar, hazine sandığının anahtarını açmak için aşağıdaki 7x7'lik matristen oluşan bir matematik problemi ile karşılaşır. Bu matris 0-9 arası rastgele dağılmış rakamlardan oluşmaktadır.

Maceracılar, şifreyi çözmek için bu rakamların kullanım miktarını (sayısını) en azdan çoğa doğru sıralayan kodu yazar. Çünkü kodun çıktısı hazine sandığının şifresidir.

3	1	7	6	3	9	5
2	8	2	7	9	8	6
7	5	9	0	7	2	2
3	1	3	5	6	1	5
8	0	5	7	2	9	8
5	2	8	3	7	0	3
7	6	2	9	6	1	6

1. İpucu: İç içe döngü yapısı kullanılabilir.

2. İpucu: Farklı rakam sayısı boyutunda bir dizi oluşturup tekrar sayılarını bu dizide tutabilirsiniz.

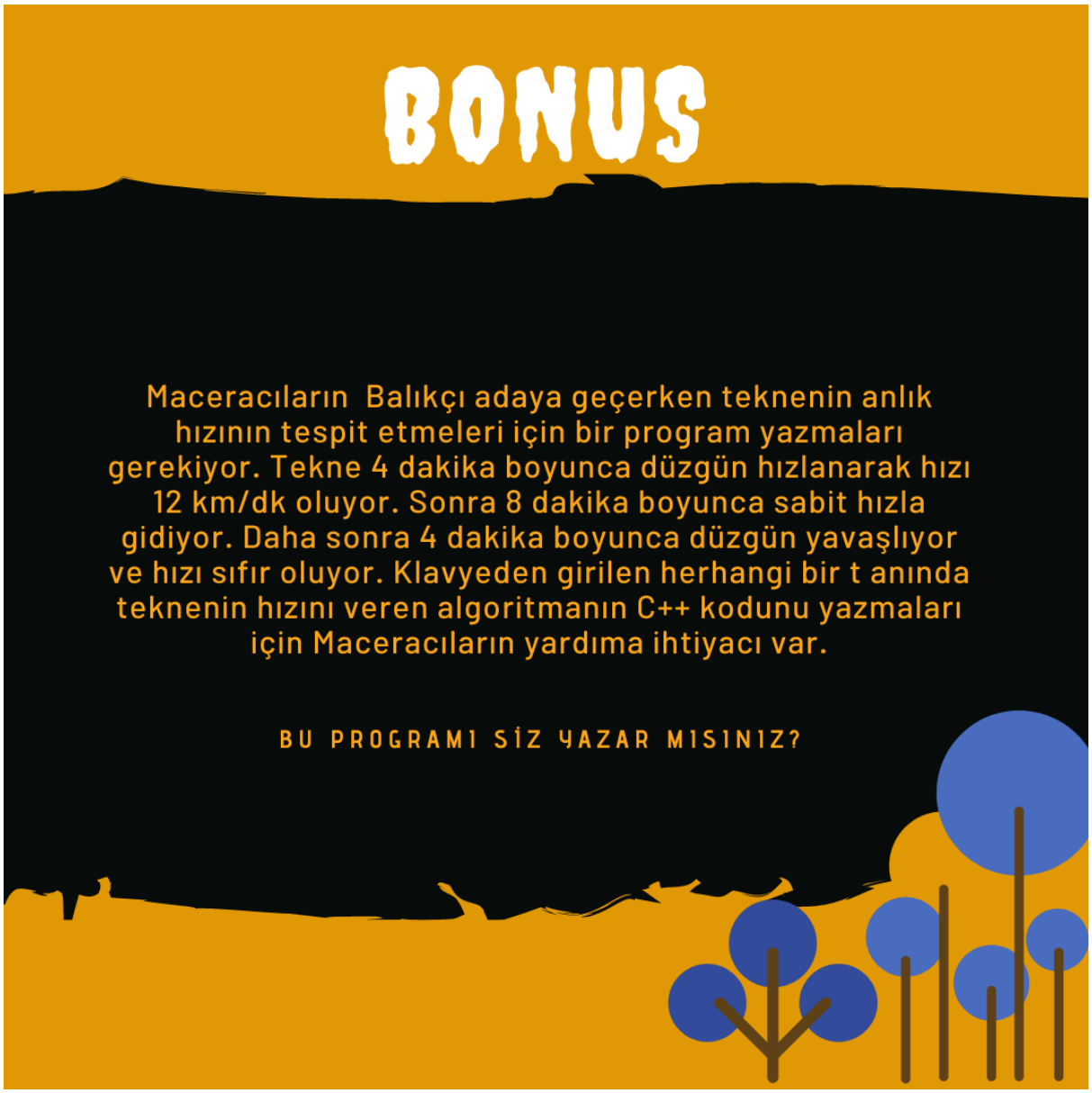

HAZİNEYİ AÇIN
Hazine adasına hoşgeldiniz.
Maceracılar, hazine sandığının anahtarını açmak için aşağıdaki 7x7'lik matristen oluşan bir matematik problemi ile karşılaşır. Bu matris 0-9 arası rastgele dağılmış rakamlardan oluşmaktadır.
Maceracılar, şifreyi çözmek için bu rakamların kullanım miktarını (sayısını) en azdan çoğa doğru sıralayan kodu yazar. Çünkü kodun çıktısı hazine sandığının şifresidir
SANDIĞI AÇMAYA HAZIR MISINIZ....
MATRİS
3 | 1 | 7 | 6 | 3 | 9 | 5
2 | 8 | 2 | 7 | 9 | 8 | 6
7 | 5 | 9 | 0 | 7 | 2 | 2
3 | 1 | 3 | 5 | 6 | 1 | 5
8 | 0 | 5 | 7 | 2 | 9 | 8
5 | 2 | 8 | 3 | 7 | 0 | 3
7 | 6 | 2 | 9 | 6 | 1 | 6
ANAHTAR
HAZİNE
ŞİFRE

BONUS PROBLEMİ

Maceracıların Balıkçı Ada'ya geçerken teknenin anlık hızının tespit etmeleri için bir program yazmaları gerekiyor. Sahilden yola çıkan tekne 4 dakika boyunca düzgün hızlanarak hızı 12 km/dk oluyor. Sonra 8 dakika boyunca sabit hızla gidiyor. Daha sonra 4 dakika boyunca düzgün yavaşlıyor ve hızı sıfır oluyor. Klavyeden girilen herhangi bir t anında teknenin hızını veren algoritmanın C++ kodunu yazmaları için maceracıların yardıma ihtiyacı var. Onlar için bu programı yazar mısınız?

1. İpucu: Hız formülü V = A*T olarak bilinmektedir. (V: hız, A: ivme, T: zaman)

2. İpucu: Tekne hızlanırken ve yavaşlarken ivmesi 3'tür.



GÖREVLERİN ÇÖZÜMLERİ

ÇÖZÜM 1:

Puanlama: Genel Yapı: 20 Puan

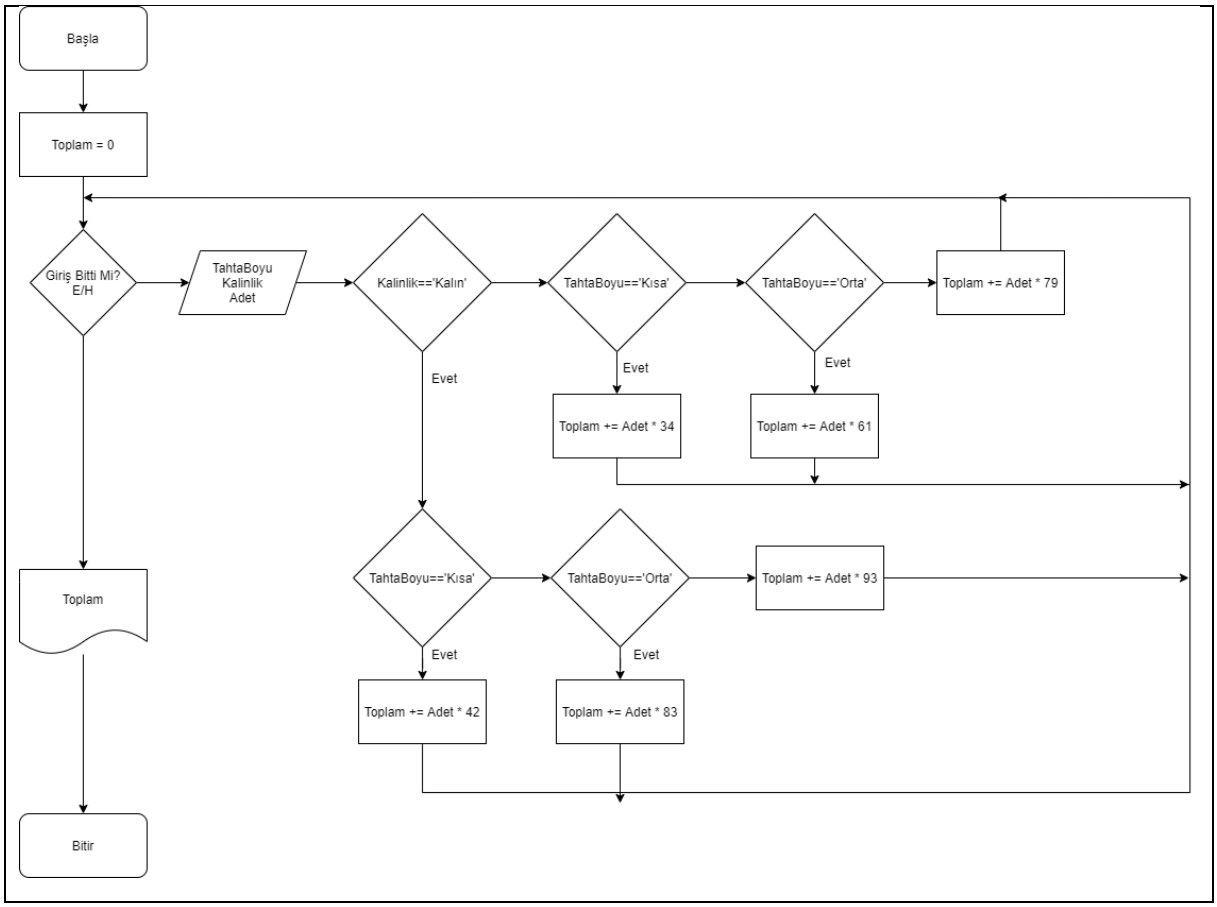
Değişken Tanımlama: 20 Puan

Koşulların Tanımlanması: 40 Puan

Toplam Hesaplama: 20 Puan

Çıktı: 1816

Akış Diyagramı:



ÇÖZÜM 2:

Puanlama: Fonksiyon Tanımlama Bölümü: 10 Puan

Satır Kontrolü: 20 Puan

Sütun Kontrolü: 20 Puan

Sağ Çapraz Kontrolü: 20 Puan

Sol Çapraz Kontrolü: 20 Puan

Ekrana Sonuç Yazdırma: 10 Puan

Çıktı: XXX

Kod:

```
// YARISMA PROBLEMI 2:
#include <iostream>
#include <stdlib.h>
using namespace std;
bool bittimi(char tahta[3][3])
{

    for(int i=0;i<3;i++)
    {
        if(tahta[i][0] == tahta[i][1] && tahta[i][1] == tahta[i][2] )
        {
            if(tahta[i][0]=='X')
                printf("XXX");
            else
                 printf("OOO");
            return true;
        }
    }
    for(int i=0;i<3;i++)
    {
        if(tahta[0][i] == tahta[1][i] && tahta[1][i] == tahta[2][i] )
        {
            if(tahta[i][0]=='X')
                printf("XXX");
            else
                printf("OOO");
            return true;
```

```
        }
    }
    if(tahta[0][0] == tahta[1][1] && tahta[1][1] == tahta[2][2] )
    {
        if(tahta[0][0]=='X')
            printf("XXX");
        else
            printf("OOO");
        return true;
    }
    if(tahta[0][2] == tahta[1][1] && tahta[1][1] == tahta[2][0] )
    {
        if(tahta[0][2]=='X')
            printf("XXX");
        else
             printf("OOO");
        return true;
    }
    return false;
}
int main()
{
    char tahta[3][3]={{'X','O','X'},{'X','X','O'},{'O','X','X'}};

    if(bittimi(tahta))
        cout <<"Oyun_Bitti";
    else
        cout <<"DEVAM";
}
```

ÇÖZÜM 3:

Puanlama: Balık Sınıfı Tanımlama Bölümü: 20 Puan

Türetilmiş Sınıfı Tanımlama Bölümü: 40 Puan

Ana Fonksiyon Bölümleri: 40 Puan

Çıktı: 288

Kod:

```
// YARISMA PROBLEMI 3:
#include <iostream>
#include <string.h>
#include <fstream>
using namespace std;
class Balik {
public:
   char tur[20];
   float agirlik;
};

class Levrek: public Balik {
public:
   Levrek(){
      strcpy(tur, "Levrek");
      agirlik = 0.450;
   }
   int hesapla(int adet){
      return adet * (agirlik * 40);
   }
};

class Palamut: public Balik {
public:
   Palamut(){
      strcpy(tur, "Palamut");
      agirlik = 0.600;
```

```
    }
    int hesapla(int adet){
        return adet * (agirlik * 25);
    }
};

class Cupra: public Balik {
public:
    Cupra(){
        strcpy(tur, "Cupra");
        agirlik = 0.500;
    }
    int hesapla(int adet){
        return adet * (agirlik * 30);
    }
};



int main() {
    ofstream dosya;
    dosya.open("hesap.txt");
    Levrek b1;
    Palamut b2;
    Cupra b3;
    int h = b1.hesapla(6) + b2.hesapla(4) + b3.hesapla(8);
    dosya << h << endl;
    dosya.close();
    return 0;
}
```

ÇÖZÜM 4:

Puanlama: Değişken Tanımlama Bölümü: 20 Puan

Tekrar Sayılarını Bulma: 40 Puan

Azdan Çoğa Doğru Rakamları Yazdırma: 40 Puan

Çıktı: 4018935627

Kod:

```
// YARISMA PROBLEMI 4:
#include <iostream>
using namespace std;

int main()
{
   // Tanimlama Bolumu
   int matris [7][7] ={{3,1,7,6,3,9,5},
                       {2,8,2,7,9,8,6},
                       {7,5,9,0,7,2,2},
                       {3,1,3,5,6,1,5},
                       {8,0,5,7,2,9,8},
                       {5,2,8,3,7,0,3},
                       {7,6,2,9,6,1,6}
                      };
   int m=7, n=7;
   int tekrar[10]={0};

   // Tekrar Sayilarini Bulma
   for(int i=0;i<m;i++){
      for(int j=0; j<n; j++){
         for(int k=0; k<10; k++){
            if(matris[i][j] == k){
               tekrar[k]++;
            }
         }
      }
```

```
    }


    /*
    //Tekrar Sayilarini Bulma Hizli Yol
    for(int i=0;i<m;i++){
        for(int j=0; j<n; j++){
            tekrar[matris[i][j]]++;
        }
    }
    */


    //Azdan Coga Dogru Rakamlari Yazdirma
    for(int i=0; i<10; i++){
        for(int j=0; j<10; j++){
                if(tekrar[j]==i)
                    printf("%d",j);
        }


    }
    return 0;
}
```

ÇÖZÜM BONUS:

Puanlama: Değişken Tanımlama ve Değer Okuma Bölümü: 20 Puan

Hızlanma Kontrolü Bölümü: 40 Puan

Yavaşlama Kontrolü Bölümü: 40 Puan

Kod:

```
// YARIŞMA PROBLEMI BONUS:
#include <iostream>
using namespace std;
int main() {
   float hiz;
   float zaman;
   float ivme;

   cout << "Zaman değerini giriniz (0-16 arasında olmali): " << endl;
   cin >> zaman;

   if(zaman <= 4.0){
      ivme = (12-0) / (4-0);
      hiz = ivme * zaman;
      cout << "Teknenin hizi: " << hiz << endl;
   }
   else if(zaman<=12.0){
      hiz = 12.0;
      cout << "Teknenin hizi: " << hiz << endl;
   }else if (zaman<=16.0){
      ivme = (0-12) / (4-0);
      hiz = 12 + ivme * (zaman-12);
      cout << "Teknenin hizi: " << hiz << endl;
   }else{
      cout << "Hatali giris!!" << endl;
   }
   return 0;
}
```

YAZILIM TEKNOLOJİLERİ

LİSE

DENEYAP TÜRKİYE