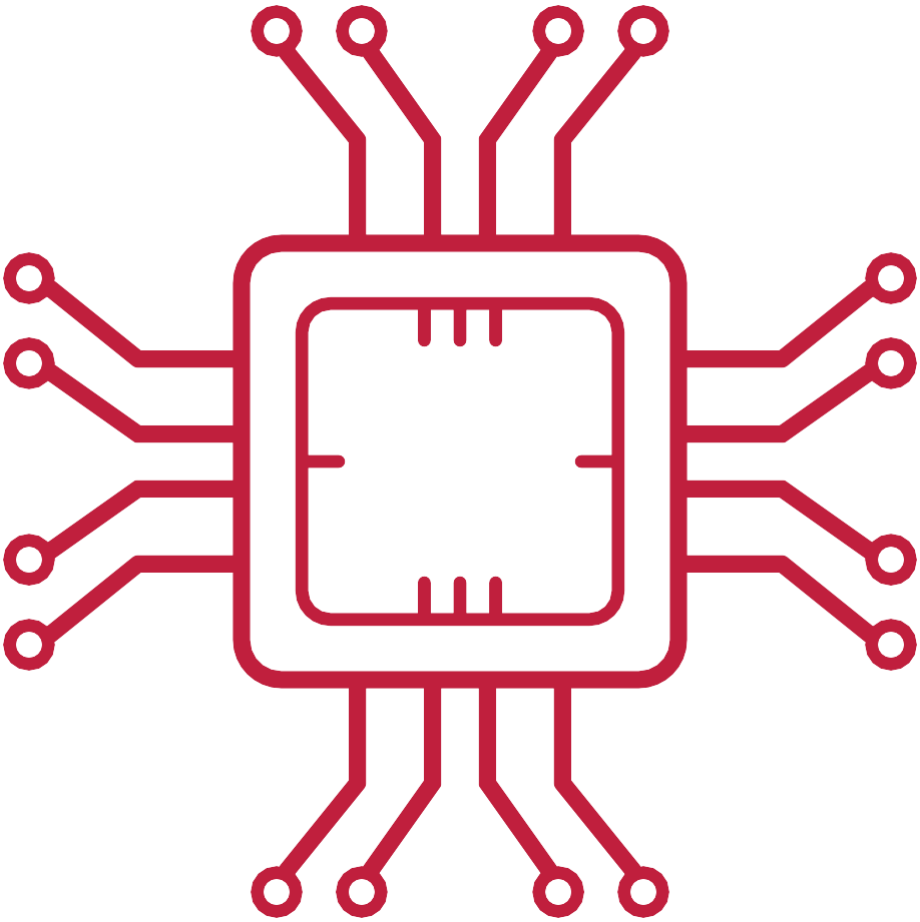
#MİLLİ TEKNOLOJİ HAMLESİ

DENEYAP TÜRKİYE

YAZILIM TEKNOLOJİLERİ

ORTAOKUL



TÜBİTAK

TÜRKİYE TEKNOLOJİ TAKIMI

DENEYAP

Teknoloji Atölyeleri

YAZILIM TEKNOLOJİLERİ

ORTAOKUL

Doç. Dr. Caner ÖZCAN
Doç. Dr. Rafet DURGUT
Doç. Dr. Sevil ORHAN ÖZEN
Dr. Sercan ÖZEN

TÜBİTAK

TÜBİTAK Deneyap Kitapları 7

YAZILIM TEKNOLOJILERI
ORTAOKUL

Doç. Dr. Caner ÖZCAN
Doç. Dr. Rafet DURGUT
Doç. Dr. Sevil ORHAN ÖZEN
Dr. Sercan ÖZEN

Yayım Tarihi: 2025

TÜBİTAK Deneyap Kitapları *DENEYAP TÜRKİYE*
Projesi kapsamında hazırlanmıştır.

ISBN 978-605-312-442-9
Yayıncı Sertifika No: 47703

TÜBİTAK Başkanı: Prof. Dr. Orhan AYDIN
Bilim ve Toplum Başkanı: Ömer KÖKÇAM
Genel Yayın Yönetmeni: Fatma BAŞAR
Editörler: Dr. Eda AŞILI, Hande BAĞDU
Telif İşleri Sorumlusu: Melike EROL

TÜBİTAK Bilim ve Toplum Başkanlığı
Tunus Caddesi No: 80 Kavaklıdere 06680 Ankara
Tel: (312) 298 96 50
e-posta: deneyap@tubitak.gov.tr
https://yayinlar.tubitak.gov.tr/deneyap-atolyesi

İçindekiler

Sunuş

Eğitim dünyası günlük hayatta teknolojisiz ortamda yer almayan, okul hayatlarında da öğrenme için teknolojiden faydalanan pek çok çevrim içi 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 çocuklarımızın adaptasyon sürecinde hazırlanan öğretim programlarının şekli ve uygulayıcıların kullandıkları öğrenme öğretme yöntemleri kilit noktadır.

Teknolojiyle birlikte değişen ve gelişen ihtiyaç ve pazar koşullarında çocuklarımızı 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 2019 yılında Deneyap Teknoloji Atölyeleri 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. Verilen eğitimlerden biri de "Yazılım Teknolojileri"dir. 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 tasarlanmıştır. Öğrenciler eğitim sonunda gerçek dünya problemlerine yönelik görev temelli bir deneyimler edinmiş olacaktır.

Okuyuculara sunulan bu kitap Deneyap Teknoloji Atölyeleri kapsamında ortaokul öğrencilerine eğitim verecek eğitmenler için hazırlanan hibrit öğretim stratejisi ile kurgulanmış Yazılım Teknolojileri öğretim programını içermektedir. Ancak kitabın tamamı ortaokul 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.

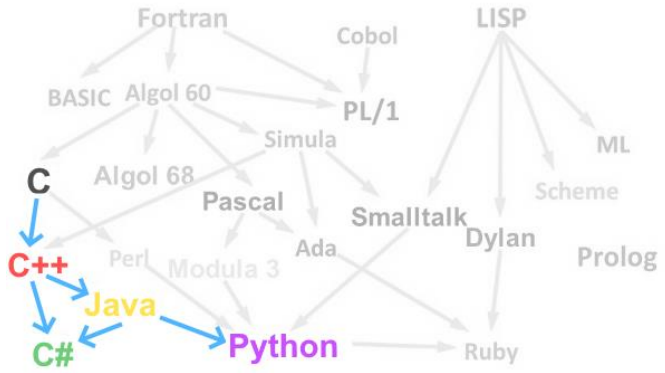
Dersin Çıktıları

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.
- İşaretçi ve 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:**
  Yüz Yüze: Programlamaya Giriş, Programlama Terimleri, Matematiksel İşlemler, Karşılaştırma İşlemleri, Mantıksal İşlemler, Sayı Sistemleri, Algoritma Tasarımı, Operatörler ve Kavramlar, Akış Şemalarına Giriş, Kodlamaya Geçiş, Program Yazmanın Adımları
  Çevrim İçi: Örnek Akış Şemaları, C++ Programlama Dilinin Özellikleri, IDE (Derleyici) Kurulumu ve İlk Programlama
- **Hafta 2:**
  Yüz Yüze: C++ Tanımlamalar ve Operatörler, Değişken Tanımlama, Veri Tipleri, Sabitlerin Tanımlanması, Operatörlerin Kullanımı
  Çevrim İçi: Değişkenler, Veri Tipleri ve Operatörlerle İlgili Örnekler
- **Hafta 3:**
  Yüz Yüze: Karar Verme Komutları (if-else, switch), If ifadesi, Switch İfadesi, Mantıksal Operatörler, Operatör Öncelikleri, Döngü Komutları (for, while, do-while), İç İçe Döngüler, Rastgele Sayılar
  Çevrim İçi: Karar Verme Komutları İle İlgili Örnekler
- **Hafta 4:**
  Yüz Yüze: Diziler, Karakter Katarları, Örnek Problem Çözümleri
  Çevrim İçi: Diziler ve Karakter İle İlgili Örnekler
- **Hafta 5:**
  Yüz Yüze: Fonksiyonlar, Fonksiyon Sözdizimi, Fonksiyon Çağırma
  Çevrim İçi: Fonksiyonlar İle İlgili Örnekler
- **Hafta 6:**
  Yüz Yüze: Nesne Yönelimli Programlama, Sınıf Tanımlama, Metod Tanımlama, Nesne Değişkenleri Oluşturma, Yapıcı ve Yıkıcı Metotlar
  Çevrim İçi: Nesne Yönelimli Programlama İle İlgili Örnekler
- **Hafta 7:**
  Yüz Yüze: Kütüphane Kullanımı ve Dosyalama İşlemleri
  Çevrim İçi: Kütüphane Kullanımı ve Dosyalama İle İlgili Örnekler

Dersin İşlenişi

Deneyap Teknoloji Atölyeleri Yazılım Teknolojileri Dersi öğretim programı, karmaşık becerilerin öğretiminde etkili olduğu kanıtlanmış Dört Bileşenli Öğretim Tasarımı (4C/ID) modelinin temel ilkelerine uygun bir şekilde tasarlanmıştır. Bu modelin uygulanması için, yüz yüze ve çevrim içi ortamların bir arada kullanıldığı hibrit bir öğretim stratejisi benimsenmiştir. Dersin haftalık akışı, her hafta bir yüz yüze ders ve bir çevrim içi etkinlik olmak üzere iki ana bileşenden oluşmaktadır. Bu yapı, öğrencilerin yüz yüze derslerde edindikleri teorik bilgileri pekiştirmelerine ve çevrim içi etkinliklerde öğrendiklerini aktif olarak uygulamalarına olanak tanımaktadır. Bu düzenleme, öğrencilere bireysel çalışma ve tekrar imkânı sunarken, aynı zamanda akran öğrenmesi ve eğitmen rehberliğiyle desteklenmelerini sağlamaktadır.

4C/ID modeli, öğrencilerin gerçek yaşam durumlarına yönelik öğrenme görevleri aracılığıyla yeni bilgi, beceri ve tutumları entegre etmesini ve koordine etmesini hedefler (Costa ve Miranda, 2019). Geleneksel yaklaşımların aksine, 4C/ID modeli bu yaklaşımı tersine çevirir; öncelikle profesyonel görevleri belirleyerek başlar ve bu görevleri öğrenme görevlerine dönüştürür. Teorik bilgiler ise bu öğrenme görevlerini tamamlamaya yardımcı olacak şekilde sunulur (Frere Jean vd., 2019). Yüz yüze derslerde teorik bilgiler (Destekleyici Bilgiler) afiş, sunum gibi araçlarla verilirken, çevrim içi etkinlikler pratik uygulamalar ve ödev sunumları (Öğrenme Görevleri ve Kısmi Görev Uygulaması) için ideal bir ortam sunmaktadır. Bu sayede, öğrencilerin öğrendikleri bilgileri aktif olarak kullanmaları ve programlama becerilerini yapılandırılmış bir şekilde geliştirmeleri amaçlanmaktadır.

Dört Bileşenli Öğretim Tasarımı (4C/ID) ve Hibrit Stratejideki Uygulaması

Öğretim tasarımı, eğitim-öğretim ortamlarında uygulanacak her türlü faaliyetin belli bir plana göre uygulanması olup genel amacı, öğrenmeyi daha verimli ve daha kolay hâle getirmektir (İşman, 2015). Yazılım geliştirmek tipik bir karmaşık beceridir çünkü programlama dilleri, veri tabanları gibi konularda kapsamlı bilgi gerektirir ve geliştirilen yazılımı çalıştırmak, temiz kod yazmak gibi birden fazla becerinin entegrasyonunu gerektirir (van Merriënboer, 2016). Bu nedenle, bu dersin öğretim tasarımı Dört Bileşenli Öğretim Tasarımı (4C/ID) modelinden esinlenerek planlanmıştır.

Bu model, geleneksel tasarım yaklaşımlarının aksine, teorik bilgi sunumuyla başlamak yerine, öncelikle profesyonel görevleri belirler ve bunları öğrenme görevlerine dönüştürür (Frerejean vd., 2019; van Merriënboer, Kirschner & Kester, 2003). Bu sayede öğrenciler, "Biz bunu neden öğreniyoruz?" sorusuna bir cevap bularak, öğrendiklerini bir yazılımcı olarak anlamakta,güçlük yaşamazlar. Seçilen hibrit strateji, bu yaklaşımı güçlendirmektedir. Model, gerçek yaşam durumlarına yönelik öğrenme görevlerinin öğrenmeyi geliştiren temel unsur olduğunu savunur (van Merriënboer ve Kester, 2014). Bu kapsamda, modelin öğrenme görevleri ile ilgili basamağı, C++ gibi zor ve dikkat gerektiren bir dilin öğretiminde öğrencilerin motivasyonunu düşürmemek amacıyla, doğrudan bütüncül görevler yerine parçalı bir yapıda ele alınmıştır. Bu parçalı görevler, birkaç haftalık konu birikiminin ardından, bütüncül ve günlük hayatla ilişkili bir proje görevi ile birleşmektedir.

(1) Öğrenme Görevleri:

Öğrenme görevleri, öğretim modelinin bel kemiğini oluşturur ve pratikte karşılaşılan gerçek yaşam durumlarına dayanır. Yüz yüze derslerde temel kavramlar işlendikten sonra, çevrim içi etkinliklerde bu kavramları içeren **transfer soruları** ile öğrenciler, edindikleri bilgiyi yeni ve farklı durumlara uygulama fırsatı bulurlar. Bu görevler, bir profesyonelin karşılaşabileceği en basit aşamalardan başlayarak öğrencinin üstesinden gelmesi gereken karmaşıklık düzeyine doğru ilerleyecek şekilde, basitten karmaşığa doğru sıralanır (van Merriënboer, Kirschner ve Kester, 2003). Örneğin, 4. haftada diziler konusu işlendikten sonra, çevrim içi etkinlikte "Dizilerde Algoritmalar Üzerine Transfer Sorusu" ile öğrencilerin konuyu pekiştirmesi sağlanır. Transfer sorusu, öğrencinin bir konuyu ezberlemesi yerine, öğrendiği beceri ve bilgiyi farklı bir problem bağlamında kullanmasını sağlayan, zorluk seviyesi artırılmış bir uygulama görevidir. Bu görevler, haftalık bütünlük içinde parçalı görevler şeklinde sunulur ve birkaç haftada bir daha bütüncül proje görevlerinde birleşir.

Bu teknik, dersin bel kemiğidir. Öğrencinin "Bu bilgiyi nerede kullanacağım?" sorusunu ortadan kaldırır ve öğrenmeyi doğrudan anlamlı bir bağlama oturtur.

- **Gerçek Yaşam Senaryoları Kullanımı:** Öğrenme görevleri, bir yazılımcının meslek hayatında karşılaşabileceği gerçekçi ve motive edici problemlerden türetilmelidir. Model, öğrenmenin en kalıcı şekilde gerçek yaşam durumlarına dayalı görevlerle geliştiğini savunur (van Merriënboer ve Kester, 2014).

  Uygulama Örneği: Basitçe "ekrana adını yazdır" görevi yerine, "kullanıcıdan adını ve yaşını alıp ehliyet alıp alamayacağını söyleyen bir mini program" tasarlatmak, görevi anında daha anlamlı kılar.

- **Basitten Karmaşığa Sıralama:** Görevler, bir profesyonelin karşılaşacağı en basit durumlardan başlayıp öğrencinin ustalaşması beklenen karmaşıklık seviyesine ulaşacak şekilde, bir yapı iskelesi gibi dikkatlice sıralanmalıdır (van Merriënboer, Kirschner ve Kester, 2003). Her görev, bir sonrakinin temelini oluşturmalıdır.
- **Parçalı ve Bütüncül Görev Dengesi:** C++ gibi zor bir dilde öğrencilerin motivasyonunu ve özgüvenini korumak kritik önemdedir. Bu nedenle konular başlangıçta sindirilebilir, *parçalı görevler* halinde sunulur. Birkaç haftalık birikimin ardından bu parçalar, daha büyük ve *bütüncül proje görevlerinde* birleştirilir.

  Uygulama Örneği: Önce değişkenler, sonra koşul yapıları, ardından döngüler ayrı ayrı öğretilir. Sonrasında ise bu üçünü birleştiren, "girilen 10 sayıdan tek olanlarını toplayan bir program" gibi bütüncül bir göreve geçilir.

- **Transfer Soruları Tekniği:** Çevrim içi etkinliklerde, öğrencilerin öğrendikleri bilgiyi yeni ve daha önce karşılaşmadıkları durumlara uygulamalarını sağlamak için **"transfer soruları"** kullanılır. Bu sorular, ezberden ziyade *gerçek problem çözme becerisini* ve öğrenmenin derinliğini gösterir.

(2) Destekleyici Bilgiler:

Destekleyici bilgi, öğrencilerin verilen görev veya problemlerin nasıl çözüleceğine yönelik yaklaşımları tanımlar ve genellikle "teori" olarak adlandırılır (Frere Jean vd., 2019). Hibrit stratejide, bu bilgiler yüz yüze derslerde afiş, sunum, tartışma panoları ve çeşitli bilgi kartları

gibi materyallerle sunulur. Bu materyaller, öğrencilerin öğrenme görevlerini yerine getirirken ihtiyaç duydukları zihinsel modelleri ve bilişsel stratejileri oluşturmalarına yardımcı olur. Eğitmen, bu aşamada öğrenciye görevdeki performansına göre bilişsel geri bildirimler vererek onların doğru çözüm yollarını keşfetmelerini teşvik eder. Örneğin, eğitmen öğrenciye çözümünü bir akranının çözümüyle karşılaştırmasını önerebilir (Costa ve Miranda, 2019).

Destekleyici bilgiler, öğrencilerin öğrenme görevlerini başarıyla tamamlamaları için gereken teorik altyapıyı ve zihinsel modelleri inşa etmelerini hedefler.

- **Çoklu ve Erişilebilir Materyal Kullanımı:** Destekleyici bilgiler (genellikle "teori" olarak adlandırılır), farklı öğrenme stillerine hitap edecek şekilde sunulmalıdır. Yüz yüze derslerde **afişler, sunumlar, bilgi kartları ve tartışma panoları** gibi zengin ve kolay erişilebilir materyaller kullanılmalıdır (Frere Jean vd., 2019).
- **Bilişsel Geri Bildirim Verme:** Etkili geri bildirim, sadece "doğru" ya da "yanlış" demek değildir. Öğrencinin *düşünme sürecini* hedef alan geri bildirimler verilir.

  Uygulama Örneği: Bir öğrencinin hatalı koduna sadece doğrusunu yazmak yerine, şu sorularla rehberlik edin: *"Bu kod sence neden beklediğin gibi çalışmıyor olabilir? Döngünün bitiş koşulunu tekrar kontrol ettin mi? Değişkenin türü, yapmak istediğin işlem için uygun mu?"* Bu yaklaşım, öğrencinin çözümünü bir akranınınkiyle karşılaştırmasını önermek kadar etkili bir bilişsel geri bildirim tekniğidir (Costa ve Miranda, 2019).

(3) Yöntemsel Bilgi:

Yöntemsel bilgi, bir görevin rutin ve tekrara dayalı basamaklarının nasıl gerçekleştirileceğine dair, ihtiyaç anında sunulan işlemsel bilgidir. Bu destek, öğrencinin bir görevde takılıp kalmasını önlemek için bir "yapı iskelesi" (scaffolding) işlevi görür. Özellikle eş zamanlı çevrim içi oturumlar sırasında, bir öğrenci görevini padlet üzerinde tamamlarken teknik bir zorluk yaşadığında, eğitmen yorumlar bölümünden veya sözlü olarak anlık yöntemsel bilgi sağlayabilir. Öğrencinin yetkinliği arttıkça bu destek kademeli olarak geri çekilir ve öğrencinin özerkliği teşvik edilir.

Yöntemsel bilgi, öğrenme görevinin rutin kısımlarının nasıl çözüleceğini öğrenciye anlatır ve öğrenciye ihtiyacı olduğu anda, anlık olarak sunulur. Hibrit stratejinin çevrim içi etkinlikleri bu bileşen için son derece uygundur. Öğrenciler bir görevi yerine getirirken zorlandıklarında, eğitmen devreye girerek onlara işlemsel bilgi içeren hızlı geri bildirimler ve ipuçları sunar. Örneğin, bir döngüde hata yapan bir öğrenciye eğitmen, döngü yapısının doğru kullanımı hakkında anlık bir hatırlatma yapabilir. Öğrenci zaman içinde deneyim kazandıkça bu tür yöntemsel bilgiye olan ihtiyaç ortadan kalkar ve destek giderek azaltılır.

Bu teknik, öğrencinin bir görevin rutin basamaklarında takılıp motivasyon kaybetmesini önlemek için anlık olarak desteklenmesini içerir.

- **İhtiyaç Anında Rehberlik:** Öğrenci bir görevi yerine getirirken zorlandığında (örneğin bir for döngüsünün söz dizimini unuttuğunda), eğitmen devreye girerek göreve devam etmesini sağlayacak anlık, *işlemsel bilgi* vermelidir. Bu, akışın devamı için kritik bir müdahaledir.
- **Desteği Zamanla Azaltma (Scaffolding):** Bu destek, bisiklete binmeyi öğrenen bir çocuğun destek tekerlekleri gibidir. Öğrenci deneyim kazandıkça ve temel becerileri

otomatikleşmeye başladıkça, eğitmen bu tür doğrudan yardımları bilinçli olarak azaltmalıdır. Amaç, öğrencinin kendi dengesini bularak bağımsız bir şekilde ilerlemesini sağlamaktır.

(4) Kısmi Görev Uygulaması:

Bu bileşen, öğrencilerin bir görevin içindeki rutin becerileri otomatikleştirinceye kadar tamamladıkları alıştırma görevleridir. Hibrit stratejinin çevrim içi ödev sunumları, bu kısmi görevleri oluşturur. Bu aşamada sunulan transfer soruları, öğrencilerin temel konuyu pratikleştirme ve otomatikleşme sağlama amacıyla çözdükleri görevlerdir. Bu görevler, eğitmenin, öğrencilerin yöntemsel bilgiye ihtiyaç duymadan bir beceriyi ne kadar iyi uygulayabildiğini anlaması için bir değerlendirme aracı olarak da kullanılır. Transfer soruları, doğrudan derste anlatılan bir örneğin aynısı olmak yerine, öğrenciden o örneğin mantığını ve temel prensibini kullanarak daha önce karşılaşmadığı bir problemi çözmesini bekleyecek şekilde tasarlanır. Örneğin, bir öğrenciye dizideki en büyük sayıyı bulan bir kod yazması öğretilmişse, transfer sorusu olarak "bir dizideki çift sayıların toplamını bulan bir program yazması" istenebilir.

Eğitmenin Rolü

Öğretim modelinde eğitmenin rolü, süreci yöneten, izleyen ve öğrencilere rehberlik eden bir rehber niteliğindedir. Unutulmamalıdır ki eğitmenin öğrenmeye rehberlik etmesi, bilgiyi doğrudan aktarmasından çok daha zor olacaktır. Eğitmen, yüz yüze derslerde öğrencileri monotonluktan çıkarmak ve motivasyonlarını artırmak için grup etkinlikleri ve motivasyon oyunları düzenlerken, çevrim içi etkinliklerde öğrencilerin transfer soruları üzerindeki çalışmalarını takip eder ve geri bildirimler sunar.

Öğrencilere doğrudan doğru cevapları vermek yerine, onları düşünmeye teşvik eden eleştirel sorular yönelterek öğrenme sürecini derinleştirmeyi hedefler. Örneğin aşağıdaki gibi geri bildirimler kullanabilir:

- "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."

Ayrıca, eğitmen bilinçli olarak akran öğrenmesini teşvik ederek doğru yanıtlara ulaşan öğrencileri diğerleriyle eşleştirip birbirlerine görevleri açıklamalarını isteyebilir. Bu esnada süreci yakından izleyerek yanlış öğrenmelerin önüne geçer. Eğitmenin en önemli görevi, her haftanın içeriğini, kazanımlarını ve ders akışını içeren eğitmen rehberini titizlikle takip etmek ve derse hazırlıklı gelmektir. Bu rehberlik yaklaşımı, öğrencinin kendi öğrenmesini yansıtma ve bütüncül bir yaklaşımla karmaşık beceriler edinme sürecini desteklemektedir.

Öğrenme Görevlerinin Tasarlanması ve Öğrenme Sürecinde Kullanılacak Öğretim Yöntem ve Teknikleri

İşbirlikli Öğrenmenin Kullanımı

Öğrencilerin karmaşık becerileri kazanmasını hedefleyen etkili öğrenme ortamlarının tasarımında, **Dört Bileşenli Öğretim Tasarımı (4C/ID) Modeli** güçlü bir pedagojik çerçeve sunar. Bu model, karmaşık öğrenme görevlerinin bireysel başarıdan ziyade, öğrencilerin işbirliği içinde çalışmasını gerektiren bir yaklaşımla ele alınmasını önerir. Bu bağlamda, işbirlikli öğrenme ve grup çalışmaları, 4C/ID modelinin temel ilkelerinden biri olarak öne çıkar.

İşbirlikli öğrenme, öğrencilerin sadece kendi bireysel öğrenmelerinden değil, aynı zamanda grup üyelerinin de en üst düzeyde öğrenmesini sağlamayı amaçlayan yapılandırılmış bir süreçtir. 4C/ID modelinin karmaşık öğrenme görevlerinin doğası gereği yüksek bilişsel yük içermesi, bu yükün grup üyelerine paylaştırılmasını ve böylece öğrenmenin daha verimli hâle getirilmesini gerektirir. Bu yaklaşım, öğrencilere gerçek dünya senaryolarını yansıtan görevler üzerinde çalışırken, kendi öğrenme hedeflerinin yanı sıra grubun ortak hedeflerine de odaklanma sorumluluğu verir.

Bu işbirlikli süreç, geleneksel grup çalışmalarının ötesine geçerek öğrencilerin bilişsel ve sosyal becerilerini eş zamanlı olarak geliştirir. Gruplar içinde öğrenciler, etkili iletişim kurma, problem çözme, açıklama yapma ve geri bildirim verme gibi üst düzey bilişsel ve sosyal becerileri doğal bir şekilde edinir. Bu etkileşimler, yalnızca konuya dair derinlemesine bir şema oluşturmalarına yardımcı olmakla kalmaz, aynı zamanda 4C/ID modelinin hedefleri arasında yer alan öz-yönetimli ve özerk öğrenme yetkinliklerini de pekiştirir. Dolayısıyla, 4C/ID modelinin rehberliğinde tasarlanan grup çalışmaları, öğrencilerin hem akademik bilgiye hakimiyetini artırır hem de gelecekteki iş yaşamlarında başarılı olmaları için kritik öneme sahip olan işbirliği ve eleştirel düşünme becerilerini geliştirir.

Eleştirel Düşünme Tekniklerinin Kullanımı

Öğrenme görevlerinin öğrencilere sunulmasında, onların eleştirel düşünme becerilerini geliştirmeyi hedefleyen çeşitli tekniklerden yararlanılmaktadır. Bu teknikler, öğrencilerin bilgiyi derinlemesine analiz etmelerine, farklı bakış açıları geliştirmelerine ve sonuçlar çıkarmalarına yardımcı olur. Bu bağlamda, öğrenme görevlerinin tasarımı sürecinde eleştirel düşünme tekniklerinin bilinçli bir şekilde entegre edilmesi büyük önem taşımaktadır.

Aşağıda, öğrenme görevlerinde sıklıkla kullanılan bazı eleştirel düşünme teknikleri örnekleriyle açıklanmıştır:

- **Sıralama:** Öğrencilerden verilen bilgileri, algoritmik adımları veya kod satırlarını mantıksal bir sıraya koymaları istenir. Bu teknik, öğrencilerin süreçleri anlamalarını, neden-sonuç ilişkilerini kurmalarını ve bilgiyi organize etme becerilerini geliştirir. *Örnek: Bir algoritmanın adımlarını karışık olarak verip, öğrencilerden doğru sırayı oluşturmalarını istemek.*
- **Gruplama:** Öğrencilerden ortak özelliklere sahip nesneleri, kavramları veya veri kümelerini belirli kriterlere göre gruplandırmaları beklenir. Bu teknik, öğrencilerin benzerlikleri ve farklılıkları ayırt etmelerine, kategorizasyon becerilerini geliştirmelerine ve bilgiyi anlamlı bir şekilde yapılandırmalarına yardımcı olur. *Örnek: Verilen farklı programlama komutlarını işlevlerine göre gruplandırmalarını istemek.*

- **Talimat Verme:** Öğrencilerden belirli bir görevi gerçekleştirmesi için bir akranına veya sanal bir varlığa adım adım talimatlar vermesi istenir. Bu teknik, öğrencilerin düşüncelerini açık ve net bir şekilde ifade etmelerini, süreçleri detaylı olarak analiz etmelerini ve başkalarının bakış açısını anlamalarını sağlar. *Örnek: İki sayının ortalamasını bulan bir kodun sözde kodunu yazan talimatları arkadaşına anlatmak.*
- **Tahmin Etme ve Çıkarımda Bulunma:** Öğrencilerden verilen bilgilere, kod parçalarına veya senaryolara dayanarak olası sonuçları veya çözümleri tahmin etmeleri ve çıkarımlarda bulunmaları beklenir. Bu teknik, öğrencilerin mantıksal akıl yürütme becerilerini geliştirmelerine, neden-sonuç ilişkilerini anlamalarına ve belirsizliklerle başa çıkmalarına yardımcı olur. *Örnek: Bir kodun çıktı vermeden önceki değerlerini verip, öğrencilerden beklenen ekran çıktısını tahmin etmelerini istemek.*
- **Analiz ve Sentez:** Öğrencilerden karmaşık bir bilgiyi veya problemi parçalarına ayırarak incelemeleri (analiz) ve ardından bu parçaları bir araya getirerek yeni bir bütün oluşturmaları veya bir çözüm üretmeleri (sentez) istenir. Bu teknik, öğrencilerin bilgiyi derinlemesine anlamalarını, farklı unsurlar arasındaki ilişkileri görmelerini ve yaratıcı çözümler üretmelerini teşvik eder. *Örnek: Farklı veri kaynaklarından elde edilen bilgileri analiz ederek hatalı ya da fazla olan koda karar vermesini istemek.*

Bu eleştirel düşünme teknikleri, öğrenme görevlerinin daha anlamlı ve etkileşimli hale gelmesini sağlayarak öğrencilerin öğrenme süreçlerine aktif katılımlarını teşvik eder ve onların üst düzey düşünme becerilerini geliştirir.

SCAMPER Tekniğinin Kullanımı

SCAMPER, öğrenme görevlerinin tasarımında ve yaratıcı düşünmenin teşvik edilmesinde kullanılan bir beyin fırtınası tekniğidir. Bu teknik, özellikle **Dört Bileşenli Öğretim Tasarımı (4C/ID) Modeli** kapsamında, karmaşık öğrenme görevlerinin geliştirilmesinde güçlü bir araç olarak kullanılır.

SCAMPER tekniği, öğrenme görevlerinin tasarım aşamasında, mevcut bir fikri, nesneyi veya problemi yenilikçi yollarla dönüştürmek için kullanılır. 4C/ID modeli, öğrencilere gerçek dünya problemlerini yansıtan karmaşık görevler sunmayı temel aldığından, bu görevlerin rutin ve basit olmaktan kaçınması gerekir. SCAMPER, tam da bu noktada devreye girerek, öğretim tasarımcılarına ya da eğitimcilere, öğrencilerin yaratıcı ve eleştirel düşünme becerilerini zorlayacak farklı senaryolar ve problemler oluşturma konusunda rehberlik eder.

Bu teknik, sadece öğretmen tarafından görevlerin tasarlanmasında değil, aynı zamanda öğrencilerin bir problem üzerinde düşünürken kendi içlerinde bir yönlendirme mekanizması olarak da kullanılabilir. Bir öğrenci grubu, karmaşık bir problemi çözerken SCAMPER'ın sunduğu soru kalıplarını kullanarak (örneğin, "Bu çözümün yerine başka ne koyabiliriz?" veya "Hangi bileşenleri birleştirebiliriz?"), probleme farklı açılardan yaklaşabilir ve daha yaratıcı çözümler geliştirebilir.

SCAMPER'ın akrostişini oluşturan temel bileşenler ve kullanım örnekleri şunlardır:

- **S - Substitude (Yerine Koyma):** Bir öğenin yerine neyin geçebileceği sorusunu yöneltir. *Örnek: "Bu algoritmanın yerine başka hangi veri yapısını kullanırsak program daha hızlı çalışır?"*
- **C - Combine (Birleştirme):** Farklı öğelerin nasıl bir araya getirilebileceğini sorgular. *Örnek: "Hangi iki kod bloğunu birleştirirsek daha etkili bir çözüm elde edebiliriz?"*

- **A - Adapt (Uyarlama):** Bir fikrin veya nesnenin nasıl başka bir bağlama uyarlanabileceğini araştırır. *Örnek: "Bu tasarım ilkesini farklı bir programlama diline nasıl uygulayabilirim?"*
- **M - Modify, Magnify, Minify (Değiştirme, Büyütme, Küçültme):** Bir şeyin özelliklerinin nasıl değiştirilebileceğini, büyütülebileceğini veya küçültülebileceğini inceler. *Örnek: "Bu fonksiyonun parametrelerini artırırsam, çıktısı nasıl değişir?"*
- **P - Put to Other Uses (Başka Amaçla Kullanma):** Bir öğenin orijinal amacı dışında nasıl kullanılabileceği üzerine düşünmeyi sağlar. *Örnek: "Bu değişkeni başka hangi amaçla kullanabilirim?"*
- **E - Eliminate (Yok Etme):** Bir şeyden hangi unsurların çıkarılabileceğini sorgular. *Örnek: "Bu döngüden bir koşulu çıkarsam, programın davranışı nasıl etkilenir?"*
- **R - Reverse, Rearrange (Tersine Çevirme, Yeniden Düzenleme):** Bir şeyin sıralamasının veya düzeninin nasıl değiştirilebileceğini araştırır. *Örnek: "Bu dizideki elemanların sıralamasını tersine çevirirsem, sonuç ne olur?"*

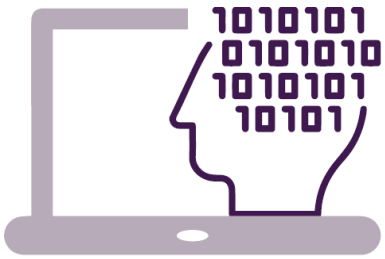
SCAMPER, öğrencilerin sadece mevcut bilgiyi kullanmasını değil, aynı zamanda bu bilgiyi dönüştürerek yeni ve orijinal çözümler üretmesini teşvik eden güçlü bir düşünme aracıdır.

Eğitmenin Yazılım Teknolojileri Dersinin Süreçlerinde Kullanabileceği Değerlendirme Teknikleri

Yazılım Teknolojileri dersinde değerlendirme, öğrenme sürecinin sonunda yer alan statik bir not verme eylemi olarak değil, sürecin kendisine entegre edilmiş dinamik ve motive edici bir unsur olarak tasarlanmıştır. Bu yaklaşımın temelinde iki felsefe yatmaktadır: Oyun Temelli Düşünme ve Otantik Değerlendirme **Oyun Temelli Düşünme (Oyunlaştırma)** ve **Otantik Değerlendirme**. Oyunlaştırma ile öğrencilerin etkinliklere katılımı, rekabet ve başarı hissi üzerinden artırılırken (Hamari, Koivisto & Sarsa, 2014); otantik değerlendirme ile öğrencilerin gerçek bir yazılım ekibinde karşılaşacakları rollere ve yetkinliklere dayalı olarak değerlendirilmeleri sağlanır (Gulikers vd., 2004).

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.

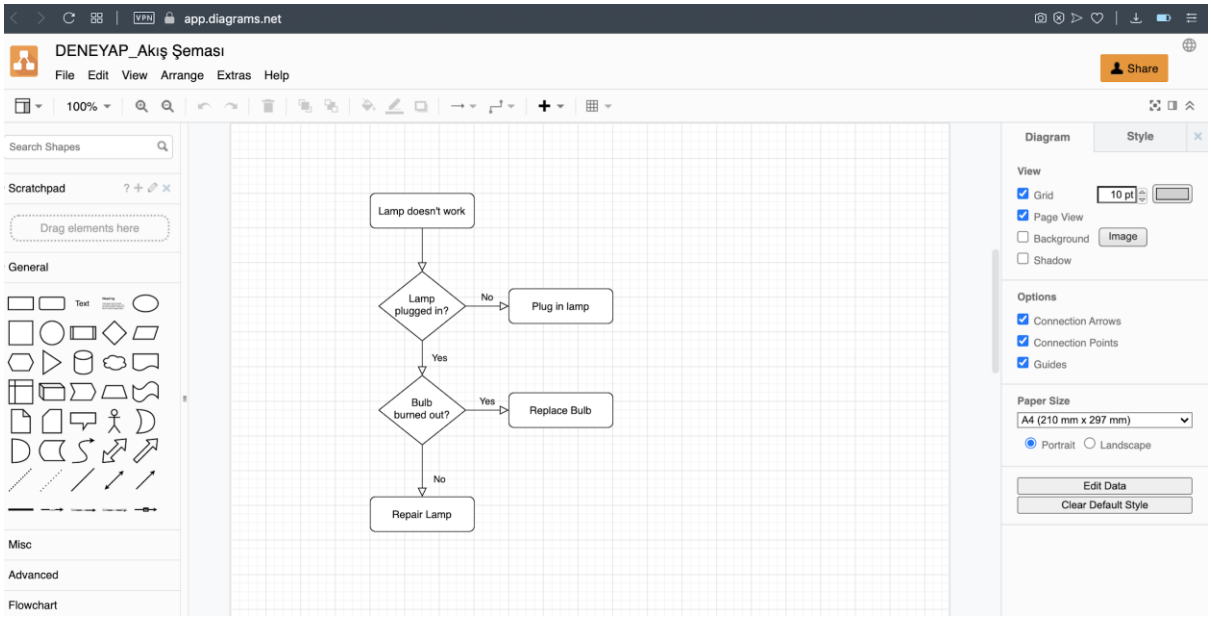
Sunulan bu yapılandırılmış öğretim modeli, 4C/ID'nin temel felsefesini hayata geçirerek öğrenme ve değerlendirme süreçlerini birbirinden ayrılmaz bir bütün haline getirir. Yüz yüze oturumlarda sunulan destekleyici bilgiler (Bileşen 2), öğrencileri senkron çevrim içi oturumlardaki hibrit kısmi görev pratiğine (Bileşen 4) hazırlar. Bu pratik sırasında hem otomatikleştirilmiş hem de otantik yöntemlerle toplanan beceri rozetleri, dönem sonunda gerçekleştirilecek olan bütüncül öğrenme görevini (Bileşen 1) yapacak olan proje ekiplerinin stratejik ve yetkinlik bazlı olarak oluşturulmasını sağlar. Bu hibrit yaklaşım, hem değerlendirme sürecinin verimliliğini ve ölçeklenebilirliğini artırır hem de otantik becerilerin derinlemesine ölçülmesine olanak tanır.

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 çevrim içi diyagram çizim uygulaması olan app.diagrams.net'tir. 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 3'te app.diagrams.net uygulamasının arayüzü verilmektedir.



Resim 3. *App.diagrams.net 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

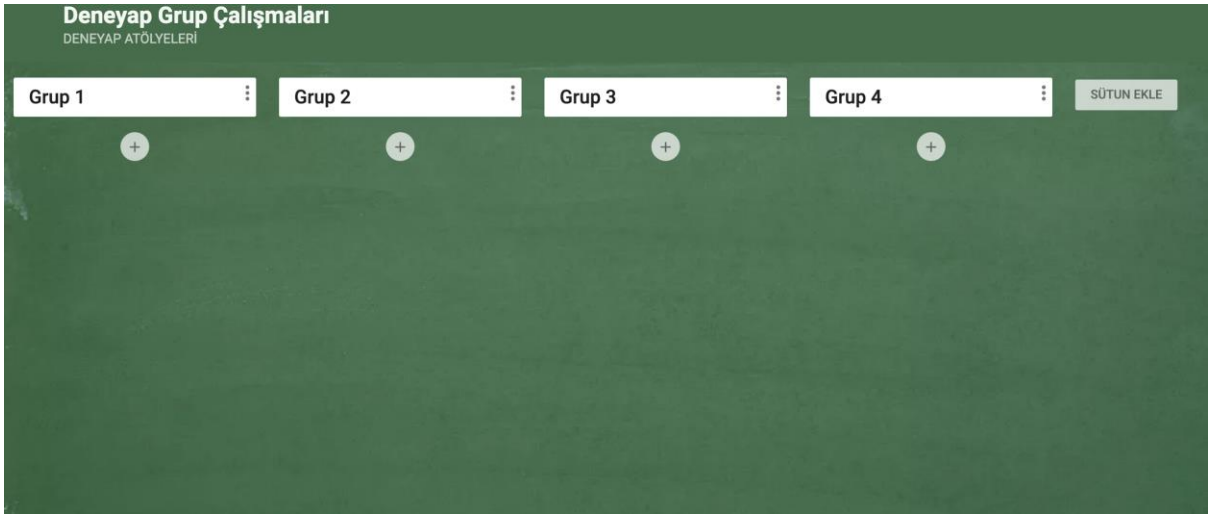
GitHub: Derste aktif şekilde kullanılacak olan arşivleme profolyo oluşturma ortamlarından biri Github'tır. GitHub açık kaynaklı projeler tarafından tercih edilen ve yazılım geliştirme projeleri için kullanılan web tabanlı popüler bir depolama servisidir. GitHub ile dünya çapında herkes tarafından görüntülenebilen 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 işbirliğini destekleyen 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 4'te örnek bir padlet raf stilinin görselini inceleyebilirsiniz.



Resim 4. *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.

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.*

Gulikers, J. T., Bastiaens, T. J., & Kirschner, P. A. (2004). *A five-dimensional framework for authentic assessment. Educational Technology Research and Development, 52(3), 67-86. https://doi.org/10.1007/bf02504676*

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does Gamification Work? - A Literature Review of Empirical Studies on Gamification. 2014 47th Hawaii International Conference on System Sciences, 3025-3034. https://doi.org/10.1109/HICSS.2014.377

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.

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

https://dergipark.org.tr/tr/pub/jgedc/issue/38681/449375

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. Yüz Yüze: Programlamaya Giriş ve Algoritma Tasarımı

Kazanımlar

- K1. Temel programlama terimlerini açıklar.
- K2. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.
- K3. Değişkenlere farklı türlerde değer ataması yapar.
- K4. Verilen algoritmanın temel özelliklerini analiz eder.
- K5. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını oluşturur.
- K6. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını bilgisayar ortamında çizer.
- K7. İhtiyaç duyulan algoritmayı tasarlar.
- K8. Bir problemin çözümünde değişkenleri kullanır.
- K9. Verilen algoritmanın akış diyagramını sözde koda dönüştürür.

Amaç

Bu haftanın amacı bir problemi algoritma ve akış diyagramları kullanarak çözmek ve problemin çözümünü bilgisayar kullanarak aktarmaktır. Programlama dünyasına giriş yapılarak, öğrencilerin temel düzeyde matematiksel, karşılaştırma ve mantıksal işleçleri (operatörleri) kullanmalarını sağlamak hedeflenmektedir. Ayrıca bir problemin çözümü için tasarlanan akış diyagramlarında kullanılan değişkenler ve değişkenlerin değerlerini keşfetmesi beklenir.

Önerilen Ders Akışı

A. Tanışma: Çember Oyunu (15 dk.)

B. Öğrenme Görevleri

B1. Nasıl Anlatırsın? (15 dk.)

B2. Beni Bul ve Değiştir (25 dk.)

Ders Arası (10 dk.)

B3. Farklı Atama Türlerini Tanıyalım (15 dk.)

B4. Algoritmayı Tanıyalım ve Eşleştirelim (20 dk.)

B5. Algoritma Yazıyorum (15 dk.)

Ders Arası (10 dk.)

B6. Çıkartma Algoritması (25 dk.)

B7. Otomatik Park Etme (25 dk.)

Ders Arası (10 dk.)

B8. Değişkenlerin Önemi (25 dk.)

B9. Algoritmayı Test Edelim (25 dk.)

A. Tanışma: Çember Oyunu

Süre: 10 dk.

Uygulama: Eğitmen öğrencileri 1 ve 2 olarak gruplar. 1 numaralı öğrenciler dış çember, 2 numaralılar iç çember olarak adlandırılır ve sınıfta iç içe iki çember şeklinde oturmaları sağlanır. Öğrenciler birbirlerine bakacak şekilde oturmaktadır. Eğitmen, öğrencilerden birtakım soruları birbirlerine sorup, bununla ilgili sohbet etmelerini ister. İlk olarak 1 numaralı dış çember öğrencileri, bir dakika süre ile 2 numaralı iç çembere cevap verecektir. Daha sonra 2 numaralılar soru soracaktır. Bir soruya iki taraf da cevap verdikten sonra yeni soruya geçilir. Her yeni soruya geçişte iç çember saat yönünde döndürülerek eşlerin yer değiştirmesi sağlanır. Eşlerin birbirlerine sorabilecekleri örnek sorulardan bazıları aşağıdadır. Bu sorular eğitmen tarafından değiştirilebilir, azaltılabilir ya da artırılabilir.

- Benim için mükemmel bir gün planı oluştursaydınız içinde neler olurdu?
- Bir robot olsaydın, ne işe yarardın?
- On yıl sonra bugün nerede olacağını düşünüyorsun?
- En komik tanıdığın kişi kim ve neden? Seni nasıl güldürüyor?

B. Öğrenme Görevleri

B1. Nasıl Anlatırsın?

Süre: 15 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 tahtada yazılarak, programlama ve programlama dili kavramları beyin fırtınası yoluyla özetlenir. Sonuç olarak öğrencilerin aşağıdaki içeriğe ulaşmaları sağlanır.

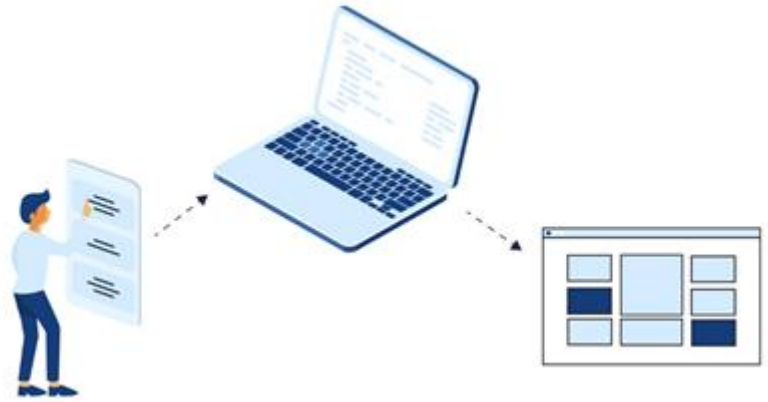
Konu içeriği:

Tablo 1. *Programlama ve Programlama Dili Kavramları*

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. Programlama, bir bilgisayara belirli görevleri yerine getirmesi için açık, mantıklı ve adım adım talimatlar vermek sürecidir. Bilgisayarlar çok hızlı işlem yapabilen güçlü araçlardır ancak kendi başlarına düşünemezler ve ne yapacaklarını bilemezler. Bu yüzden, bir insan tarafından yönlendirilmelidirler. Programlama, bilgisayara ne yapacağını söyleyen adımların (algoritmaların) yazılmasıdır. Bu talimatlar Python, C, Java gibi özel dillerle yazılır. Yazılan programlar, bilgisayarın anlayabileceği dile yani makine koduna dönüştürülerek çalıştırılır. Bilgisayara verilen talimatlar açık ve kesin olmalıdır. Bilgisayarlar yorum yapamaz; sadece verilen komutları uygular.

Bilgisayarlar kendi başlarına karar veremez. Hangi veriyi nasıl işleyeceklerini bilemez. Belirsiz veya eksik tanımlanmış görevleri yerine getiremez. Bu yüzden, onlara ne yapacaklarını detaylı şekilde söylemek gerekir. Bu tarif de programlama yoluyla gerçekleştirilir.



Resim 5. *Programlama süreci*

Tıpkı insanların Türkçe, İngilizce, Fransızca gibi farklı doğal dilleri konuşması gibi, bilgisayarlar da farklı programlama dillerini "konuşur." Python, C++, Java gibi diller bu kapsamdadır. Ancak burada önemli bir fark vardır:

Doğal Diller (İnsan Dilleri): Esnek, bağlama göre anlam kazanabilir. Küçük hatalar çoğu zaman anlamı bozmaz. Örneğin: "Yarın hava yağmurlu olabilir." cümlesi farklı biçimlerde de anlaşılır.

Programlama Dilleri (Bilgisayar Dilleri): Katı ve kurallıdır. En küçük yazım veya sözdizimi hatası (örneğin noktalı virgülün unutulması) kodun çalışmasını engeller. Bilgisayarlar yoruma yer bırakmaz; yalnızca açık, kesin ve doğru yazılmış komutları yerine getirir. Programlama, bilgisayara şu üç sorunun cevabını adım adım anlatmaktır:

- Ne yapacağını (örneğin: toplama işlemi yap),
- Ne zaman yapacağını (örneğin: kullanıcı bir tuşa bastığında),
- Nasıl yapacağını (örneğin: iki sayıyı al, topla, sonucu göster).

Bu talimatlar genellikle üç tür işlem içerir:

Girdi: Kullanıcıdan veya başka bir kaynaktan veri alma işlemidir. Örnek: Klavyeden sayı girilmesi, bir dosyadan veri okunması.

İşlem: Alınan veriler üzerinde yapılan işlemlerdir. Örnek: Toplama, karşılaştırma, döngü çalıştırma.

Çıktı: İşlenen verilerin ekrana veya başka bir yere yansıtılmasıdır. Örnek: Sonucun ekrana yazdırılması, dosyaya kaydedilmesi.

Programlama sayesinde bilgisayarlara hem basit hem de karmaşık görevler yaptırmak mümkündür. Örneğin, sadece 5 + 3 işlemini hesaplamak gibi basit bir görev birkaç satırlık kodla yapılabilirken, bir hava tahmin sistemi, bir sosyal medya uygulaması ya da bir oyun motoru gibi karmaşık yapılar binlerce satır kod içerebilir ve çok daha kapsamlı bir planlama gerektirir. Programlama bu görevleri bilgisayara adım adım anlatma sürecidir; ne yapılacağını, ne zaman yapılacağını ve nasıl yapılacağını açık ve kesin biçimde tarif eder. İnsan dilleri bağlama göre esnek anlamlar taşıyabilirken, bilgisayar dilleri katıdır ve en küçük yazım hatası bile kodun çalışmasını engelleyebilir.

B2. Beni Bul ve Değiştir

Süre: 25 dk.

Kazanımlar: K2. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.

Materyaller: O1.B2.1. Temel İşlemler Tablosu

O1.B2.2. İşleç Bilgi Kartları

O1.B2.3. Örnek Olay Sunum Kartı

Hazırlık: İşleç bilgi kartları, ön yüz ile arka yüzü iki tarafı eşleşecek şekilde arkalı önlü çıkartılmalıdır. 4 grup için 4 adet çıktı alınır ve kesikli çizgiler aracılığıyla kartlar oluşturulur. Diğer iki materyalde (temel işlem tablosu ve örnek olay sunum kartı) tüm gruplara bir tane dağıtılacak şekilde dörder tane 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 kullanıldıklarında herhangi bir anlam ifade etmeyen ancak programın akışına katkıda bulunan sembol veya sembol topluluklarına işleç

denir. İşleçler, bilgisayara belirli bir işlemi gerçekleştirmesini söyler. Örneğin, toplama (+) ya da çıkarma (-) işleçleri bu türdendir. + sembolü tek başına anlamlı değildir; ancak iki sayı arasında kullanıldığında, bu sayıların toplanmasını sağlar. Aynı şekilde - sembolü de iki sayı arasında çıkarma işlemini gerçekleştirir. Bu işleçler, programın mantıksal yapısını kurmak ve işlemleri tanımlamak açısından temel bir rol oynar."

Bu kısa bilgiden sonra her gruba temel işlem tablosu ve işleç bilgi kartları karışık şekilde verilir. İşleç bilgi kartının ön yüzünde işleç ve örnek kullanımı, arka yüzünde ise açıklaması bulunmaktadır. Grupların ilk görevi işleç bilgi kartlarındaki işlemleri gruplayarak, temel işlem tablosunda doğru yere yazmaktır. İkinci görevde ise, öğrencilere bir örnek olay sunum kartı verilir. İki örnek olay kartı ve kartın arka yüzünde örnek olay görevleri verilmiştir. Onlardan sunum kartının arkasındaki görevleri 5 dk. süresince yapmaları ve grup kâğıtlarına yazmaları istenir. Burada öğrenciler örnek olayı inceleyecek ve görevler sırasında tamamladıkları temel işlem tablosunda yer alan bilgileri kullanacaklardır. Bir örnek olay üzerinde çalışan grup, diğer örnek olayı tamamlamak üzere kartını değiştirir. Görevlerin 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 oldukça yaygın olarak kullanılır. Programlama dilinde bir işleç, bir değer veya değişken üzerinde işlem yapan bir semboldür. İki sayıyı birbirine eklemek (3+5) gibi basit bir hesaplama yapabilen ya da x*x - 2*x + 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.

Tablo 2. *Matematiksel ifadeler ve bilgisayar kodlamaları*

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

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
/	Birinci işlenenin ikinci ile bölümünden elde edilen tam kısımdır (bölme işlemindeki bölüm).	x / y
%	Birinci işlenenin ikinci ile bölümünden elde edilen kalan kısımdır.	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.

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*

İşleç	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 işleci kullanılmalıdır.

Tablo 5. *Mantıksal işleçler*

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 "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

B3. Farklı Atama Türlerini Tanıyalım

Süre: 15 dk.

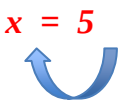
Kazanımlar: K3. Değişkenlere farklı türlerde değer ataması yapar.

Materyaller: O1.B3.1. Yer Değiştirme Görev Kartları

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 tabloda 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)

B4. Algoritmayı Tanıyalım ve Eşleştirelim

Süre: 20 dk.

Kazanımlar: K4. Verilen algoritmanın temel özelliklerini analiz eder.

Materyaller: O1.B4.1. Doğru Algoritma Hangisi?

O1.B4.2. Algoritmaları Eşleştirelim

Hazırlık: Ders materyali ÖYS üzerinden öğrenci erişimine açılır ya da çıktı alınır.

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, belirli bir problemi çözmek için takip edilmesi gereken sonlu sayıda adımdan oluşan bir çözüm yoludur. Algoritmalar, mevcut bilgilerden yola çıkarak istenilen sonuçlara ulaşmamızı sağlar. Bir algoritmayı daha iyi anlamanın en kolay yolu, onu bir tarif olarak düşünmektir; tıpkı yemek ya da ilaç tariflerinde olduğu gibi, belirli işlemler belirli sırayla yapılmalıdır. Günlük yaşantımızda da birçok işi, farkında olmadan algoritmik şekilde gerçekleştiririz; çünkü çoğu zaman bir işi tamamlamak için o işe ait alt işlemleri belirli bir sıra ile yapmamız gerekir. Bu yönüyle algoritmalar, hem programlamanın hem de mantıklı düşünmenin temelini oluşturur.*

1) **Tanımlayıcı:** Algoritmadaki değişkenleri, sabitleri, kayıt alanlarını ve özel bilgi tiplerini adlandırmak programcı tarafından gerçekleştirilen işlemlerdir. Tanımlayıcıların, yerini tuttukları ifadelere anlamca çağrışım yapacak şekilde adlandırılmaları, algoritmanın okunabilirliği ve anlaşılabilirliği açısından büyük önem taşır. Tanımlayıcı isimlerinde İngiliz alfabesindeki "A-Z" ve "a-z" harfleri,

"0-9" arasındaki rakamlar ve alt çizgi (_) sembolü kullanılabilir. Ancak tanımlayıcılar rakamla başlayamaz ve sadece rakamlardan oluşamaz.

2) Değişken: Algoritmanın her çalıştırılmasında, girilen değerleri alan veya programın çalışması sırasında bazı değerlerin atandığı bellek alanlarıdır. Değişkenler, yukarıda belirtilen tanımlayıcı kurallarına uygun olarak adlandırılmalıdır. Örneğin, bir ismin aktarıldığı değişken ad, bir isim ve soyismin aktarıldığı adSoyad, ev telefon numarasının saklandığı evTel, ev adresinin bulunduğu evAdres ve iş adresinin bulunduğu isAdres olarak tanımlanabilir. Burada özellikle İngiliz alfabesinin kullanılması gerektiğine dikkat edilmelidir.

3) Sabit: Uygulamanın çalışması boyunca değeri değişmeyen sabit veri ya da ifadeleri saklamak için kullanılır. Sabitler, algoritma boyunca kolayca takip edilebilmesi ve okunabilirliği artırmak amacıyla kullanılır. Sabitler de tıpkı değişkenler gibi tanımlayıcı kurallarına uygun şekilde adlandırılmalıdır.

4) Atama/Aktarma: Bir değişkene değer atamak için kullanılan işlemdir. Atama işleminde genellikle eşittir (=) sembolü kullanılır. Sağdaki değer, soldaki değişkene atanır. Örneğin sayi = 5 ifadesinde, 5 değeri sayi isimli değişkene atanır. Bu işlem algoritmanın temel yapı taşlarındandır.

5) Sayaç: Algoritma tasarımlarında belirli işlemlerin sayılmasını veya belirli sayıda tekrarlanmasını sağlamak için kullanılır. Özellikle döngülerde hangi adımda olunduğunu takip etmek veya kaç kez işlem yapıldığını kontrol etmek amacıyla sayaçlar kullanılır.

6) Döngü: Bir veya daha fazla işlem adımının, bir koşula bağlı olarak tekrarlanmasını sağlayan yapıdır. Döngüler, belirli sayıda veya bir koşul sağlandığı sürece işlem satırlarını tekrar tekrar çalıştırır. Örneğin, 1 ile 100 arasındaki çift sayıların toplamını hesaplamak için tek tek her değeri yazmak yerine, 2'şer artan bir döngü kurulabilir ve bu döngüde toplama işlemi gerçekleştirilir.

7) Ardışık Toplama/Çarpma: Algoritmalarda mevcut bir değere yeni bir değerin eklenmesi (toplama) ya da mevcut bir değerin yeni bir değerle çarpılması (çarpma) işlemlerine denir. Bu tür işlemler genellikle döngüler içinde yapılır. Örneğin, toplam = toplam + sayi şeklinde yapılan bir işlem ardışık toplama örneğidir. Benzer şekilde carpim = carpim * sayi ifadesi de ardışık çarpma işlemini ifade eder.

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

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 algoritmanın diğerinden farklı ve benzer olan yanları üzerine sorular sorabilir. Bunlar algoritmanın özellikleri için ipuçları verecektir.

Hazırlık: Nokta hâlinde oturan öğrencilerden yanlarındaki bir noktayla birleşmeleri ve dörtlü gruplar hâlinde oturmaları istenir. Çalışma kâğıdının her gruba ikişer tane dağıtmak üzere 8 adet çıktısı alınır.

Uygulama: Eğitmen dörtlü gruplar hâlinde oturan öğrencilerden "Algoritmaları Eşleştirelim" materyalini açmalarını ister. Materyalde iki 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 problemin çözüm adımları açık ve sade cümlelerle düz metin şeklinde yazılır. Her adım numaralandırılır ve algoritma genellikle "Başla" ile başlayıp "Bitir" ile sonlanır.

b) Sözde kod (pseudocode) yazım: Problemin çözüm adımları, komutlara benzeyen ve programlama mantığına uygun biçimde ifade edilir. "Eğer", "iken" gibi koşul kelimeleri ile '>', '<', '=' gibi operatörler kullanılır. Bu yöntem, konuşma dili ile programlama dili arasında bir köprü kurar ve doğru yazıldığında, sözde koddan gerçek bir programlama diline kolayca geçiş yapılabilir.

c) Akış diyagramlarının çizilmesi: Problemin çözüm adımları görsel olarak, çeşitli simgeler ve sembollerle ifade edilir. Bu şemalarda her işlem bir kutu içinde gösterilir ve adımlar arasındaki ilişki yönlendirme oklarıyla belirtilir. Akış diyagramları, algoritmanın görsel temsilini sunarak adımlar arasındaki akışı kolayca anlamamıza yardımcı olur.

B5. Algoritma Yazıyorum

Süre: 15 dk.

Kazanımlar: K5. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını oluşturur.

Materyaller: O1.B5.1. Nereye Gidelim Haritası

O1.B5.2. Görev Kartları

O1.B5.3. Akış Diyagramları Bilgi Afişi

Hazırlık: Materyallerden "Nereye Gidelim Haritası" ve "Bilgi Afişi" ÖYS'de öğrencilerin kullanımına açılır. Görev kartları ise iki adet çıktı alınarak kesilir ve tüm kartlar bir torba içine karışık olarak atılır. Öğrencilerin grup görevlerini paylaşması beş grup sütunundan oluşan raf temelli padlet linki öğrencilere ÖYS üzerinden paylaşılır.

Uygulama: Öğrenciler noktalar hâlinde ikişerli gruplara geri döner. Gruplar ÖYS'den ders materyali olan haritayı açar. Eğitmen her grubun torba içinden bir görev kartı seçmesini ister. Görev kartlarında haritada bir yerden diğerine ulaşmak için izlenmesi gereken talimatların algoritmasını sözde kod ile yazması istenir. Torbada bir görev iki kere tekrarlanmaktadır. Bu nedenle iki grup aynı problem üzerine çalışır. İkili gruplar sözde kodları oluşturduktan sonra, aynı problem üzerine gruplar bir araya gelip yeni grupları oluştururlar. Birleşen yeni gruplar artık dörder kişiden oluşmaktadır. Yeni grupların oluşturdukları sözde kodları akış diyagramlarının kullanarak akış şemasına dönüştürmeleri istenir. Eğitmen bu aşamada öğrencilerden "Akış diyagramları bilgi afişini" kullanmalarını ister ve afişe ÖYS'den açabileceklerini belirtir. Öğrencilerden etkinlik sonunda grup ürünlerinin ekran görüntülerini padlet ortamında paylaşmaları istenir. Paylaşımlar üzerinden eğitmen konuyu özetler.

Eğitmene Öneriler: Eğitmen bu etkinlikte akış diyagramlarına giriş yapmaktadır. App.diagrams.net uygulamasını kullanarak öğrencilerin akış diyagramlarını bilgisayar ortamında oluşturabileceklerini söyleyebilir. Etkinlik süresine bağlı olarak kısaca program öğrencilere tanıtılabilir. Programın kullanım kılavuzu ders öncesi öğrencilere ÖYS üzerinden gönderilebilir.

B6. Çıkartma Algoritması

Süre: 25 dk.

Kazanımlar: K6. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını bilgisayar ortamında çizer.

Materyaller: O1.B6.1. Çıkartma Algoritması Çalışma Kâğıdı

O1.B6.2. App.diagrams.net Program Kullanım Kılavuzu

Hazırlık: Eğitmen nokta oluşturan öğrencilerle ikişerli gruplar oluşturur. Çalışma kâğıdının 10 adet çıktısı alınır. Akış Diyagramları Bilgi Afişi ve program kullanım kılavuzu ÖYS'de öğrencilerin erişimine açılır. Öğrencilerin app.diagrams.net uygulamasını açmaları istenir. Öğretmen öğrenciler ile birlikte program kullanım kılavuzunu incelerler.

Uygulama: Eğitmen nokta oluşturan öğrencilerle ikişerli gruplar oluşturur. Eğitmen her gruba sözde kodlar hâlinde ancak karışık sırada yazılmış, iki sayı arasındaki çıkartma algoritmasının çıktısı verilir. Gruplar sırasıyla aşağıdaki görevleri tamamlamalıdır.

1. Sözde kod hâlinde verilen çıkartma 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 "app.diagrams.net" uygulamasını kullanarak algoritmayı çizin. Bu aşamada app.diagrams.net uygulaması kullanım kılavuzundan yararlanın.

Eğitmene Öneriler: App.diagrams.net uygulamasında akış şemaları çizim özelliklerinden öğrencilere kısaca bahsedilebilir. Program kılavuzunu ders öncesinde inceleyenler, incelemeyenler ile gruplanabilir.

B7. Otomatik Park Etme

Süre: 25 dk.

Kazanımlar: K7. İhtiyaç duyulan algoritmayı tasarlar.

Materyaller: O1.B7.1. Otomatik Park Etme Algoritması

Hazırlık: Eğitmen iki nokta oluşturan öğrencileri birleştirerek dörderli gruplar oluşturur.

Materyalin beş adet çıktısı alınır.

Uygulama: Öğrenciler dörderli gruplar hâlinde çalışır. Eğitmen öğrencilerden "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.

B8. Değişkenlerin Önemi

Süre: 25 dk.

Kazanımlar: K8. Bir problemin çözümünde değişkenleri kullanır.

Materyaller: O1.B8.1. Örnek Kod Çalıştırma Tablosu

Hazırlık: Materyal 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 ayırt edin.
2. Bu değişkenleri kullanarak akış şemasını çizin ve sözde kodları oluşturup sıralayın.

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 ÖYS üzerinden açıp incelemelerini ister. Eğitmen değişkenler ve değerleri üzerine konuyu özetler. Bunu yaparken eğitmen her bir adımın açıklaması için aşağıdaki tabloyu kullanılır.

Adım	Açıklama
1) Başla	Programın Başlangıcı
2) 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.
3) 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.
4) Aksi takdirde	**Sayi1, Sayi2**'den büyük değilse
4.1) Yaz, Sayi2	Ekrana **Sayi2**'nin değerini yaz
5) 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"*

B9. Algoritmayı Test Edelim

Süre: 25 dk.

Kazanımlar: K9. Verilen algoritmanın akış diyagramını sözde koda dönüştürür.

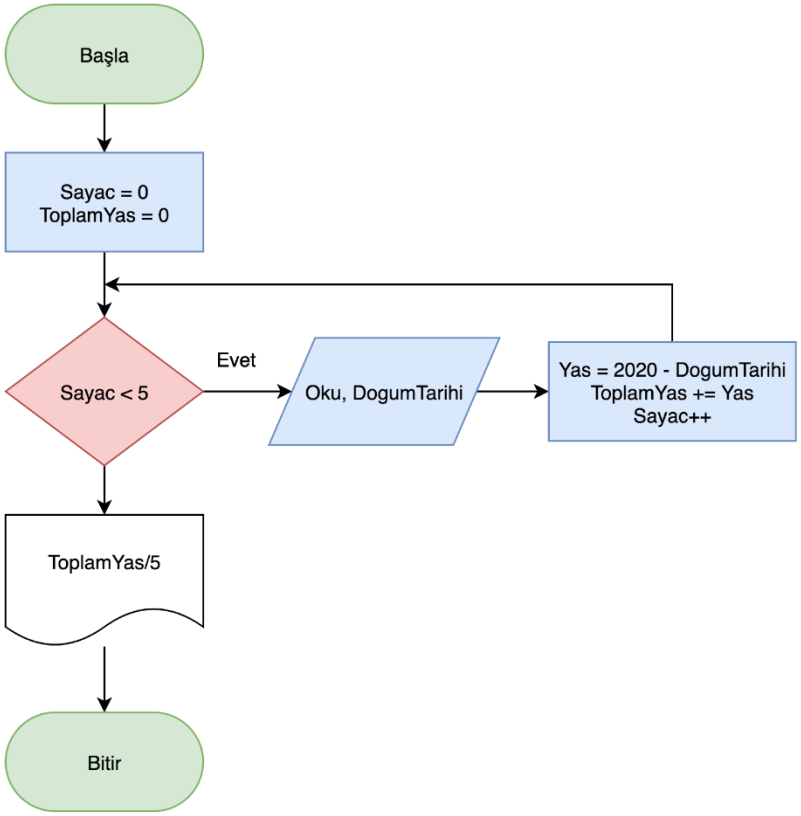
Materyaller: O1.B9.1. Sözde Kod Çalıştırma Tablosu

Hazırlık: App.diagrams.net uygulaması öğrenci bilgisayarlarında kurulu olmalıdır.

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 app.diagrams.net uygulamasını kullanarak bilgisayarda çiziniz.

Öğrencilerin çizmesi gereken akış şeması aşağıdaki gibi olmalıdır:



Eğitmen ikinci göreve geçmeden önce iki grubu birleştirerek dörderli yeni gruplar oluşturur ve aşağıdaki talimatı verir.

Görev 2: Çizdiğiniz akış şemasındaki sözde kodları karşılaştırın ve adım adım çalıştırıp, test etmek için Sözde Kod Çalıştırma tablosu oluşturun.

Öğrenciler bir önceki öğrenme görevi olan B4 etkinliğindekine benzer bir sözde kod çalıştırma tablosu hazırlayacaktır. Eğitmen bu materyalin aynısını kopyalayarak tekrar ÖYS üzerinden öğrencilerin erişimine açabilir. Hazırlanacak sözde kod çalıştırma tablosu aşağıdaki gibi olmalıdır:

Tablo 12. *Sözde kod çalıştma tablosu*

Adım	Değişken değerleri	Çıktı
1) Başla	Programın Başlangıcı	

Adım	Değişken değerleri	Çıktı
2) Sayac=0, ToplamYas=0	Sayac=0 ToplamYas=0	
3) Eğer 0<5	Doğru	
3.1) Oku, DogumTarihi	DogumTarihi=2001	
3.2) Yas = 2020 - 2001	Yas = 19	
3.3) ToplamYas += Yas	ToplamYas = 19	
3.4) Sayac ++	Sayac=1	
4) Eğer 1 <5	Doğru	
4.1) Oku, DogumTarihi	Doğum Tarihi=2003	
4.2) Yas = 2020 - 2003	Yas = 17	
4.3) ToplamYas += Yas	ToplamYas = 36	
4.4) Sayac ++	Sayac=2	
5) Eğer 2<5	Doğru	
5.1) Oku, DogumTarihi	DogumTarihi=2005	
5.2) Yas = 2020 - 2005	Yas = 15	
5.3) ToplamYas += Yas	ToplamYas = 51	
5.4) Sayac ++	Sayac=3	
6) Eğer 3<5	Doğru	
6.1) Oku, DogumTarihi	DogumTarihi=2007	
6.2) Yas = 2020 - 2007	Yas = 13	
6.3) ToplamYas += Yas	ToplamYas = 64	
6.4) Sayac ++	Sayac=4	
7) Eğer 4<5	Doğru	
7.1) Oku, DogumTarihi	DogumTarihi=2004	
7.2) Yas = 2020 - 2004	Yas = 16	
7.3) ToplamYas += Yas	ToplamYas = 80	
7.4) Sayac ++	Sayac=5	
8) Eğer 5<5	Yanlış	
9) Yaz, ToplamYas/5	80/5	16
10) Bitir.	Programın sonu	

Öğrencilerin oluşturduğu tablolar tahtaya asılır ve eğitmen tarafından tabloda sıralanan adımlar aşağıdaki gibi açıklamalarla özetlenir.

Tablo 13. *Sözde kod çalıştma tablosu*

Adım	Açıklama
1) Başla	Programın Başlangıcı
2) 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.
3) Eğer Sayac<5	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.
4) Yaz, ToplamYas/5	Ekrana ortalama yaş değerini (Toplam/5) değerini yazdırıyoruz.
5) Bitir.	Programın sonu

Eğitmene Öneriler: Eğitmen açıklama sırasında değişkenler ve değerlerine vurgu yapmalıdır. Öğrencilerin grup tablolarını online oluşturup, Öğrenme Yönetim Sisteminde paylaşmaları istenebilir.

Hafta 1. Yüz Yüze: Ders Materyalleri

O1.B2.1. Temel İşlemler Tablosu

Aritmetik İşlemler	Karşılaştırmalı İşlemler	Mantıksal İşlemler
Programlama dilinde iki ifade arasında toplama, çıkarma, bölme, yüzdesini alma gibi matematiksel işlemler için kullanılan sembollerdir.	*Programlama dilinde aynı türdeki değişkenler arasında belli bir koşula göre karşılaştırma yaptıran sembollerdir. Koşul doğru ise 1, yanlış ise 0 olarak program sonuçlanır.*	*Programlama dilinde iki ya da daha fazla koşulun birlikte değerlendirilmesinde kullanılan sembollerdir.*

O1.B2.2. İşleç Bilgi Kartları

Not: İşleç bilgi kartları önlü arkalı olarak tasarlanmış olup, bu şekilde çıktı alınmalıdır. Ön yüz işleç hakkında açıklama verirken arka yüz kullanım örneğini oluşturur.

Ön Yüz	Arka Yüz
İki işleneni toplar. İşleç: a **+** d	Arda'nın yaşı (a=5) ve Duru'nun yaşını (d=4) toplayalım. Çıktı: 5 + 4 = 19
İkinci işleneni birinciden çıkarır. İşleç: a **-** d	Arda'nın yaşından (a=5) ve Duru'nun yaşını (d=4) çıkaralım. Çıktı: 5 - 4 = 1
İki işleneni çarpar. İşleç: a ***** d	Arda'nın yaşı (a=5) ve Duru'nun yaşını (d=4) çarpalım. Çıktı: 5 * 4 = 20
İkinci işlenen birinciye böler. İşleç: a **/** d	Arda'nın yaşını (a=5) ve Duru'nunkine (d=4) bölelim. Çıktı: 5 / 4 = 1
Bölme işleminden kalanı verir. İşleç: a % d	Arda'nın yaşını (a=5) ve Duru'nun yaşına (d=4) bölüp, kalanı yazalım. Çıktı: 5 % 4 = 1

Sayıyı bir arttırma İşleç: a++	Arda'nın yaşını (a=5) bir arttıralım. Çıktı: 5++ = 6
Sayıyı bir azaltma İşleç: a--	Duru'nun yaşını (d=4) bir azaltalım. Çıktı: 4-- = 3
Büyüktür. İşleç: a > d	Arda'nın yaşı (a=5), Duru'nun (d=4) yaşından büyüktür. Çıktı: 1 (Doğru)
Küçüktür. İşleç: <	Arda'nın yaşı (a=5), Duru'nun (d=4) yaşından küçüktür. a < d Çıktı: 0 (Yanlış)
Küçük eşittir İşleç: <=	Arda'nın yaşı (a=5), Duru'nun (d=4) yaşından küçük eşittir. a <= d Çıktı: 0 (Yanlış)
Büyük eşittir İşleç: >=	Arda'nın yaşı (a=5), Duru'nun (d=4) yaşından büyük eşittir. a >= d Çıktı: 1 (Doğru)

Eşittir İşleç: ==	Arda'nın (a=5) ve Duru'nun (d=4) yaşı eşittir. a == d Çıktı: 0 (Yanlış)
Eşit değildir. İşleç: !=	Arda'nın (a=5) ve Duru'nun (d=4) yaşı eşit değildir. a != d Çıktı: 1 (Doğru)
VE İşleç: &&	Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. ((a==5) && (d>5)) Çıktı: 1 && 0 = 0 (Yanlış) ((a!=5) && (d>5)) Çıktı: 0 && 0 = 0 (Yanlış) ((a==5) && (d<5)) Çıktı: 1 && 1 = 1 (Doğru)
VEYA İşleç: ||	Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. ((a==5) || (d>5)) Çıktı: 1 || 0 = 1 (Doğru) ((a!=5) || (d>5)) Çıktı: 0 || 0 = 0 (Yanlış)

	((a==5) || (d<5)) Çıktı: 1 || 1 = 1 (Doğru)
DEĞİL İşleç: !	Arda'nın yaşı a=5 ve Duru'nun yaşı d=4'tür. ! (a==4) Çıktı: 1 (Doğru) ! (d!=5) Çıktı: 0 (Yanlış)

O1.B2.3. 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 **&&** Sınıf == 9 ise, öğrenci ismi yaz 1. Koşul 2. Koşul	GÖREV Örnek olaydaki koşul ifadelerini temel işlem tablosundaki semboller ile değiştirmeyi deneyin. Önceki sonuçlar ile yeni sonuçları karşılaştırın.

O1.B3.1. Yer Değiştirme Görev Kartları

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

Görevler	İpuçları
Görev Örnekteki = atama işlecinin ardından += işlecini uygulayın ve sonucu tahmin etmeye çalışın.	**İpucu** += atama işleci '+' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere ekler ve ardından sonucu soldaki değişkene atar.
Görev Örnekteki = atama işlecinin ardından, -= işlecini uygulayın ve sonucu tahmin etmeye çalışın.	**İpucu** '-' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerden çıkarır ve ardından sonucu soldaki değişkene atar.
Görev Örnekteki = atama işlecinin ardından, *= işlecini uygulayın ve sonucu tahmin etmeye çalışın.	**İpucu** '*' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerle çarpar ve ardından sonucu soldaki değişkene atar.

Görev Örnekteki = atama işlecinin ardından /= işlecini uygulayın ve sonucu tahmin etmeye çalışın.	**İpucu** '/' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere böler ve ardından sonucu soldaki değişkene atar.
Görev Örnekteki = atama işlecinin ardından %= işlecini uygulayın ve sonucu tahmin etmeye çalışın.	**İpucu** '%' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere göre modunu alır ve ardından sonucu soldaki değişkene atar.

O1.B4.1. Doğru Algoritma Hangisi?

1. Başla
2. Sebzeleri hazırla
3. Tencereye suyu koy
4. Ocağı aç
5. Suyun kaynamasını bekle
6. Sebzeleri tencereye koy
7. Baharatları ekle
8. 10 dakika bekle
9. Ocağı kapat
10. Bitir

A

1. Başla
2. Sebzeleri hazırla
3. Tencereye suyu koy
4. Ocağı aç ve suyun kaynamasını bekle
5. Sebzeleri tencereye koy
6. Baharatları ekle
7. 10 dakika bekle ve ocağı kapat
8. Bitir

B

1. Sebzeleri hazırla
2. Tencereye suyu koy
3. Ocağı aç
4. Suyun kaynamasını bekle
5. Sebzeleri tencereye koy
6. Baharatları ekle
7. 10 dakika bekle
8. Ocağı kapat

C

1. Sebzeleri hazırla
2. Tencereye suyu koy
3. Ocağı aç
4. Suyun kaynamasını bekle
5. Sebzeleri tencereye koy
6. Baharatları ekle
7. Dördüncü adıma geri dön
8. 10 dakika bekle
9. Ocağı kapat

D

O1.B4.2. Algoritmaları Eşleştirelim

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

P1. Çemberin alanını hesaplama

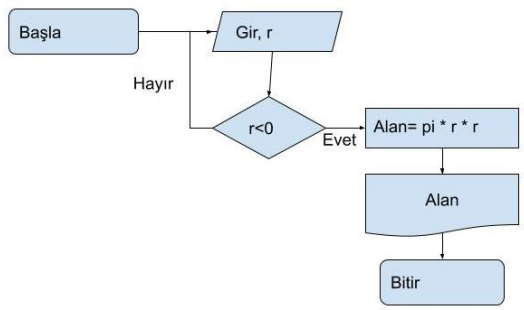
P2. İki sayıyı toplama

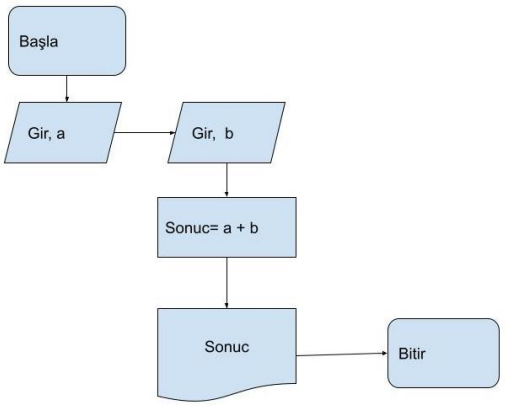
Algoritma İfade Şekli

MY: Metinsel Yazım

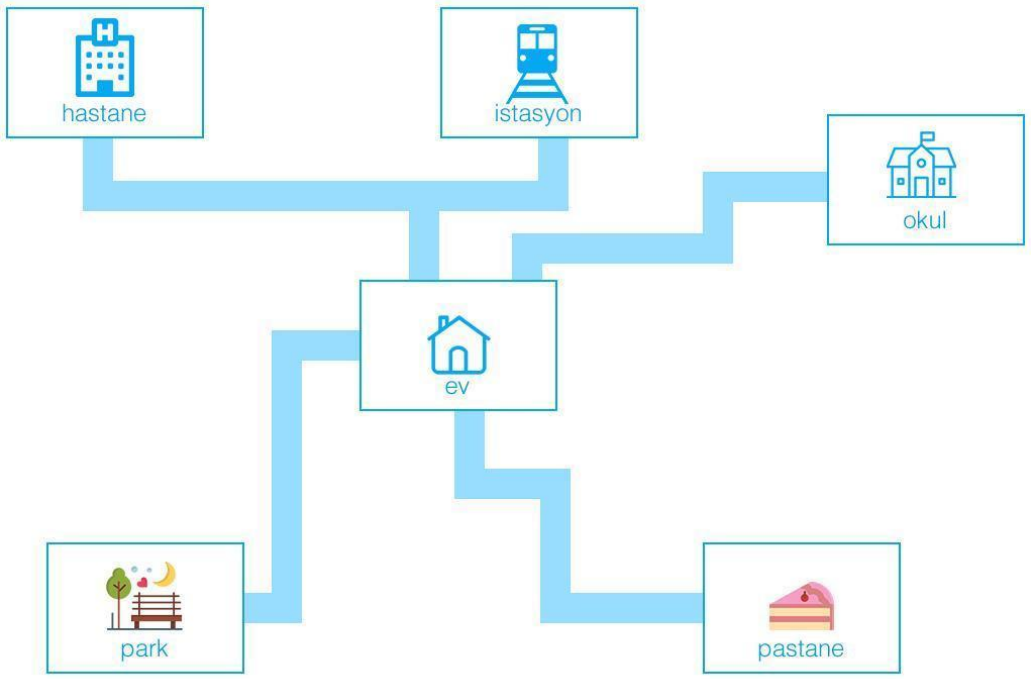
SK: Sözde Kod

AD: Akış Diyagramı

1. Başla 2. Birinci sayıyı gir (a) 3. İkinci sayıyı gir (b) 4. İşlemi yap. (sonuc = a+b) 5. Ekrana yaz (sonuc) 6. Bitir.	1. Başla. 2. Gir, r 3. Eğer r < 0 ikinci adıma git. 4. $alan = \pi \times r \times r$ 5. Yaz, Sonuc 6. Bitir.
Problem: *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*
	1. Başla. 2. Yarıçap gir (r). 3. Eğer yarıçap sıfırdan küçükse ikinci adıma git. 4. Alanı hesapla ($alan = \pi \times r^2$) 5. Sonucu ekrana yazdır. 6. Bitir.
Problem: *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*

1. Başla 2. Gir, a 3. Gir, b 4. Sonuc = a+b 5. Yaz, Sonuc 6. Bitir.	
Problem: *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*

O1.B5.1. Nereye Gidelim Haritası



Resim 6. *Nereye gidelim haritası*

O1.B5.2. Görev Kartları

Görev

Evinden çıkıp okula gitmek isteyen Arda'nın, izlemesi gereken yolu tarif eden bir algoritma oluştur.

Görev

Evinden çıkıp hastaneye gitmek isteyen Karaca'nın, izlemesi gereken yolu tarif eden bir algoritma oluştur.

Görev

Evinden çıkıp parka gitmek isteyen Defne'nin izlemesi gereken yolu tarif eden bir algoritma oluştur.

Görev

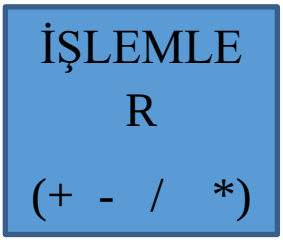
Evinden çıkıp pastaneye gitmek isteyen Güney'in, izlemesi gereken yolu tarif eden bir algoritma oluştur.

Görev

Hastaneden istasyona gitmek isteyen Karaca'nın izlemesi gereken yolu tarif eden bir algoritma oluştur.

O1.B5.3. Akış Diyagramları Bilgi Afişi

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

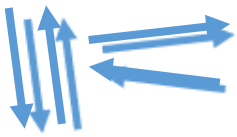


Toplama, çıkarma, aritmetik işlemlerin yapılması gerektiği durumlarda dikdörtgen kullanılır. Örneğin; üçgenin alanını hesaplama

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 şekilde bir "döngü" oluşur.



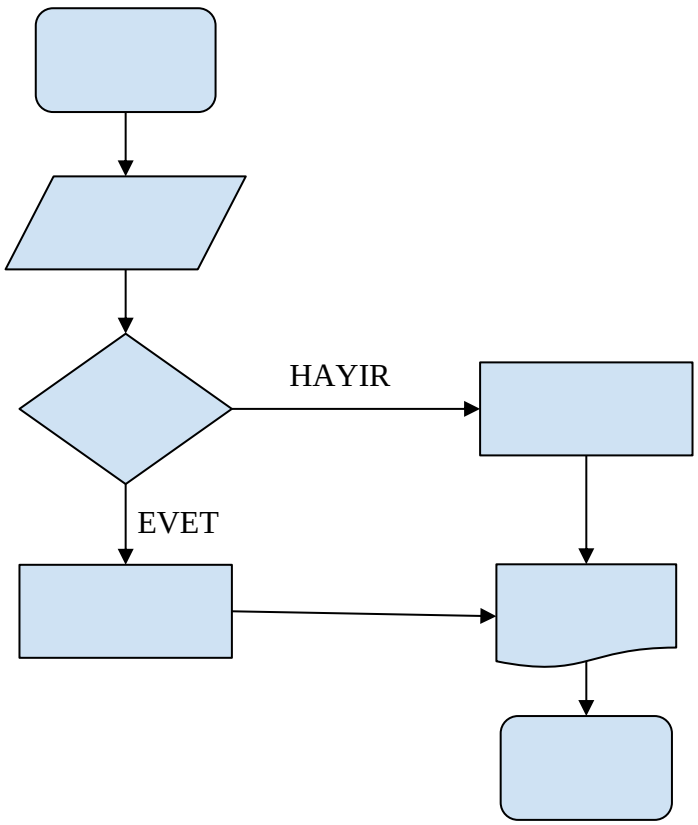
Yandaki oklar algoritmanın akış yönünü bize belirtir.

O1.B6.1. Çıkartma Algoritması Çalışma Kâğıdı

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

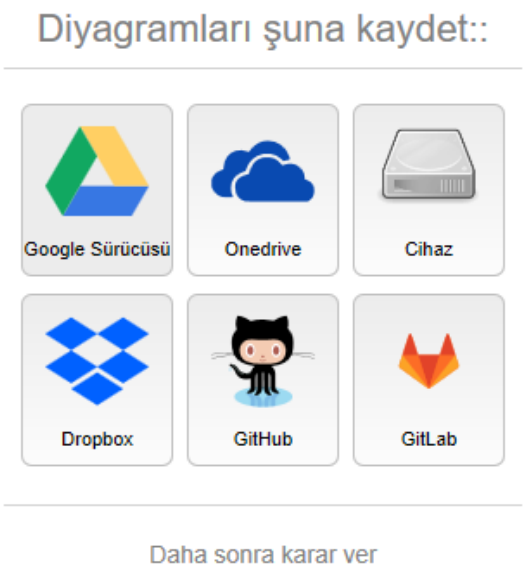
Problem: Klavyeden girilen iki sayı arasında küçük olanı büyük olandan çıkartma işlemini yazınız.

Başla	Değilse, Sonuc=Sayi2 - Sayi1	Gir, Sayi1, Sayi2	Eğer Sayi1 > Sayi2
Yaz, Sonuc	Sonuc=Sayi1 - Sayi2	Bitir	



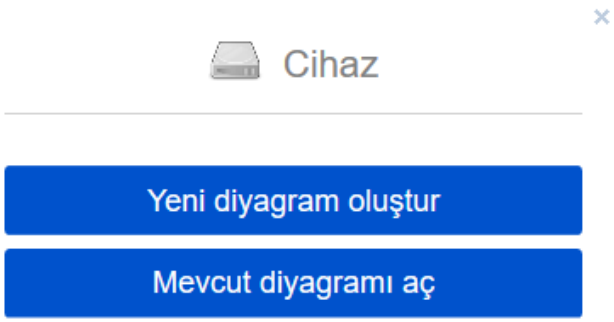
O1.B6.2. App.diagrams.net Program Kullanım Kılavuzu

Akış diyagramını bilgisayar ortamında çizmek için app.diagrams.net isimli websitesini kullanacağız. Ücretsiz olan bu uygulama sayesinde çizdiğimiz şekilleri kaydedebilir, istediğimiz kişilerle paylaşabiliriz. App.diagrams.net uygulamasına ilk giriş yaptığımızda aşağıdaki görüntü karşımıza gelecektir. Çizdiğimiz diyagramı nereye kaydetmek istediğimizi belirtiyoruz. Cihaz seçeneğini tıklayarak, şimdilik kendi bilgisayarımızda kaydedelim.



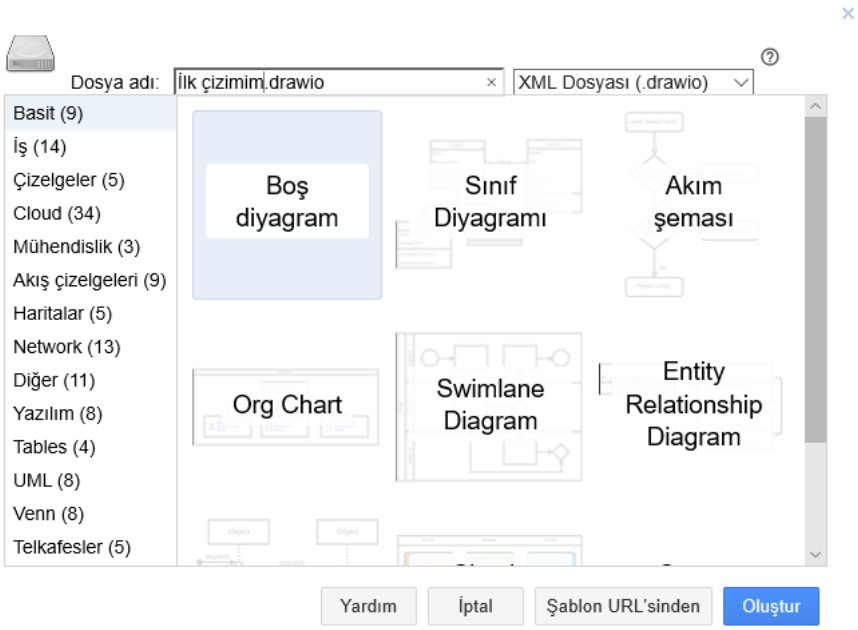
Resim 7. *App.diagrams.net giriş ekranı*

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



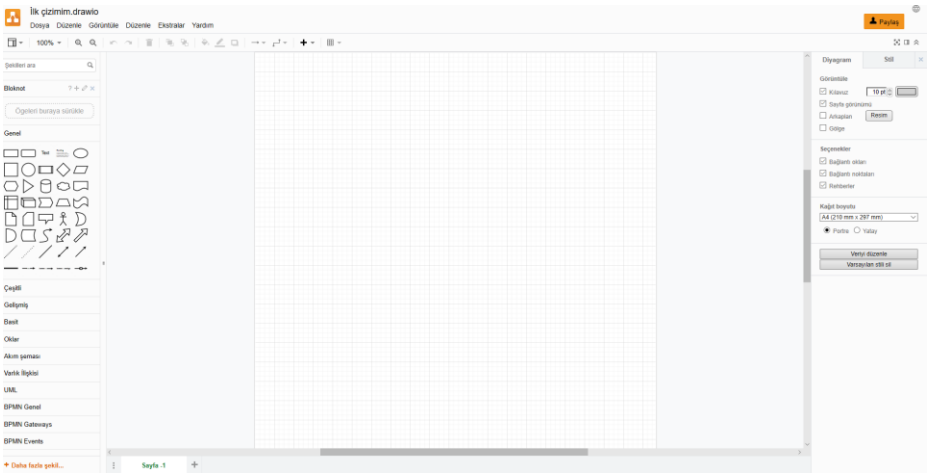
Resim 8. *App.diagrams.net yeni diyagram oluşturma ekranı*

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



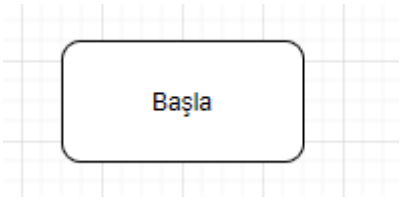
Resim 9. *App.diagrams.net yeni diyagram tipi seçme ekranı*

Karşımıza aşağıdaki çizim ekranı gelecektir.



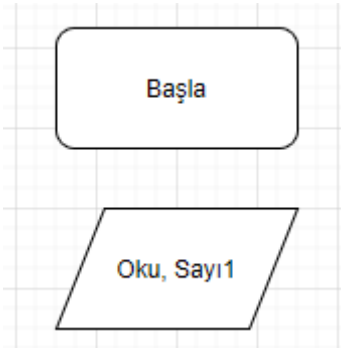
Resim 10. *App.diagrams.net çizim ekranı*

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.



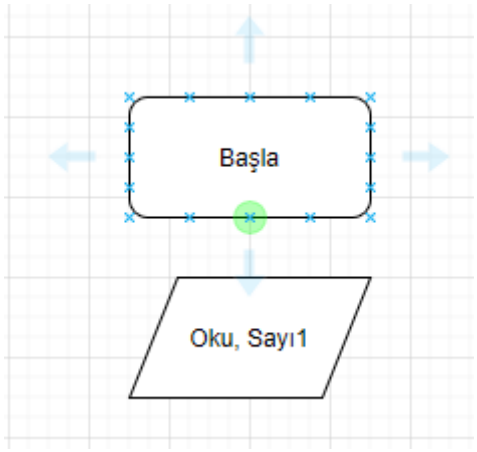
Resim 11. *Genel bloğu*

Ardından bir tane de paralel kenar ekleyelim.



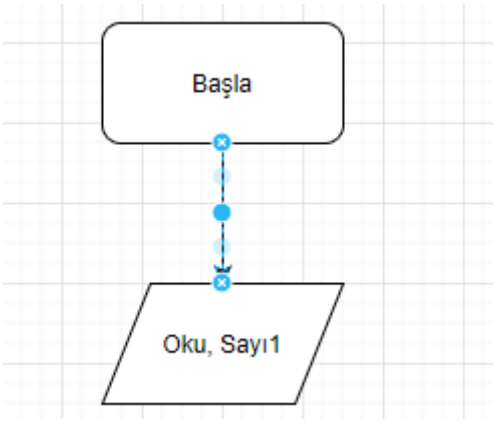
Resim 12. *Paralel kenar bloğu*

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



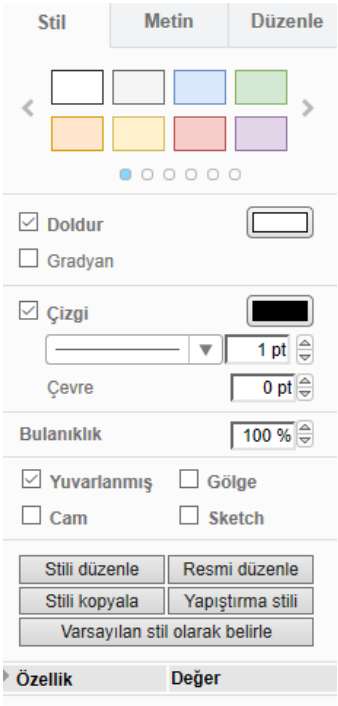
Resim 13. *Blokları bağlama öncesi*

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



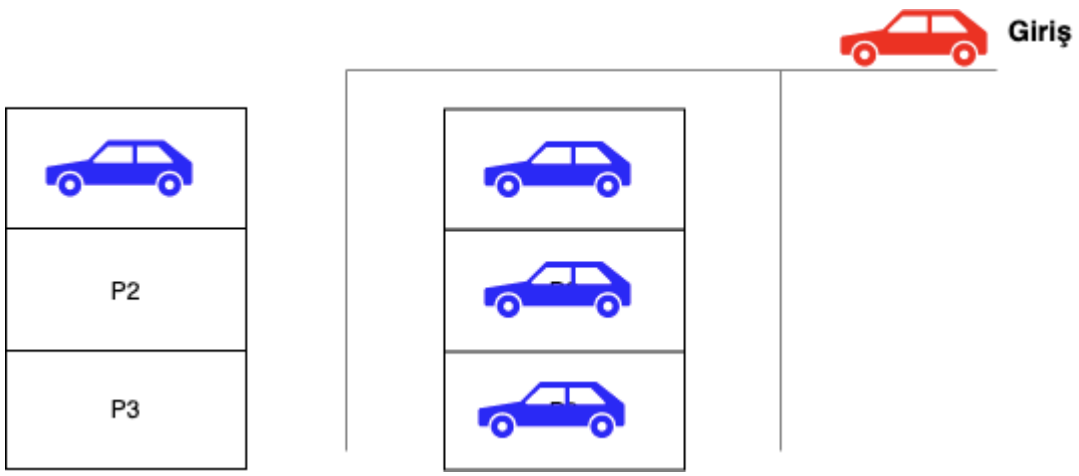
Resim 14. *Blokları bağlama sonrası*

Eğer bloğun renklerini değiştirmek istersek sağ paneli kullanabiliriz.

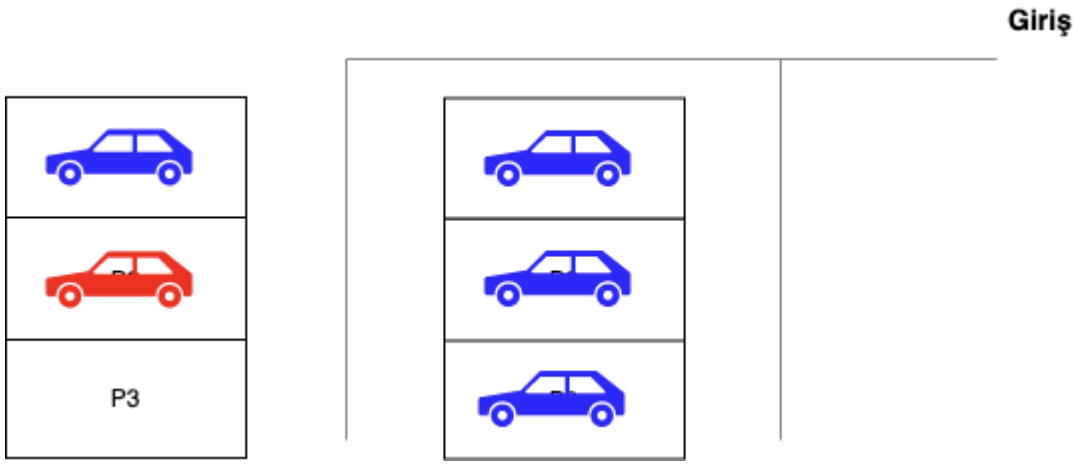


Resim 15. *Blok özellikleri ekranı*

O1.B7.1. 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.

O1.B8.1. 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 15. *Örnek kod çalıştma tablosu*

Adım	Değişken değerleri	Ekran Çıktısı
1) Başla.		
2) Oku, (Sayi1,Sayi2)	**Sayi1**=19, **Sayi2**=15	
3) Eğer Sayi1>Sayi2	Doğru, Evet	
3.1) Yaz, Sayi1		19
4) Aksi takdirde		
4.1) Yaz, Sayi2		
5) 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.

O1.B9.1. Sözde Kod Çalıştırma Tablosu

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

Tablo 16. *Sözde kod çalıştma tablosu*

Adım	Değişken değerleri	Ekran Çıktısı

Hafta 1. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftanın amacı, öğrencilerin bir problemi algoritma ve akış diyagramı kullanarak çözmelerini sağlamaktır. Temel programlama kavramlarını pekiştirme, akış diyagramlarını çizme ve ilk algoritmalarını bilgisayar ortamında ifade etmeleri hedeflenir.

Etkinlik Uygulama Ortamı:

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Aklına Ne Geliyor? (10 dk.)

B. Öğrenme Görevleri

B1. Akışı Bozanı Bulalım (20 dk.)

B2. Akış Diyagramı Çizelim (20 dk.)

Ders Arası (10 dk.)

B3. Ankete Katılalım (15 dk.)

B4. İlk Programın için IDE Kuralım (35 dk.)

A. Giriş: Aklına Ne Geliyor?

Süre: 10 dk.

Uygulama: Eğitmen tüm öğrencilerin derse katılımını beklerken eğlenceli bir müzik açarak öğrencilerini karşılar. Öğrencilerin derse girişi tamamlandıktan sonra, haftanın konu ve kazanımlarını paylaşarak öğrencileri hedeften haberdar eder. Daha sonra ön bilgileri hatırlatmak için Aklına Ne Geliyor? etkinliğini yürütür. Bu etkinlikte eğitmen öğrencilerden sanal sınıfta "sohbet" kısmını kullanmalarını ister. Eğitmen öğrencilere;

> *"Algoritma dediğimde aklınıza gelen ilk kelimeyi bana sohbet kısmından herkese açık olacak şekilde yazın."* der.

Tüm öğrenciler sohbet kısmına paylaşımlarını yazarken, bir dk. süre ile bir müzik açılır. Müzik sonunda, 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.*

B. Öğrenme Görevleri

B1. Akışı Bozanı Bulalım

Süre: 20 dk.

Materyaller: Dijital Tartışma Panosu: Grup Çalışması

OÇ1.B1.1. Grup Görevleri

Bknz. O1.B6.2. App.diagrams.net Kullanım Kılavuzu

Hazırlık: Eğitmen derse girmeden önce grup çalışması için raf temalı (sütunlardan oluşan) bir dijital pano oluşturur. Link ders sırasında öğrencilere paylaşılan notlar kısmından iletecektir. Eğitmen padlet linkinin gönderilmesi için sohbet ortamını da kullanabilir. Grup görevleri başlıklı OÇ1.B1.1. materyalindeki örnek görev ve grup görevleri sütunlar halinde öğrenciye görev sunumu için yüklenecektir.

Uygulama: Eğitmen OÇ1.B1.1 hazırladığı padlet materyalini Eğitmen app.diagrams.net arayüzünü öğrencilere açıklar ve programı gösterir. OÇ1.B1.1. de yer alan padlette ilk sütundaki örnek görevi ve akış diyagramını eğitmen açıklayarak app.diagrams.net de çizerek gösterir. Bu aşamada öğrencilerin izlemesi isteniz. 10 dk'yı geçmeyecek şekilde akış diyagramı şekilleri app.diagrams.net üzerinden tekrar edilir. Bu süreçte öğrencilere padlet ortamında başlık altında yer alan app.diagrams.net kullanım kılavuzunu açtırılır. Bu kılavuza erişim

linkini paylaşılan notlar kısmından da ayrıca gönderir. Kılavuz ÖYS ortamında sunumla birlikte paylaşılmalıdır.

> ***Örnek görev:*** Bir öğrencinin sınav puanına 5 puanlık ek performans notu eklenecek. Yeni puan 50'den büyükse "Geçti", değilse "Kaldı" yazan programın algoritmasını çiziniz.

```
Başla
Puanı Gir
YeniPuan = Puan + 5
YeniPuan > 50 ?
  Evet → "Geçti" yaz
  Hayır → "Kaldı" yaz
Bitir
```

Örnek görevin birlikte incelenmesinin ardından, grup görevleri hakkında eğitmen bilgi verir. Eksik gereksiz bir kod satırı içeren bu algoritmaları incelemeleri, akışı bozanı bulmaları için ilk grup etkinliğini başlatacağını ifade eder. Ardından eğitmen öğrencilere materyaldeki grup görevlerinin de yer aldığı padlet linkini paylaşılan notlar kısmından atar. Öğrenciler bu etkinlikte çalışma odalarında 5 dk. süre ile dört grup halinde tartışır ve padlette ilgili görev altına akışı bozan kod satırını padlette ilgili grup sütunu altına yazar. Süre sonunda öğrenciler ana odaya döner ve doğru yanıtlar için gönderiler birlikte kontrol edilir.

B2. Akış Diyagramı Çizelim

Süre: 20 dk.

Materyaller: Dijital Tartışma Panosu: Grup Çalışması

OÇ1.B1.1. Grup Görevleri

Bknz. O1.B6.2. App.diagrams.net Kullanım Kılavuzu

Uygulama: Öğrenciler akışı bozanı bulma etkinliğinden sonra tekrar gruplara ayrılacaktır. Eğitmen şimdiki görev için rastgele dört gruba ayrılacaklarını ve grup halinde oda numaraları ile aynı olan grup görevine ait algoritmanın akış diyagramını çizmelerini ister. Bu sefer 15 dk. süreleri vardır. İkinci görevde gruplar, bir önceki grubun göreve ilişkin yanıtını kontrol eder ve düzeltme gerekirse yeni yanıtlarını padlet ortamında yazarlar. Şimdi tüm grup üyelerinin yeni yanıtın üzerinden yeniden belirlenen görevin akış diyagramını app.diagrams.net kullanım kılavuzunu kullanarak çizmeleri istenir. Burada tüm öğrenciler ilgili görevin yeni halinin akış diyagramını kendi bilgisayarlarında çizecektir. Bir göreve ait grup üyelerinin her birinin akış diyagramını çizmesi ve padlette ilgili grup sütunu altında paylaşması istenir. Süre sonunda bir görevin altında tüm üyelerden gelen akış diyagramı ekran görüntüsü bulunmalıdır. Her bir öğrencinin çizim programını deneyimlemesi ve pekiştirmesi açısından bu aşama önemlidir.

Ana odaya dönen öğrenciler grup üyelerinin akış diyagramlarını inceler. Bunun için eğitmen 1 dk. süre ile müzik açar ve grup üyelerinin kendi grup görevine ait ekran görüntülerinden doğru olduğunu düşündükleri bir çizimi oylamalarını ister. Eğitmen her bir görevde en çok oy alan yanıt üzerinden konuyu özetler.

Bu etkinlik tamamlandıktan sonra 10 dk. ders arası verilir. Ders arası sohbetten ya da süre akışı ekranda yansıtılarak diğer oturumun başlama saati hakkında net bir yönlendirme yapılmalıdır.

B3. Ankete Katılalım

Süre: 15 dk.

Materyaller: OÇ1.B3.1. Anket Sorusu

Hazırlık: Bir önceki etkinlikte oluşturulan dijital tartışma panosu üzerinde yeni bir sütun açılır. Bu sütun ankete katıl etkinliklerini içerecektir. Grup Çalışması Padlet ortamında çoktan seçmeli sorudan oluşan bir anket açılır. Soruların oluşturulması için OÇ1.B3.1. materyali kullanılır.

Uygulama: Ders kapsamında algoritma ve akış şemaları konusunu işlerken, öğrencilerin koşul yapılarıyla ilgili düşünme becerilerini geliştirmek amacıyla mevcut Padlet panosuna yeni bir sütun eklenmiştir. Bu sütunda, öğrencilerden verilen akış şemasını incelemeleri ve soruya ilişkin çoktan seçmeli bir anketi cevaplamaları beklenmektedir. Bu aşamada eğitmen öğrencilere anket sorularını maksimum 1 dk. süre içinde cevaplama hakkı tanır. Eğitmen gelen anket yanıtları ile öğrenci performanslarını topladıktan sonra, seçimler üzerinden doğru yanıt hakkında açıklama yapar. Doğru algoritma çizimini padlette anket altına paylaşır.

Eğitmen anket sonunda öğrencilerin bu görevdeki akış diyagramını da ders sonunda app.diagrams.net ile çizim yaparak padlette anket altına iletmelerini ister. Bu şekilde öğrencilere ödev yerine, akış diyagramı tasarlama ve çizimini pekiştirme fırsatı verilir.

B4. İlk Programın için IDE Kuralım

Süre: 35 dk.

Materyaller: OÇ1.B4.1. C++ Program Tanıtımı *(Ek Materyaller Dosyasından Erişebilirsiniz)*

OÇ1.B4.2. Derleyici Kurulum Kılavuzu *(Ek Materyaller Dosyasından Erişebilirsiniz)*

Uygulama: Eğitmen C++ programlamanın tarihi, derleyici ihtiyacı ve seçimi ile ilgili bilgi edinebilecekleri OÇ1.B4.1 sunusunu kullanarak Code::Blocks isimli derleyici kurulumu öncesi bilgi verir. Ardından Bu sunumun sonunda yer alan OÇ1.B4.2 derleyici kurulum kılavuzunu takip ederek, bilgisayarlarına kurulum gerçekleştirmeleri için rehberlik eder. Her öğrencinin kurulumu tamamlaması için yükleme sırasında sıkıntı yaşayanlardan ekran paylaşımı yapmalarını isteyen eğitmen, tüm öğrencilerin derleyici kurulumunu tamamlandıklarından emin olur. Kurulum tamamlandıktan sonra derleyici arayüzü tanıtılır ve ilk programları için ekrana C++ dilinde aşağıdaki kodu yazdırır.

```
#include <iostream>  // Giris-cikis islemleri icin kutuphane
int main()
{
    std::cout << "Merhaba Dunya!" << std::endl;  // Ekrana Merhaba Dunya!
yazdir
return 0;  // Programin basarili sekilde bittigini belirt
}
```

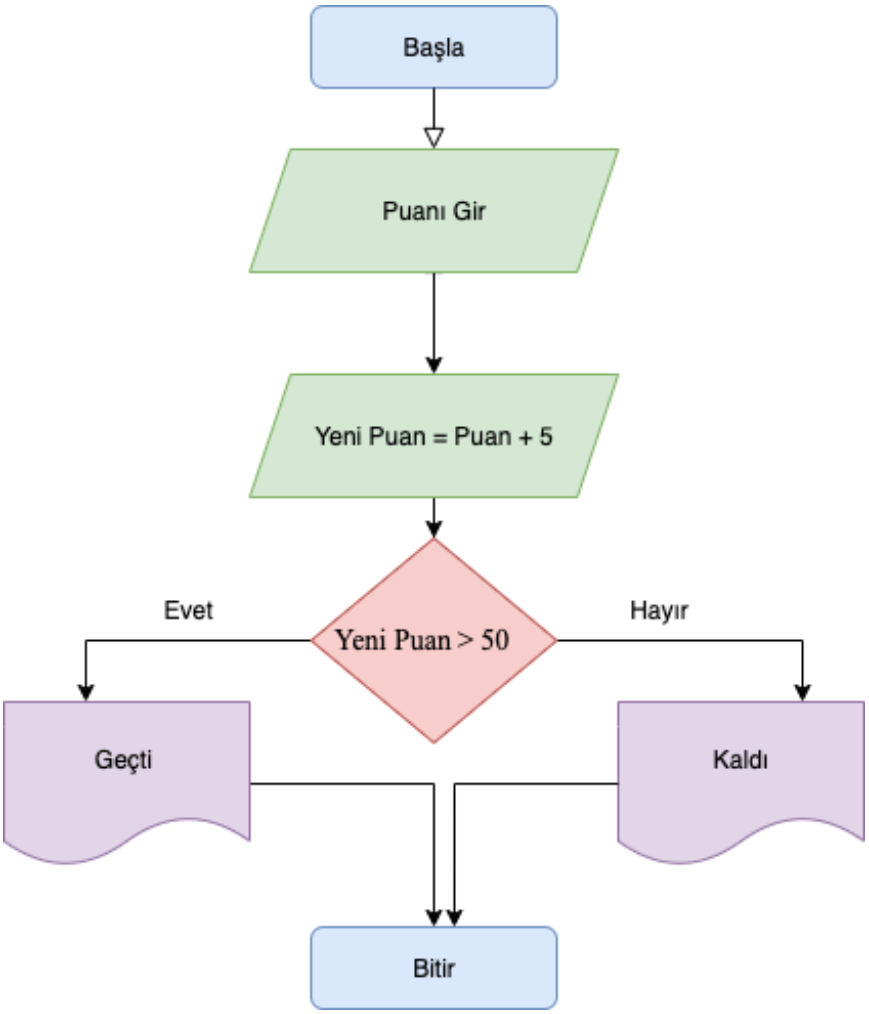
Gelecek hafta için öğrencilerin derleyiciyi bilgisayarlarında kurmuş olmaları ve ilk programlarını kaydetmeleri istenir.

Hafta 1. Çevrim İçi: Ders Materyalleri

OÇ1.B1.1. Grup Görevleri

Örnek Görev ve Akış Diyagramı Çizimi
Bir öğrencinin sınav puanına 5 puanlık ek performans notu eklenecek. Yeni puan 50'den büyükse "Geçti", değilse "Kaldı" yazan programın algoritması:

```
Başla
Puanı Gir
YeniPuan = Puan + 5
YeniPuan > 50
  Evet → "Geçti" yaz
  Hayır → "Kaldı" yaz
Bitir
```



Grup 1: Bir paket için temel kargo ücreti 10 TL. Eğer paketin ağırlığı 5 kg'dan fazlaysa 5 TL ekleniyor. Toplam ücreti yazdıran algoritma:

```
Başla
Ağırlık Gir
Ücret = 10
Ağırlık = 2 + 3  (Akışı Bozan Kod)
Ağırlık > 5
  Evet → Ücret = Ücret + 5
  Hayır →Ücreti Yazdır
Bitir
```

Grup 2: Bir müze girişinde 12 yaşından küçük olanlara indirim uygulanıyor. Giriş ücreti 30 TL, indirimli fiyat 15 TL. Yaşa göre uygun fiyatı yazdıran algoritma:

```
Başla
Yaşı Gir
Yaş = 13 (Akışı Bozan Kod)
Yaş < 12
  Evet → Ücret = 15
  Hayır → Ücret = 30
Ücreti Yazdır
Bitir
```

Grup 3: Bir öğrenciden rastgele bir sayı girilmesi isteniyor. Sayı üç basamaklıysa "Geçerli", değilse "Uygun değil" yazan algoritma:

```
Başla
Sayıyı Gir
Sayı == 100 (Akışı Bozan Kod)
Sayı >= 100 ve Sayı <= 999
  Evet → "Geçerli" yaz
  Hayır → "Uygun değil" yaz
Bitir
```

Grup 4: Öğrenci kantinden bir sandviç (15 TL) ve meyve suyu (10 TL) almak istiyor. Cebindeki para yeterli mi kontrol eden algoritma:

```
Başla
Parayı Gir
Toplam = 15 + 10
Toplam = 25 (Akışı bozan kod)
Para >= Toplam
  Evet → "Yeterli" yaz
  Hayır → "Yetersiz" yaz
Bitir
```

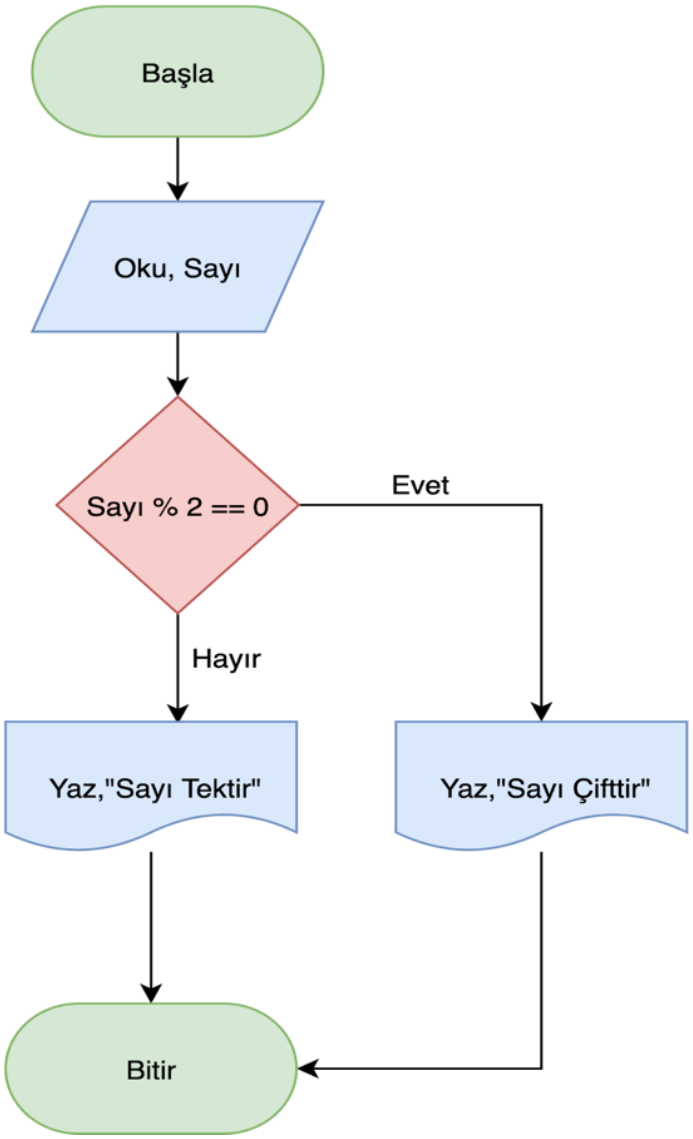
OÇ1.B3.1. Anket Sorusu

Aşağıdaki akış şeması, girilen bir sayının çift olup olmadığını belirlemektedir. Eğer bu algoritma, sayının 8 ile bölünüp bölünmediğini kontrol edecek şekilde değiştirilecek olsaydı, şemada hangi adım/adımlar değiştirilirdi?

***Birden fazla seçeneği işaretleyebilirsiniz.**

Seçenekler:

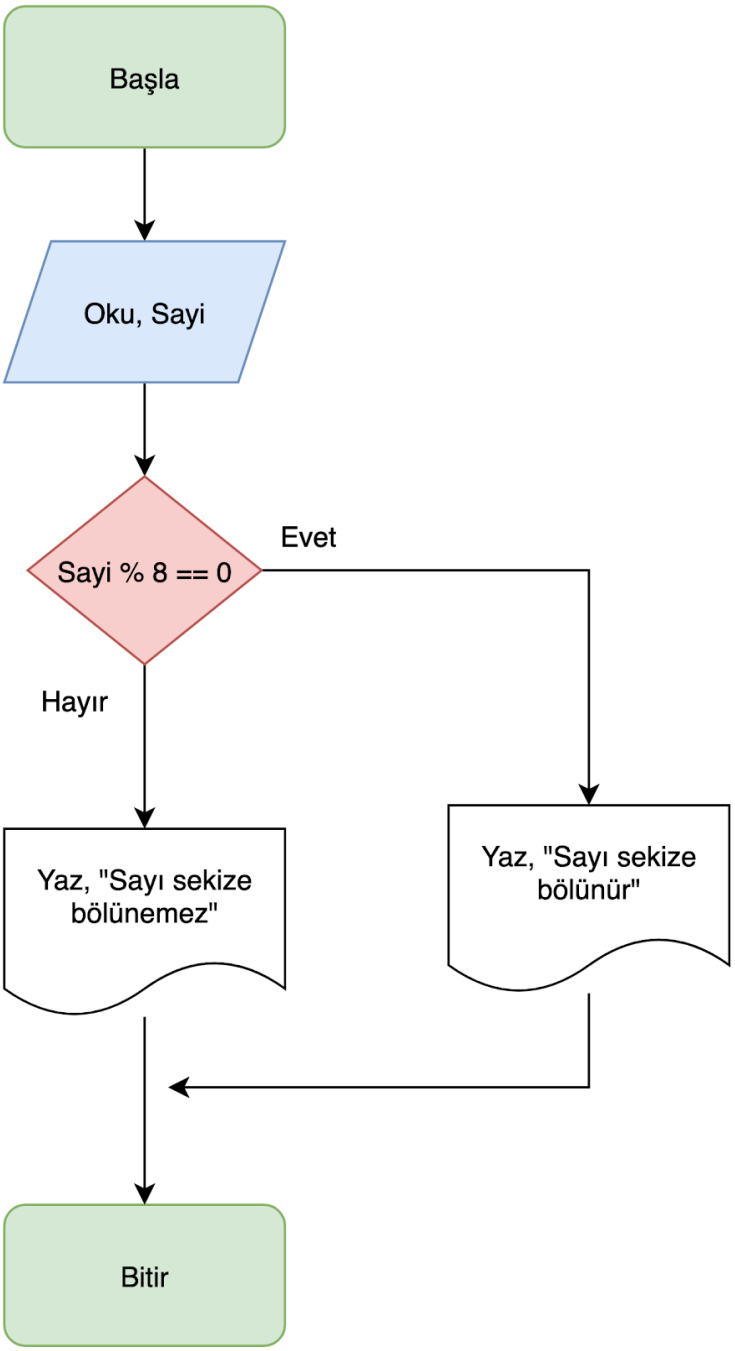
A) Karar kutusundaki ifade **"Sayı % 2 == 0"**, **"Sayı % 8 == 0"** olarak değiştirilirdi.
B) "Yaz, 'Sayı Çifttir'" kutusu yerine **"Yaz, 'Sayı 8'e Tam Bölünür'"** yazılırdı.
C) Karar kutusu **değişmeden kalır**, sadece yazı kutularında güncelleme yapılırdı.
D) Hiçbir değişiklik gerekmezdi, çünkü çiftlik kontrolü ile 8'e bölünme aynı anlama gelir.



Doğru Cevap(lar): A) ve B)

Açıklama:

- **A)** doğru çünkü sayı artık **2'ye değil, 8'e bölünüp bölünmediği** kontrol edilecek. Bu nedenle **karar kutusundaki matematiksel ifade** değişmelidir.
- **B)** doğru çünkü artık sayı çift olup olmadığından değil, **8'e bölünüp bölünmediğinden** bahsedilecek. Yazı kutularında da buna uygun bir güncelleme yapılmalıdır.
- **C)** yanlış çünkü yalnızca yazı kutularını değiştirmek, kontrol edilen koşulu değiştirmez. Sayının 8'e bölünüp bölünmediğini anlamak için karar kutusunun da değişmesi gerekir.
- **D)** yanlış çünkü **çift olmak** (2 ile tam bölünmek) ile **8 ile tam bölünmek** aynı şey değildir. Örneğin 10 çifttir ama 8'e tam bölünmez.


Başla
Oku, Sayi
Sayi % 8 == 0
Evet
Hayır
Yaz, "Sayı sekize bölünemez"
Yaz, "Sayı sekize bölünür"
Bitir

Hafta 2. Yüz Yüze: C++ Dilinde Değişken ve Veri Tipleri

Kazanımlar

- K1. C++ proramlama dilinde IDE üzerinde kod yazmaya başlar.
- K2. C++ programlama dilinde değişken tanımlamayı bilir.
- K3. C++ programlama dilinde kullanılan veri tiplerini bilir.
- K4. C++ programlama dilinde sabitleri tanımlamayı bilir.
- K5. C++ temel giriş/çıkış akışlarını bilir.
- K6. C++ işleçlerinin kullanımını bilir.

Amaç

Haftanın amacı, C++ programlama dilinde IDE üzerinde kod yazmaya başlamak ve değişken tanımlama adımlarının veri tipleri ile birlikte öğrenilerek sabit tanımlamanın nasıl gerçekleştirildiğini öğrenmektir. Ayrıca C++ temel giriş/çıkış akışlarının nasıl kullanıldığını, C++ işleçlerin örnekler üzerinde nasıl çalıştığını görmelerini sağlamaktadır.

Önerilen Ders Akışı

A. Giriş: İnmeyen Çubuk (15 dk.)

B. Öğrenme Görevleri

B1. İlk C++ Programım (35 dk.)

Ders Arası (10 dk.)

B2. Değişkenlere İsim Verelim (30 dk.)

B3. Veri Tiplerini Öğrenelim (20 dk.)

Ders Arası (10 dk.)

B3. Veri Tiplerini Öğrenelim (30 dk.)

B4. Sabitleri Ayıralım (20 dk.)

Ders Arası (10 dk.)

B5. Çıktıları Karşılaştıralım (15 dk.)

B6. Operatörlerle Yüzleşelim (35 dk.)

A. Giriş: İnmeyen Çubuk

Süre: 15 dk.

Materyaller: 2 adet ince ve ortalama 1–1,5 metre uzunluğunda çubuk (Oyuncu sayısına göre uzunluk değişebilir)

Uygulama: Oyuncular 2 gruba ayrılır ve birbirleriyle karşı karşıya iki sıra hâlinde hizalanırlar. Oyunculardan işaret parmaklarını kaldırmaları ve kollarını ileri doğru uzatmaları istenir. Çubuk yatay şekilde tutulur ve herkesin sadece işaret parmağıyla tutacağı şekilde yerleştirilir. Herkesin işaret parmağı çubuğa her zaman değmesi gerektiği ve diğer parmakların değmesinin yasak olduğu söylenir. Amaç; çubuğun tüm grupla birlikte, yere paralel olacak şekilde yere kadar indirilmesidir. Hangi grup çubuğu önce yere indirirse, o grup oyunu kazanır (Kaynak**:** https://app.ogrenmetasarimlari.com).

B. Öğrenme Görevleri

B1. İlk C++ Programım

Süre: 35 dk.

Kazanımlar: K1. C++ proramlama dilinde IDE üzerinde kod yazmaya başlar.

Materyaller: O2.B1.1. IDE Üzerinde Kod Yazımı

Hazırlık: Eğitmen IDE programını ve O2.B1.1. materyalini hali hazırda projeksiyondan yansıtacak şekilde açmalıdır.

Uygulama: Eğitmen, derse başlarken öğrencilere ilk haftanın çevrim içi dersinde kurup test ettikleri Code::Bloks IDE üzerinde yazdıkları ilk "Merhaba Dünya" kod satırlarını hatırlatır. "Bugün yazdığınız ilk programın bileşenlerini inceleyeceğiz. Böylece basit bir C++ kodunun her bir parçasının ne işe yaradığını tam olarak anlayacağız." diye devam eder.

Bu girişin ardından, "göster ve anlat" yöntemiyle öğrencilere Code::Blocks'ta yeni bir proje açtırır ve O2.B1.1 materyalini projeksiyonda yansıtır. Materyaldeki "Kod Açıklamaları" bölümünü kullanarak, main.cpp dosyasındaki her bir satırın (#include, int main, cout vb.) görevini detaylıca açıklar. Öğrenciler kodu çalıştırdıktan sonra, materyaldeki "Görev" bölümünü göstererek öğrencileri cout satırını kendi bilgileriyle düzenlemeye yönlendirir. Etkinliği, materyaldeki "Öğrenmeyi Derinleştirme Soruları" ile öğrencileri kod üzerinde denemeler yapmaya teşvik ederek ve sonuçları birlikte tartışarak tamamlar.

Eğitmene Öneriler: Eğitmen açıklamaları sırasında aşağıdaki bilgileri vurgulamalıdır.

C++ dili, C dili temel alınarak geliştirilmiş ve nesne yönelimli programlama özellikleri kazandırılmış güçlü bir programlama dilidir. Bir C++ programı yazarken belirli bir yapı izlenir:

1. Başlık Dosyaları: Başlık dosyaları, programda kullanılacak kütüphaneleri dahil etmek için kullanılır. Bunlar #include yönergesiyle eklenir:

```
#include <iostream>  // Giriş-çıkış işlemleri için
#include <cmath>     // Matematiksel işlemler için
```

2. Ad Alanları: C++'ta isim çakışmalarını önlemek için namespace kavramı kullanılır. Standart kütüphane bileşenleri std ad alanı içinde yer alır.

```
using namespace std;
```

Eğer using namespace std; yazmazsak, cout ve cin gibi ifadeleri std::cout ve std::cin şeklinde kullanmamız gerekir.

3. Ana Fonksiyon: Her C++ programının çalışmaya başladığı yer main() fonksiyonudur:

```
int main() {
    // Kodlar buraya yazılır
    return 0;
}
```

return 0; ifadesi, programın başarılı şekilde tamamlandığını gösterir.

4. Değişkenler ve Veri Tipleri: C++ dilinde değişkenler bir veri tipi ile tanımlanır. Örnekler:

```
int sayi = 5;        // Tam sayı
double pi = 3.14;    // Ondalıklı sayı
char harf = 'A';     // Tek karakter
```

5. Kontrol Yapıları: Koşullar ve döngülerle program akışı kontrol edilir:

```
if (sayi > 0) {
    cout << "Pozitif bir sayi!";
} else {
    cout << "Negatif bir sayi!";
}
```

6. Yorumlar: Kod içine açıklama eklemek için yorum satırları kullanılır:

```
// Bu bir tek satırlık yorumdur
/* Bu
   birden fazla
   satırdaki yorumdur */
```

Code::Blocks ile İlk C++ Projem

- Code::Blocks'u açın.
- Menüden File > New > Project yolunu izleyin.
- Console Application seçeneğini seçin.
- Dil Seçimi ve Proje Ayarı: Next tıklayın. C++ dilini seçin.
- Projenize bir isim verin. Örn: MerhabaDunya.
- Kodunuzu Yazın: main.cpp dosyası otomatik açılır ve kod yazılır.
- Derleme ve Çalıştırma: Üst menüden Build and Run butonuna tıklayarak programınızı çalıştırın.

B2. Değişkenlere İsim Verelim

Süre: 30 dk.

Kazanımlar: K2. C++ programlama dilinde değişken tanımlamayı bilir.

Materyaller: O2.B2.1. Değişkenler Çalışma Yaprağı

O2.B2.2. Hatalı Değişkenler

O2.B2.3. Değişken İsmi Olamazlar

Hazırlık: Eğitmen "Hatalı Değişkenler" materyalinin çıktısını alıp, tek tek keserek materyali beş gruba ayırır. Her grupta dörder tane kart bulunmalıdır. Diğerleri ise öğrenme yönetim sistemi üzerinden açılır.

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

"C++ programlarında değişken, bir verinin bilgisayarın belleğinde geçici olarak saklanmasını sağlayan, belirli bir isimle tanımlanan bir yapıdır. Değişkenler, program çalışırken verilerin tutulduğu "saklayıcı kutular" gibi düşünülebilir. Her değişken, kendisine ait bir isim (değişken adı), bir veri türü ve bellek adresi ile tanımlanır. Bu sayede programcı, veriye ulaşmak ya da veriyi değiştirmek istediğinde doğrudan değişkenin adını kullanabilir."

Resim 16. *Saklayıcı yapı kutu gösterimleri*

Eğitmen sunumun ardından bu etkinlikte değişkenlere isim verme üzerinde duracaklarını ifade eder. Bunun için öğrenciler dörderli gruplara ayrılır. Eğitmen tek tek kesilip beş gruba ayrılmış Örnek 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.

"Artık değişkenlere isim verme kuralları belli olduğuna göre, herkes kendisine bir kâğıt çıkarsın. Bu kuralların tümünü uygulayan bir veriyi tutacak değişken ismi belirleyin ve grup olarak verdiğiniz isimleri inceleyin."

Eğitmen etkinlik sonunda 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."

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

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. Ayrıca programlarınızın okunabilirliğini artırmak için, ogrenci_yas, sinav_notu vb. gibi daha uzun ve daha açıklayıcı adlar seçebilirsiniz.

B3. Veri Tiplerini Öğrenelim

Süre: 50 dk.

Kazanımlar: K3. C++ programlama dilinde kullanılan veri tiplerini bilir.

Materyaller: O2.B3.1. Veri Tipleri Çalışma Kâğıtları

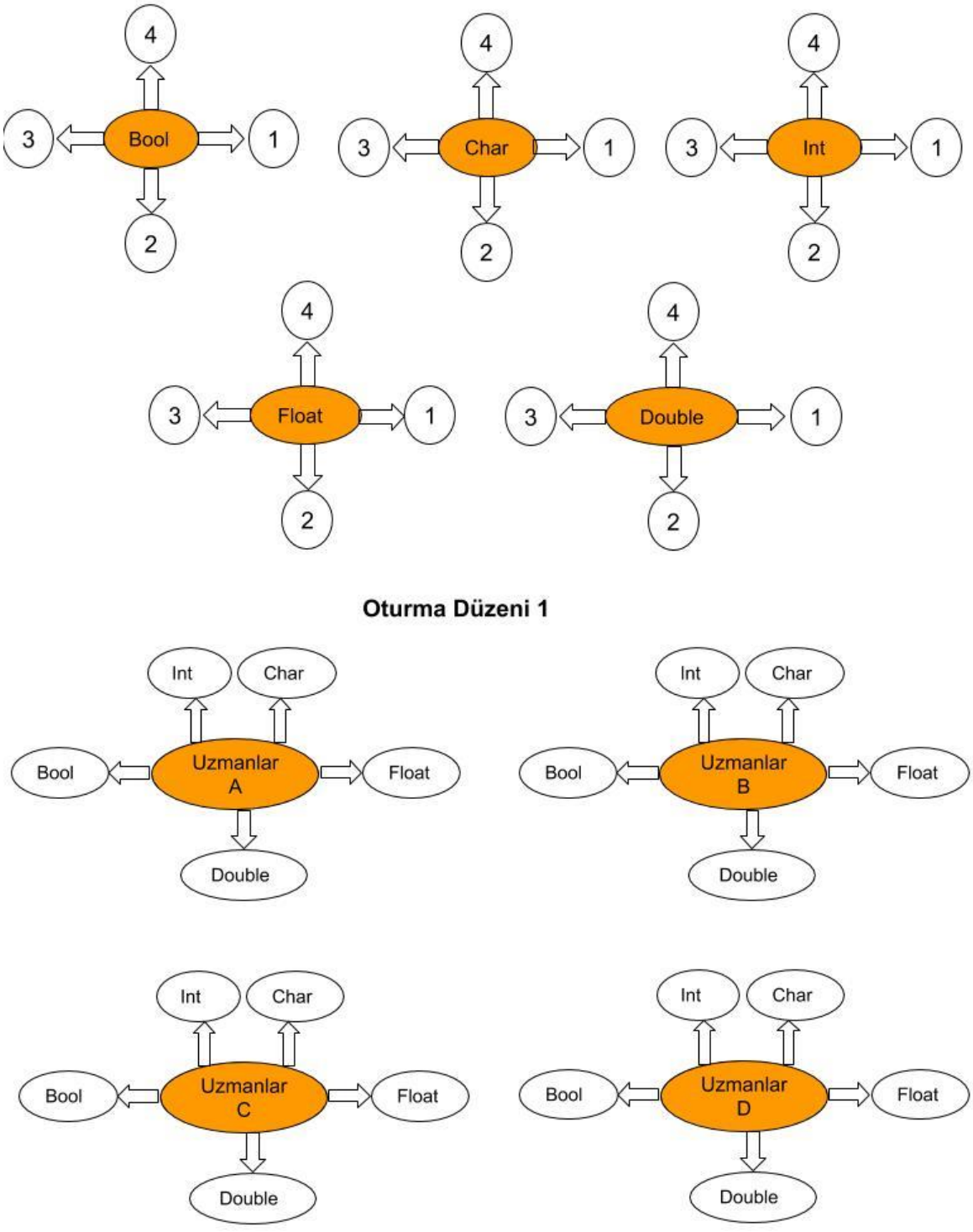
O2.B3.2. Değişken Atama Kodları

Hazırlık: Eğitmen veri tipleri çalışma kâğıtlarını ve değişken atama kodlarının birer çıktısını alır, bunları keserek 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. Ayrıca 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 bilgi kartlarını incelemeleri ve örnekleri grup içinde beş dakika içinde tartışmalarını ister. 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. Uzman gruplarına dağılmadan önce öğrencilerin grup içinde birlikte anlamlandıramadıkları noktaları eğitmenlere danışmaları istenebilir.



Resim 17. *Oturma düzenleri*

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. Sabitleri Ayıralım

Süre: 20 dk.

Kazanımlar: K4. C++ programlama dilinde sabitleri tanımlamayı bilir.

Materyaller: O2.B4.1. Sabitler İçin Örnek Kod

Hazırlık: Her grup için materyalin bir çıktısı alınır. 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 etmelerini ister. Bunu yaparken aşağıdaki üç temel kurala dikkat etmeleri gerektiğini belirtir:

1. "const" anahtar sözcüğünü kullanarak tanımlanır.
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:

> *Bir programın yürütülmesi sırasında içeriği hiç değişmeyecek olan veriler, bir değişken yerine sabit bir tanımlama ile saklanmalıdır. Bu, derleyicinin kodu hatalar açısından kontrol etmesini sağlar. Yazdığımız program, sabitte depolanan değeri değiştirmeye çalışırsa, derleyici bir hata bildirir ve derleme başarısız olur.*
>
> *Aşağıda verilen bilgileri sabit olarak değerlendirebilirsiniz.*
>
> - *Pi sayısı*
> - *Evinizdeki oda sayısı*
> - *Eviniz ile okulunuz arasındaki uzaklık*
> - *Kimlik numaranız*
> - *Öğrenci numaranız*
> - *Doğum tarihiniz*

```
const sabitTipi 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 tanımlasaydık ve daha sonra hassasiyeti artırmamız gerektiğinde sadece en baştaki tanımlamadaki değeri değiştirmemiz yeterli olacaktır.

B5. Çıktıları Karşılaştıralım

Süre: 15 dk.

Kazanımlar: K5. C++ temel giriş/çıkış akışlarını bilir.

Materyaller: O2.B5.1. Kod Çıktılarını Karşılaştıralım

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. 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.*

B6. Operatörlerle Yüzleşelim

Süre: 35 dk.

Kazanımlar: K6. C++ işleçlerinin kullanımını bilir.

Materyaller: O2.B6.1. Operatör Çalışma Yaprağı

O2.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.

Hafta 2. Yüz Yüze: Ders Materyalleri

O2.B1.1. IDE Üzerinde Kod Yazımı

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

int main() {
    cout << "Merhaba Deneyap!" << endl;
    return 0;
}
```

Kod Açıklamaları:

#include <iostream> Ekrana yazı yazmak için gerekli kütüphane

using namespace std; cout, cin gibi ifadeleri doğrudan kullanabilmek için

int main() Programın başladığı ana fonksiyon

cout << "Merhaba Deneyap!" << endl; Ekrana mesaj yazdırır

return 0; Programın başarıyla sona erdiğini belirtir

Görev: Ekrana kendi isminizi ve yaşınızı yazdıran bir C++ programı yazın.

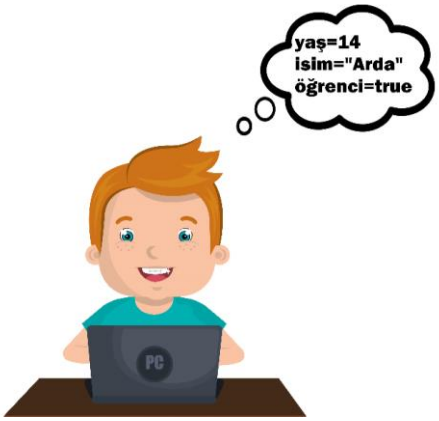
```
#include <iostream>
using namespace std;

int main() {
   cout << "Benim adim Arda, 12 yasindayim." << endl;
   return 0;
}
```

Öğrenmeyi Derinleştirme Soruları:

- using namespace std; satırını silersek ne olur?
- return 0; ifadesini yazmazsak ne olur?

O2.B2.1. Değişkenler Çalışma Yaprağı

C++ programlarında veri değerlerinin bilgisayarın belleğinde saklanabildiği saklayıcı yapılara değişken adı verilir. Saklanan değere değişkenin adı kullanılarak ulaşılabilir. Değişken dediğimiz kavram bir öğrencinin yaşı, ismi veya öğrencilik durumu olabilir.



Resim 18. *Değişkenleri düşünme*

Değişken kavramını daha iyi anlamak için şu örnekleri kullanabiliriz.

- *Bir bardak ve bir şişe süt olduğunu varsayalım. Bardağa biraz süt koyalım ve bardakta ne kadar süt olduğuna bakalım. Bardağa biraz daha süt koyalım ve tekrar bardakta ne kadar süt olduğuna bakalım. Sütün miktarı bardak dolana kadar az az değişerek herhangi bir değer olabilir. İşte bu nedenle süt miktarına "değişken" diyoruz. Yani değişebilen bir miktar söz konusudur.*

Resim 19. *Değişken kavramını anlama*

- *Kilomuz ve yaşımız bir değişkendir.*

Resim 20. *Kilo ve yaş değişkenleri*

- *Kumbaradaki para miktarımız bir değişkendir.*

Resim 21. *Kumbara değişkeni*

- *Günlük yürüme miktarımız bir değişkendir.*

Resim 22. *Günlük yürüme değişkeni*

- *Genel olarak farklı kişilerin adları ve soyadları değişkendir (tabii ki aynı olması da mümkündür).*

Resim 23. *Kimlik değişkeni*

Bir değişkenin bir program içerisinde kullanılabilmesi için, öncelikle tanımlanması gerekir. Değişken tanımlama işlemi, programcıya o değişkenin hangi türde veri tutacağını belirtme imkânı sunar. Tanımlama sırasında kullanılan veri türü, derleyicinin değişken için bellekte ne

kadar yer ayıracağını belirlemesini sağlar. Bu işlem sonucunda, bilgisayarın belleğinde o değişken için özel bir alan tahsis edilir.

Tanımlanan değişken, programın ilerleyen bölümlerinde adı kullanılarak erişilebilir hale gelir. Yani, bellekte ayrılmış olan bu alana ulaşmak ve içerisindeki değeri okumak veya değiştirmek için değişkenin adı bir referans olarak kullanılır.

```
veri-tipi degisken_adi;
```

Eğer aynı veri türünden birden fazla değişken tanımlanmak isteniyorsa, her biri ayrı ayrı satırlarda tanımlanabileceği gibi, daha kısa ve düzenli bir yazım için aynı satırda tanımlanıp aralarına virgül konularak da yazılabilir.

```
veri-tipi deg_adi1, deg_adi2, deg_adi3;
```

O2.B2.2. Hatalı Değişkenler

*Her satır bir grubu temsil etmektedir. Satırdaki her sütun ise bir 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.				

O2.B2.3. 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.

O2.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’

<table>
<tr>
<td>int</td>
<td>
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.
<pre>
int sayi = 5;
cout << sayi;
</pre>
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.
<pre>
short sayi = 5;
cout << sayi;
</pre>
<pre>
short int sayi = 5;
cout << sayi;
</pre>
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:
<pre>
long sayi = 597;
cout << sayi;
</pre>
<pre>
long int sayi = 597;
cout << sayi;
</pre>
</td>
<td>4 Bayt</td>
<td>Yaş (17), Sınıf (3)</td>
</tr>
</table>

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.	4 Bayt	Boy (1,75), Kilo (60,5)
double	Double veri tipi 8 bayt uzunluğundadır. Görüldüğü üzere son derece yüksek bir hassasiyete sahiptir. Virgüllü sayıların kullanımında bilimsel gösterim kullanılabilir. Aşağıda bazı sayıların bilimsel gösterimde nasıl yazıldığını görebilirsiniz: 2.27e+5 -3.15e+3 342.2e-3 44.78e-2 Yukarıdaki örnekteki 2.27e+5 değeri, $2{,}27 \times 10^5$ değeri ile aynıdır.	8 Bayt	Pi Sayısı (3.1415926535)

O2.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
```

O2.B4.1. Sabitler İçin 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;
}
```

O2.B5.1. Kod Çıktılarını Karşılaştıralım

```
#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;
}
```

O2.B6.1. 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;
}
```

Çıktı:

```
#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;
}
```

Çıktı:

```
#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;
}
```

Çıktı:

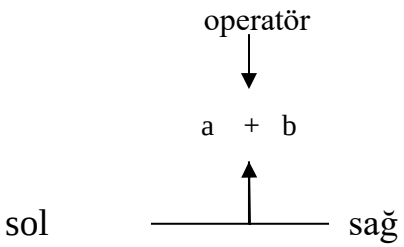
```
#include <iostream>
using namespace std;
int main()
{
    float x, y, z;
    x = 2.4;
    y = x;
    z = x + 1.3 + (y * 2.0);

    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;
}
```

Çıktı:

O2.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 17. *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 arttırmak 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 18. *Karşılaştırma operatörleri*

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 19. *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 tam 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 bölümünden kalanı 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 20. *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 "1" olacaktır. A ve B değişkenleri için oluşturulan aşağıdaki tabloyu inceleyebilirsiniz.

Tablo 21. *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 22. *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

Hafta 2. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftanın amacı, C++'ta geçerli değişken isimlendirme kurallarını ve anlamlı isimlerin önemini kavrayarak, farklı veri türlerini (int, double, char, float, bool) gerçek hayattan örneklerle ilişkilendirmek ve bu değişkenleri aritmetik, mantıksal ve karşılaştırma operatörleriyle Code::Blocks ortamında uygulamaktır.

Etkinlik Uygulama Ortamı

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır. Bireysel çalışma ürünleri ise dijital tartışma panosu aracılığıyla toplanacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Günün Rotası (10 dk.)

B. Öğrenme Görevleri

B1. Değişkenleri İsimlendirelim (20 dk.)

B2. Veri Tiplerini Okuyalım (20 dk.)

Ders Arası (10 dk.)

B3. Operatör Bulmacaları Çözelim (50 dk.)

A. Giriş: Günün Rotası

Süre: 10 dk.

Uygulama: Eğitmen, çevrim içi ders saati başladığında, öğrencilerin sanal sınıfa katılmasını beklerken arka planda rahatlatıcı ve motive edici bir müzik açarak pozitif bir atmosfer oluşturur. Yeterli katılım sağlandıktan sonra müziği yavaşça kısarak öğrencilere "Hoş geldiniz" der ve bugünkü dersin yol haritasını tek bir paragrafta net bir şekilde özetler: "Arkadaşlar, bugünkü temel amacımız, C++'ta bir bilgiyi (bir sayı, bir harf veya bir evet/hayır durumu gibi) bilgisayara nasıl doğru bir şekilde tanıtacağımızı ve saklayacağımızı öğrenmektir. Bu amaç doğrultusunda konumuz ise int, double, bool gibi temel veri tipleri, bu verilere verdiğimiz değişken isimleri ve onlarla toplama, karşılaştırma gibi basit işlemler yapmamızı sağlayan operatörler olacak. Gün sonunda, bu bilgileri kullanarak Code::Blocks üzerinde küçük ama anlamlı kod parçaları yazabiliyor hale geleceğiz." Bu kısa açıklama ile eğitmen, hem beklentileri belirlemiş hem de öğrencileri konuya hazırlamış olur.

B. Öğrenme Görevleri

B1. Değişkenleri İsimlendirelim

Süre: 20 dk.

Materyaller: OÇ2.B1.1. Öğrenci Yönerge Metni

OÇ2.B1.2. Kod Bloğu

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu yönergenin sonunda yer alan OÇ2.B1.1 ve OÇ2.B1.2 materyalleri, padlet ortamında öğrencilere sunulmak üzere paylaşılmalıdır.

Uygulama: Etkinlik akışı, eğitmenin ilk 7 dakikayı ekranını paylaşarak Code::Blocks üzerinde canlı bir kodlama oturumuna ayırmasıyla başlar. Bu bölümde eğitmen, herhangi bir ek materyal vermeden, değişken isimlendirmenin program okunabilirliği üzerindeki etkisini bir analoji ile vurgular ve geçersiz isimlendirme denemeleriyle bunların neden olduğu derleyici hatalarını gösterir. Ardından, yalnızca camelCase yazım stilini kullanarak doğru isimlendirme pratiklerini somut örneklerle uygular.

Gösterimin ardından gelen 3 dakika içinde, eğitmen Padlet panosunun bağlantısını sohbet üzerinden paylaşır ve ekran paylaşımında padlet üzerinde eğitmenin paylaştığı OÇ2.B1.1'de yer alan öğrenci yönerge metnini sunarak görevi açıklar. Görevin teknik detayı olan kod bloğunun ise (OÇ2.B1.2) öğrencilerin kendi bilgisayarlarındaki Code::Blocks programına aktarmaları gerektiğini belirtir. Etkinliğin devam eden 10 dakikalık bölümü, öğrencilerin bireysel çalışma zamanıdır. Bu sürede öğrenciler, OÇ2.B1.2'deki kodu kullanarak yönergeye uygun şekilde değişkenleri isimlendirir ve çalışmalarını Padlet panosuna yüklerler. 10 dk. Sürenin planlanması için zaman yönetimi için eğitmen ekran paylaşımında sayaç açılabilir. Böylece öğrencilerin 10 dk. Sonunda gönderim yapmaları gerektiği vurgulanmış olur.

Eğitmen bu esnada panoyu gözlemleyerek süreci takip eder. Son 10 dakikalık bölümde ise sentez ve değerlendirme aşamasına geçilir. İlk 3-4 dakika öğrencilerin birbirlerinin

çalışmalarını yorumlamaları için akran değerlendirmesine ayrılır. Öğrencilerin diğer gönderileri inceleyerek, değişken isimlerinde varsa hatalı gördüklerini yorumda yazmaları istenir. Kalan sürede eğitmen, Padlet panosu üzerinden genel bir değerlendirme yaparak başarılı camelCase uygulamalarını vurgular, sık yapılan hatalara değinir ve dersin ana kazanımını özetleyerek, bir sonraki etkinlikte veri tiplerini inceleyeceklerini belirtir.

B2. Veri Tiplerini Okuyalım

Süre: 20 dk.

Materyaller: OÇ2.B2.1. Anket Soruları

Hazırlık: Eğitmen B1 etkinliğindeki padlet ortamında ilerlemeye devam eder. Bu ortamda OÇ2. B2.1.'i materyalinde yer alan soruları kullanarak bir anket paylaşacaktır.

Uygulama: Eğitmen, bu bölümde her bir veri tipini sırayla ve tempolu bir şekilde işler. Eğitmen, bir önceki B1 etkinliğinde kullanılan ve öğrencilerin kodlarını içeren Padlet panosunu ekranına yansıtarak 15 dakikalık pekiştirme oturumuna başlar. Oturum, her bir temel veri tipi (int, bool, double, float, char) için yapılandırılmış bir döngü ile ilerler. Eğitmen, önce ilgili veri tipinin amacını Padlet'teki mevcut kodlar üzerinden açıklar ve ardından "Şimdi sen de bu veri tipine uygun kendi değişken örneğini sohbetten bana yaz" yönlendirmesiyle öğrencilerden bir dakika içinde kendi örneklerini oluşturmalarını ister. Sohbet etkileşiminin ardından, eğitmen kod çıktısı tahmin etme görevine geçer. Bunun için, OÇ2.B2.1. materyalindeki ilgili kod parçacığını ve "Bu kodun ekran çıktısı ne olur?" sorusunu seçenekleriyle birlikte kopyalayarak Padlet panosuna yeni bir gönderi olarak ekler. Anket işlevi için öğrencilerden, bu yeni gönderinin altına yorum olarak doğru olduğunu düşündükleri şıkkı (A, B, C veya D) yazmalarını ister. Bu yorumlama-anketi için yaklaşık bir dakika süre tanıdıktan sonra, eğitmen yorumları hızla gözden geçirir, doğru yanıtı ve arkasındaki mantığı açıklayarak veri tipine dair anahtar noktaları pekiştirir. Eğitmen, bu döngüyü 15 dakikalık ders süresi içinde diğer temel veri tipleri için de tekrarlayarak oturumu tamamlar.

Eğitmene Öneriler

- **Padlet ile Anket Yönetimi:** Padlet üzerinde yorumlarla anket yapmak, öğrencilerin birbirlerinin yanıtlarını görmesine neden olacaktır. Bu durumu bir "beyin fırtınası" veya "grupça tahmin etme" etkinliği olarak pozitif bir çerçeveye oturtabilirsiniz. "Bakalım çoğunluk ne düşünüyor?" gibi ifadelerle ortamı yönetebilirsiniz.
- **Akışı Kontrol Etme:** Yorum akışını yönetmek için öğrencilerden sadece tek bir harf (A, B, C, D) yazmalarını isteyin. Sürenin sonunda "Yorumları inceliyorum, artık yazmayalım." diyerek akışı sonlandırabilirsiniz.
- **Tempo:** 15 dakikada 5 veri tipini bu yöntemle işlemek yüksek bir tempo gerektirir. Her adıma ayrılan sürelere sadık kalmak ve açıklamaları net ve kısa tutmak önemlidir.

B3. Operatör Bulmacaları Çözelim

Süre: 50 dk.

Materyaller: OÇ2.B3.1. Operatör Bulmacaları ve Cevapları

Hazırlık: Eğitmen, ders öncesinde bir sunum hazırlar. OÇ2.B3.1'deki her bulmaca için ikişer slayt tasarlar:

1. **Tahmin Slaytı:** Sadece bulmacanın başlığını ve **açıklama yorumları olmayan** "temiz" kod versiyonunu içerir.
2. **Cevap Slaytı:** Aynı kodu, bu kez **açıklama yorumları eklenmiş halde** ve doğru çıktıyı belirgin bir şekilde göstererek içerir.

Uygulama: Bu etkinlik, eğitmenin sunumu yönettiği ve öğrencilerin sohbet üzerinden hızla katılım gösterdiği canlı bir oyun formatında ilerler.

Aşama 1: Giriş ve Kurallar (3 Dakika) Eğitmen enerjik bir başlangıç yapar: "Şimdi ekrana sırayla 6 tane kod bulmacası yansıtacağım. Her bulmaca için tam 1 dakikanız olacak. Göreviniz, kodun çıktısının ne olacağını tahmin edip, sadece sayıyı sohbet bölümüne yazmak! Hazır mısınız?" Kurallar net bir şekilde anlatılır.

Aşama 2: Canlı Bulmaca Çözme Döngüsü (15 Dakika) Bu aşama, her bulmaca için yaklaşık 2.5 dakika süren 6 turdan oluşur.

Örnek Tur (Bulmaca 1):

1. **Tahmin Aşaması (1 Dakika):** Eğitmen, **Bulmaca 1'in "Tahmin Slaytı"nı** ekrana yansıtır. "Bulmaca 1 için süreniz başladı! Sadece cevabınızı sohbete yazın!" der ve bir zamanlayıcı başlatır ya da 1 dk. müzik açar. Bu sırada öğrencilerin sohbetten yazdığı tahminleri sessizce gözlemler.
2. **Değerlendirme ve Açıklama (1.5 Dakika):** Süre dolduğunda, "Süre doldu, harika tahminler geldi, teşekkürler!" diyerek sohbeti durdurur. Ardından **Bulmaca 1'in "Cevap Slaytı"nı** (açıklamalı kod versiyonu) ekrana getirir. "Doğru cevap **15**'ti! 15 yazan herkesi tebrik ediyorum! Gördüğünüz gibi, altin = altin + 5; satırı, mevcut 10 altının üzerine 5 tane daha ekleyerek sonucu 15 yaptı." diyerek kodun arkasındaki mantığı basitçe açıklar. Sohbetten doğru cevap veren birkaç öğrencinin ismini okuyarak onları motive eder.

Eğitmen, bu iki adımlık döngüyü diğer 5 bulmaca için de sırayla tekrarlar.

Aşama 3: Kapanış ve Özet (2 Dakika) Tüm bulmacalar tamamlandıktan sonra eğitmen etkinliği özetler: "Bugün toplama, kalan bulma, karşılaştırma ve daha birçok operatörü çözdünüz. Bu operatörler, yazdığınız tüm oyunların ve programların temelidir. Katılımınız için teşekkürler!" diyerek dersi pozitif bir şekilde bitirir. Öğrencileri, bu kodları ders sonrasında kendi Code::Blocks programlarında denemeye teşvik eder.

Hafta 2. Çevrim İçi: Ders Materyalleri

OÇ2.B1.1. Öğrenci Yönerge Metni

(Bu metin padlette paylaşılarak, görev verilirken ekran paylaşımında gösterilecektir.)

Göreviniz: Bu görevde, kütüphanedeki bir kitabı anlatan değişkenler oluşturacaksınız. Tek yapmanız gereken, her bir değişken için yorum satırında (/* ... */ içinde) belirtilen anlama uygun, geçerli ve anlamlı bir isim bulmak. İsimleri yazarken **camelCase** (deve hörgücü) yazım şeklini kullanmalısınız. Bu yöntemde ilk kelime küçük harfle başlar, sonraki her kelimenin ilk harfi büyük olur (örneğin: kitapFiyati).

Adımlar:

1. Aşağıda verilen OÇ2.B1.2 kod bloğunu kopyalayın ve bilgisayarınızdaki Code::Blocks programında açtığınız yeni .cpp dosyasına yapıştırın.
2. Kodun içindeki her satırda ... ile gösterilen yerlere, yorum satırındaki açıklamaya uygun bir değişken ismi yazın.
3. Tamamladığınız kodun ekran görüntüsünü ve metin halini, başlık kısmına kendi adınızı yazarak öğretmeninizin söylediği Padlet panosuna yükleyin.

OÇ2.B1.2. Paylaşılacak Kod Bloğu

```
#include <iostream>

int main() {

//GOREV: Bu degiskenlere anlamli ve camelCase formatinda isimler verin

    int    ... /* Kitabin toplam sayfa sayisi */ ;
    double ... /* Kitabin Turk Lirasi cinsinden fiyati (Orn: 89.99) */ ;
    float  ... /* 10 uzerinden okuyucu puani (Orn: 8.5) */ ;
    bool   ... /* Kitabin kutuphanede olup olmadigi (var icin true, yok
icin false) */ ;
    char   ... /* Kitabin bulundugu rafin harfi (Orn: 'A', 'B', 'C') */ ;

    // ------------------------------------------------------------

    // Bu asamada degiskenlere bir deger yazmaniza gerek yok.
    // Sadece dogru ve anlamli isimlendirme yapmaniz yeterli.

    return 0;
}
```

OÇ2.B2.1. Anket Soruları

(Bu içerikler, her bir veri tipi için sırayla Padlet'e yeni bir anket gönderisi olarak eklenecektir.)

Gönderi 1: (int) *İçerik:*

```
#include <iostream>
int main() {
    int sayfaSayisi = 416;
    std::cout << sayfaSayisi;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) sayfaSayisi B) 416 C) 416.0 D) Hata verir.

Gönderi 2: (bool) *İçerik:*

```
#include <iostream>
int main() {
    bool stoktaVarMi = true;
    std::cout << stoktaVarMi;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) true B) stoktaVarMi C) 1 D) Hata verir.

Gönderi 3: (double) *İçerik:*

```
#include <iostream>
int main() {
    double kitapFiyati = 189.99;
    std::cout << kitapFiyati;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) 189 B) 190 C) 189.99 D) kitapFiyati

Gönderi 4: (float) *İçerik:*

```cpp
#include <iostream>
int main() {
    float okuyucuPuani = 8.5f;
    std::cout << okuyucuPuani;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) 8 B) "8.5f" C) 8.5 D) Hata verir.

Gönderi 5: (char) *İçerik:*

```cpp
#include <iostream>
int main() {
    char depoBolumu = 'C';
    std::cout << depoBolumu;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) "C" B) 'C' C) depoBolumu D) C

Cevaplar: 1-B, 2-C, 3-C, 4-C, 5-D

OÇ2.B3.1. Operatör Bulmacaları ve Cevapları

(Her bulmaca için iki versiyon sunulmuştur: Biri tahmin için, diğeri açıklama için.)

Bulmaca 1: Altın Toplama
Bir video oyunu karakterinin 10 altını varken hazine sandığından 5 altın daha çıkarsa, toplamda kaç altını olur?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int altin = 10;
    altin = altin + 5;
    std::cout << altin; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int altin = 10;
    // Mevcut 10 altina 5 tane daha ekleniyor.
    altin = altin + 5;
    std::cout << altin; // Cikti: 15
    return 0;
}
```

Doğru Çıktı: 15

Bulmaca 2: Kurabiye Paylaşımı
Bir sepetteki 17 kurabiyeyi 5 çocuğa eşit olarak paylaştırdığımızda, geriye paylaştırılamayan kaç kurabiye kalır?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int toplamKurabiye = 17;
    int kisiSayisi = 5;
    int artanKurabiye = toplamKurabiye % kisiSayisi;
    std::cout << artanKurabiye; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int toplamKurabiye = 17;
    int kisiSayisi = 5;
    // 17'nin 5'e bolumunden kalani bulur (17 = 3*5 + 2).
```

```
    int artanKurabiye = toplamKurabiye % kisiSayisi;
    std::cout << artanKurabiye; // Cikti: 2
    return 0;
}
```

Doğru Çıktı: 2

Bulmaca 3: Seviye Atlama
Özel bir kalkanı kullanma şartı 10. seviyeden yüksek olmaksa, 12. seviyedeki bir oyuncu bu şartı sağlar mı?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int oyuncuSeviyesi = 12;
    bool yuksekSeviyeMi = (oyuncuSeviyesi > 10);
    std::cout << yuksekSeviyeMi; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int oyuncuSeviyesi = 12;
    // 12, 10'dan buyuk oldugu icin bu kosul dogrudur (true).
    // Dogru (true) ekrana 1 olarak yazdirilir.
    bool yuksekSeviyeMi = (oyuncuSeviyesi > 10);
    std::cout << yuksekSeviyeMi; // Cikti: 1
    return 0;
}
```

Doğru Çıktı: 1

Bulmaca 4: Gizemli Giriş

Bir online oyuna giriş için hem kullanıcı adının hem de şifrenin doğru olması gerekiyorsa, ikisini de doğru giren birisi giriş yapabilir mi?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    bool anahtarVarMi = true;
    bool parolaDogruMu = true;
    bool kapiAcilirMi = anahtarVarMi && parolaDogruMu;
    std::cout << kapiAcilirMi; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```cpp
#include <iostream>
int main() {
    bool anahtarVarMi = true; // Evet, anahtar var (1)
    bool parolaDogruMu = true; // Evet, parola dogru (1)
    // VE (&&) operatoru, her iki taraf da dogru (1) ise dogru (1) olur.
    bool kapiAcilirMi = anahtarVarMi && parolaDogruMu;
    std::cout << kapiAcilirMi; // Cikti: 1
    return 0;
}
```

Doğru Çıktı: 1

Bulmaca 5: Tecrübe Kazanma
Bir karakterin 50 tecrübe puanı varken, tamamladığı bir görevden 25 puan daha kazanırsa, yeni tecrübe puanı kaç olur?

Tahmin için Gösterilecek Kod:

```cpp
#include <iostream>
int main() {
    int tecrubePuani = 50;
    tecrubePuani += 25;
    std::cout << tecrubePuani; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```cpp
#include <iostream>
int main() {
    int tecrubePuani = 50;
    // 'tecrubePuani = tecrubePuani + 25' isleminin kisayoludur.
    tecrubePuani += 25;
    std::cout << tecrubePuani; // Cikti: 75
    return 0;
}
```

Doğru Çıktı: 75

Bulmaca 6: Fotoğraf Beğenme

Sosyal medyadaki bir fotoğrafın 99 beğenisi varken, siz de 'beğen' butonuna tıklarsanız beğeni sayısı kaça yükselir?

Tahmin için Gösterilecek Kod:

```cpp
#include <iostream>
int main() {
    int begeniSayisi = 99;
    begeniSayisi++;
    std::cout << begeniSayisi; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int begeniSayisi = 99;
    // '++' operatoru, sayiyi 1 artirir.
    begeniSayisi++;
    std::cout << begeniSayisi; // Cikti: 100
    return 0;
}
```

Doğru Çıktı: 100

Hafta 3. Yüz Yüze: Karar ve Döngü 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.
- K4. Problem çözme süreçlerinde döngü yapılarını kullanarak algoritma tasarlar.
- K5. Problem çözümünde döngü yapısını kullanır.

Amaç

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

Önerilen Ders Akışı

A. Giriş: Ders Öncesi Enerji (5 dk.)

B. Öğrenme Görevleri

B1. Karar ve Mantık Yapılarını Tanıyalım (20 dk.)

B2. Görevleri Kodlayalım (25 dk.)

Ders Arası (10 dk.)

B3. İç İçe Koşula Farklı Bir Bakış (50 dk.)

Ders Arası (10 dk.)

B4. Döngüleri Tanıyalım (20 dk.)

B5. Döngülerin Neden Kullanıldığını Keşfedelim (30 dk.)

Ders Arası (10 dk.)

B6. Döngü Görevlerini Kodlayalım (50 dk.)

A. Giriş: Ders Öncesi Enerji

Süre: 5 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 ve Mantık Yapılarını Tanıyalım

Süre: 20 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: O3.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 görev 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 bulunan çıktı sonuçlarının doğru veya yanlışlığı hakkında öğrencilere geri bildirim verir. 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 programın hangi çıktıyı vereceği öğrenciler tarafından tahmin edilir.

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

Eğitmene Öneriler: Yukarıdaki etkinliğin amacı; öğrencinin program akışının bir noktasında verilen karara göre (seçilen sayıya göre) yapılacak işlemlerin ve ekran çıktısının nasıl değişebileceğini öğrencilere hissettirmektir.

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

Karar yapıları, bir programın farklı durumlara göre farklı işlemler yapmasını sağlayan temel yapılardandır. Gerçek hayatta da çoğu zaman kararlar alırız: hava yağmurluysa şemsiye alırız, değilse almamıza gerek kalmaz. İşte programlar da benzer şekilde, belirli koşulların doğru ya da yanlış olmasına göre nasıl davranacaklarını karar yapıları ile belirler. Bu sayede programın akışı daha esnek ve dinamik bir hale gelir.

Programlamada en sık kullanılan karar yapıları arasında if, if-else, else-if zincirleri ve switch ifadeleri yer alır. if ifadesiyle belirli bir koşul kontrol edilir ve doğruysa ilgili kod bloğu çalıştırılır. if-else yapısıyla hem doğru hem de yanlış durumda yapılacak işlemler tanımlanabilir. Daha karmaşık çoklu koşullarda ise else-if kullanılarak birçok alternatif senaryo kontrol edilebilir. switch ifadesi ise sabit değerlere bağlı dallanmalarda kodu daha okunabilir hale getirmek için tercih edilir.

Karar yapıları, kullanıcıdan alınan girişlere veya dış ortamlardan gelen verilere göre programın doğru tepki vermesini sağlar. Örneğin bir menü sistemi, girilen seçeneğe göre farklı işlemleri tetikleyebilir. Veya bir sensör değerine göre alarm sistemi devreye girebilir. Karar yapıları olmadan programlar, yalnızca sırayla çalışan, tepki veremeyen yapılar olurdu. Bu nedenle, karar yapıları yazılım geliştirme sürecinin vazgeçilmez temel taşlarındandır.

Eğer koşula göre yapılacak bir veya daha çok işlem varsa ilgili blok ‘{‘ ve ‘}’ arasına yazılır.

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

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: 25 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: O3.B1.1 Karar Yapılarını Tanıyalım Görev Kartları

Hazırlık: Eğitmen B1 öğrenme görevindeki materyali bu etkinlikte de kullanmaya devam eder.

Uygulama: Öğrencilerin materyalde belirtilen ve bir ders önceki yaptıkları "Karar Yapılarını Tanıyalım" etkinliği içerisinden seçtiği gerçek yaşamla ilgili iki karmaşık görevi kendi başlarına kodlaması beklenir. Öğrenciler kodlama yaparken eğitmen sınıfta dolaşarak öğrencilerin ihtiyaç duyduğu anda yöntemsel bilgiyi verecektir. Ayrıca kodları bilgisayarda yazarak öğrencilerin seçtiği görevlerdeki bazı aşamaları ihtiyaç hâlinde öğrencilerin görmesine olanak sağlar.

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

eğer her öğrenci için bir bilgisayar varsa her öğrencinin en az iki görevi kodlaması istenir. Kodlanamayan diğer görevler öğrencilere ödev olarak verilebilir.

Uygulama esnasında eğitmenin görevi, öğrencilerin ihtiyaç duyduğu zamanlarda yöntemsel bilgi vermesidir. Bu yöntemsel bilgi; kodun tamamını öğrencilere göstermek yerine, öğrencilerin zorluk yaşadığı aşamalarda ihtiyaç duydukları ipuçlarıdır. İpuçları öğrencileri doğru cevabı direkt söylemez ancak üzerinde düşünmesine teşvik eder.

B3. İç İçe Koşula Farklı Bir Bakış

Süre: 50 dk.

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

Materyaller: O3.B3.1. SWITCH Yapısı Kullanımı

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

Uygulama: Bu bölüm için hazırlanan materyal sunum üzerinden paylaşılarak if yapısı ile switch yapısı konusunda kod yazımında ne gibi farklılıkların olduğuna yönelik sınıf içi tartışma ortamı oluşturularak öğrencilerin bu farklılıkları bulması istenir. Daha sonra eğitmen tarafından aşağıdaki gibi konu öğrencilerin de katılımı ile özetlenir.

Birden çok koşul olduğu durumlarda “switch” bloğunu kullanabiliriz. Her koşul durumunu “case” komutları ile belirterek daha kolay okunabilir bir kod parçası oluşturabiliriz. Söz dizimi şu şekildedir;

```
switch(değiskenAdi)
{
        case durum 1:
                Yapılacak işlemler
        break;
        case durum 2:
                Yapılacak işlemler
        break;
        …
        default:
                Yapılacak işlemler
        break;
}
```

break: Bu komut, koşul bloğundan çıkmamızı sağlar.

default: Verilen koşulların gerçekleşmemesi hâlinde çalışır.

Eğitmene Öneriler: Eğitmenler tarafından konu özetlendikten sonra öğrencilerin switch komutunu kullanarak, aynı örneği bilgisayarda kodlaması istenir. Kodlama esnasında eğitmenlerin öğrencilere işlemsel bilgiyi ihtiyaç duydukları anda vermesi beklenir.

B4. Döngüleri Tanıyalım

Süre: 20 dk.

Kazanımlar: K4. Problem çözme süreçlerinde döngü yapılarını kullanarak algoritma tasarlar.

Materyaller: O3.B4.1. Döngüleri Tanıyalım 1

O3.B4.2. Döngüleri Tanıyalım 2

Hazırlık: Eğitmen iki materyali de dört gruba dağıtmak için birer 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 çiçeklerden nektarı alması için gerekli kodu yazması beklenir. Etkinliklerin cevabı aşağıda verilmiştir.

Resim 24. *Çözüm 1*

Resim 25. *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.

Döngü yapıları, bir işlemin birden fazla kez ve genellikle belirli bir koşul sağlandığı sürece tekrarlanmasını sağlayan temel programlama araçlarıdır. Günlük yaşamda da benzer şekilde tekrar eden işler vardır: alarm çalana kadar uyumak, yemek pişene kadar beklemek gibi. Programlama dünyasında da benzer şekilde, aynı işlemi defalarca yazmak yerine döngüler kullanılarak hem kod sadeleşir hem de zaman kazandırılır. Döngüler, bir programın daha az kodla daha fazla iş yapmasını sağlar.

Programlamada en yaygın kullanılan döngü türleri for, while ve do-while döngüleridir. for döngüsü, genellikle tekrar sayısının önceden belli olduğu durumlarda tercih edilir. Örneğin 1'den 100'e kadar olan sayıların ekrana yazdırılması for döngüsü ile kolayca yapılabilir. while döngüsü ise, tekrar sayısı belli olmayan ama bir koşul sağlandığı sürece işlemlerin devam etmesi gereken durumlar için uygundur. do-while döngüsü ise, işlemin en az bir kez çalışmasının garanti edilmesi gereken durumlarda kullanılır.

Döngüler; liste işlemleri, veri girişi kontrolleri, oyun döngüleri, dosya işlemleri gibi birçok alanda kritik öneme sahiptir. Kullanıcıdan sürekli veri almak, sensör verilerini belli aralıklarla kontrol etmek ya da bir animasyonu sürekli olarak güncellemek gibi işlemler döngüler sayesinde etkin bir şekilde yapılabilir. Döngüler olmadan programlar, yalnızca sınırlı sayıda ve tekrarsız işlemler gerçekleştirebilirdi. Bu nedenle döngüler, özellikle büyük ölçekli ya da kullanıcı etkileşimli uygulamalarda vazgeçilmez yapılardır.

Döngü olmadan olmaz mı?

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

B5. Döngülerin Neden Kullanıldığını Keşfedelim

Süre: 30 dk.

Kazanımlar: K5. Problem çözümünde döngü yapısını kullanır.

Materyaller: O3.B5.1 Döngü Çeşitleri Afişi

O3.B5.2 Döngüleri Neden Yazıyorum

Hazırlık: Eğitmen O3.B5.1 kodlu materyali ÖYS üzerinden erişime açar. Diğer O3.B5.2. kodlu materyalden ise 10 adet çıktı alır.

Uygulama: Her iki öğrenci için "O3.B5.2" kodlu materyalin çıktısı alınarak 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 (O3.B5.1) 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.

B6. Döngü Görevlerini Kodlayalım

Süre: 50 dk.

Kazanımlar: K5. Problem çözümünde döngü yapısını kullanır.

Materyaller: O3.B6.1. Döngü Görevlerini Kodlayalım

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan materyalleri ÖYS üzerinden paylaşır. Code::Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Öğrenciler ikili gruplar hâlinde bilgisayar başındadır. Eğitmenler döngü görevi etkinliğindeki uygulamalarından iki tanesini öğrencilerin seçmesini sağlayarak, öğrencilerin bu derste kodlama yapmasını ister. Diğer görevler ise öğrencilere kodlanması için ev görevi olarak verilebilir. Öğrenciler kodlama esnasında daha önceki derste hazırlanan afişi destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmen sınıfı dolaşarak ve 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. 0 ila 32767 arasında rastgele sayı üretir. Üst sınırı sınırlandırmak için mod (%) operatörü kullanılır. 0-100 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.

Döngü görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Merve adlı öğrenci 1'den 20'ye kadar olan tek sayıları bulan bir program yazıp ekranda göstermek istemektedir. Merve bunun için sizce nasıl bir kod yazmalı?

Cevap 1:

For

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

int i=1;
```

While

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

int i=1;
```

Do-While

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

Görev 2: Ali kardeşi Buğra'nın 1'den 30'a kadar 3'erli olarak sayı saymasını istemektedir. Buradan hareketle kardeşinin doğru sayıp saymadığını kontrol etmesi için bilgisayarda 1'den 30'a kadar küçükten büyüğe olacak şekilde 3'e bölünebilecek sayıları ekranda göstermek istiyor. Bunun için nasıl bir kod yazmalı?

Cevap 2:

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=0; i < 30; i++)
            if(i%3==0)
                    cout << i <<endl;
    return 0;
}
```

Görev 3: Caner öğretmen, sınıfında bulunan 5 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 < 5; i++)
    {
        int puan;
        cout << i+1 <<". ogrenci puani:";
        cin >> puan;
        toplam += puan;
    }
    cout << "ortalama : " << toplam /5 ;
}
```

Görev 4: Defne öğretmen aldığı 10 tane kitabı sınıfındaki öğrencilere çekiliş yoluyla dağıtmak istemektedir. Sınıftaki öğrencilerin okul numaraları 50 ile 100 arasındadır. Defne öğretmen bunun için 50 ile 100 arasında 10 tane rastgele sayı üreten bir program yazarak çıkan numaraya sahip öğrencilere kitabı vermeyi planlamaktadır. Bunun için nasıl bir kod yazmalıdır?

Cevap 4:

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

Görev 5: Ahmet okul kütüphanesindeki raflara herkesin kolayca kitapları bulabilmesi için sayı etiketleri yapıştırmak istiyor. Kütüphanede 30 tane raf olduğu düşünülürse Ahmet'in 1'den 30'a kadar sayıları sıralayıp ekranda göstermesi gerekmektedir. Buradan hareketle Ahmet'in nasıl bir kod yazması gereklidir, bilgisayarımızda kodlayalım.

Cevap 5:

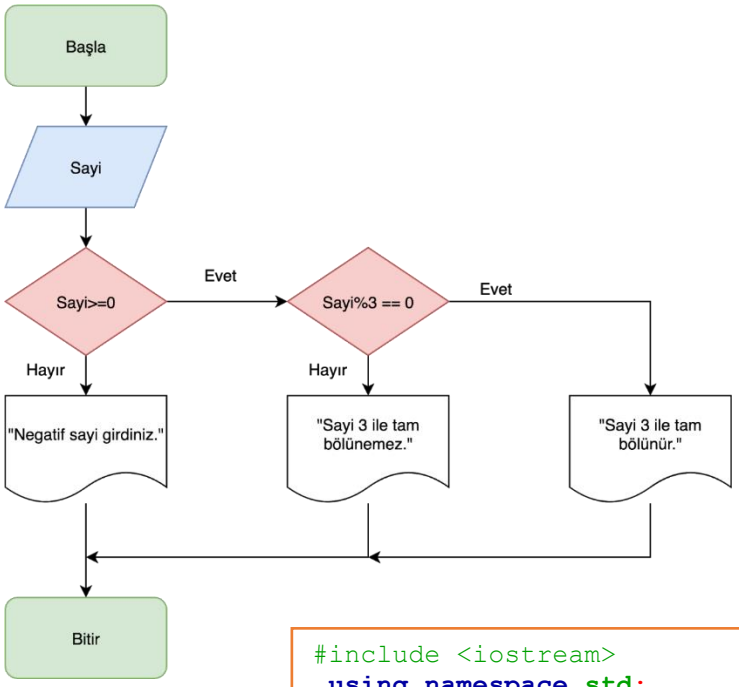
```
#include <iostream>
using namespace std;
int main()
{
    int sayi = 1;
    while(sayi<31)
    {
        cout << sayi <<endl;
        sayi ++;
    }
}
```

Hafta 3. Yüz Yüze: Ders Materyalleri

O3.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



1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

4. **Grup Ekran Çıktısı:**

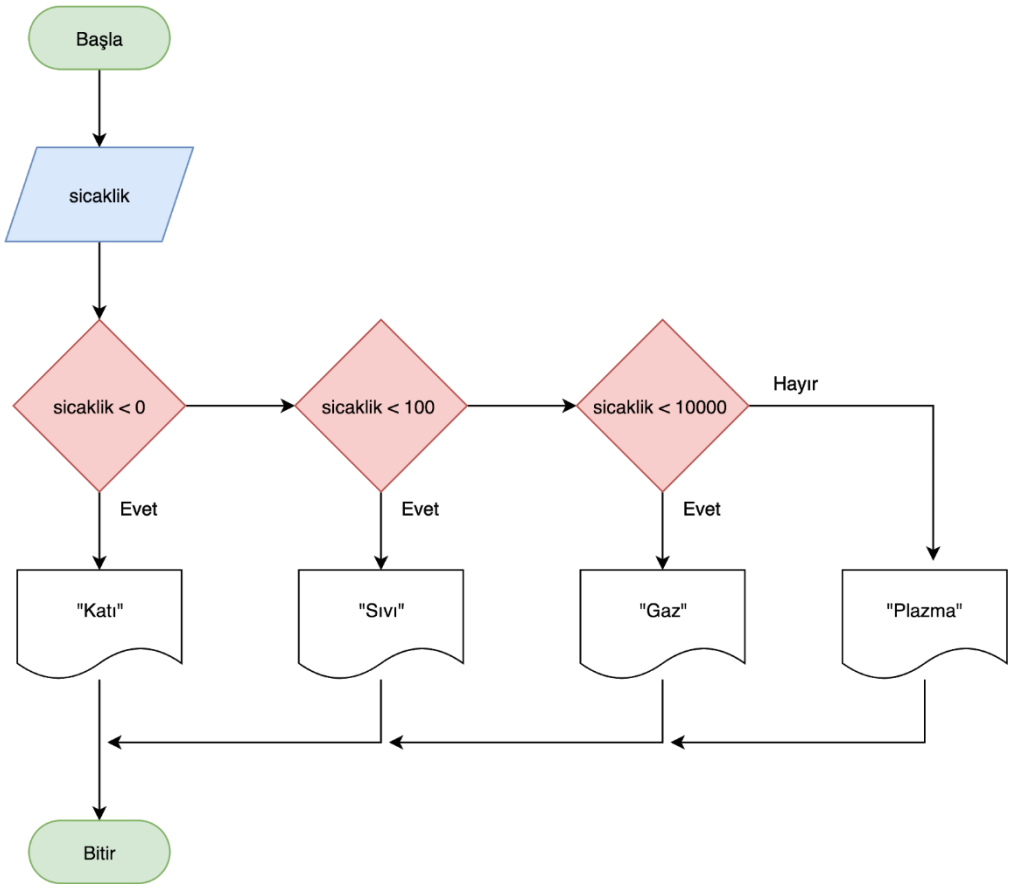
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

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

2. Görev



1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

4. **Grup Ekran Çıktısı:**

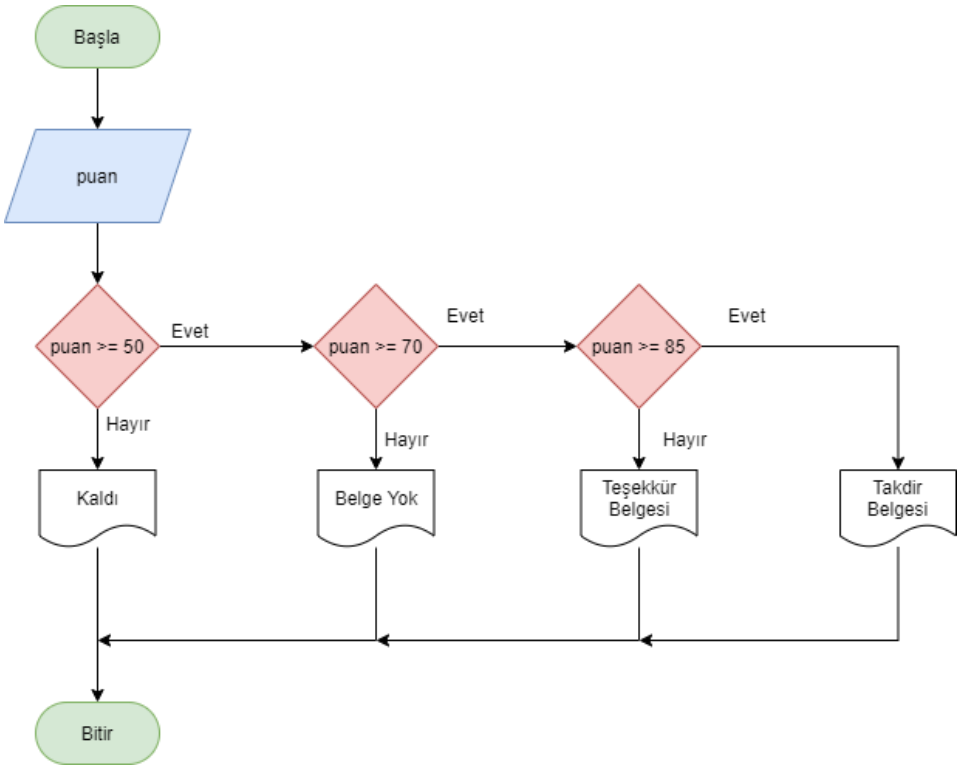
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

```cpp
#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:**

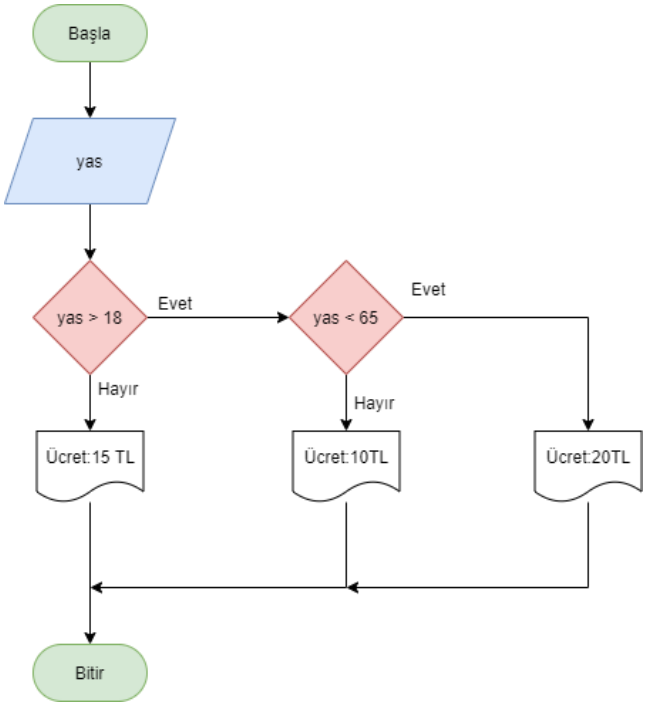
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

```
#include <iostream>
using namespace std;
int main()
{
    int puan;
    cout << "notu giriniz:";
    cin >> puan;
    if(puan<50)
        cout << "kaldi";
    else if(puan<70)
        cout << "belge yok";
    else if(puan<85)
        cout << "tesekkur 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:

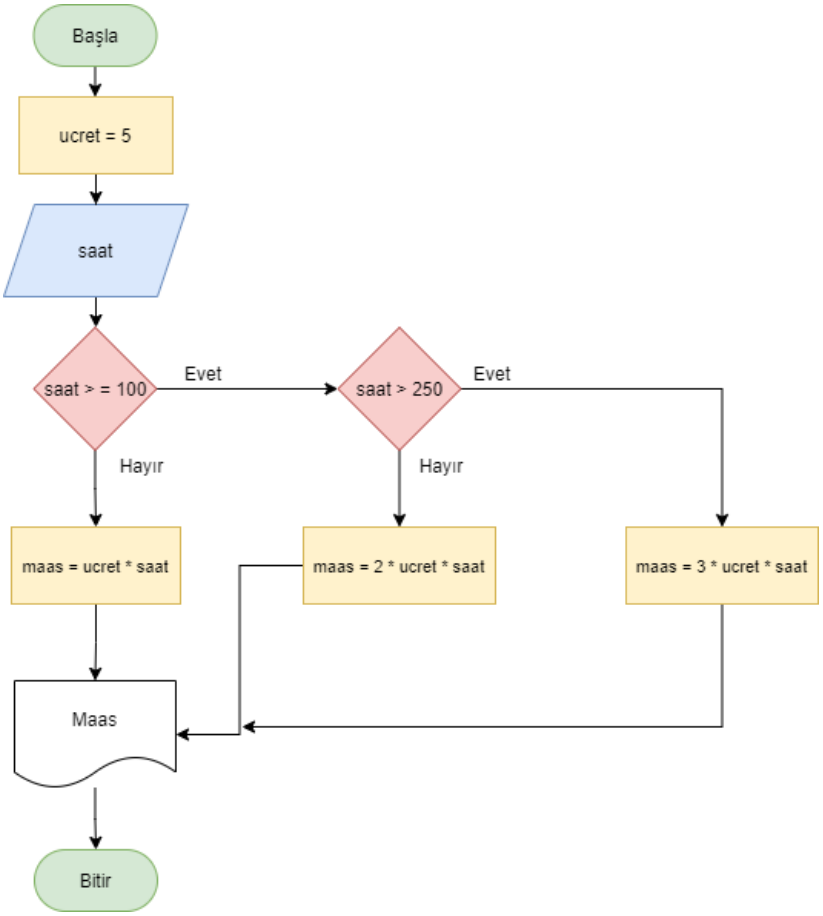
……

5. Grup Ekran Çıktısı:

……

```
#include <iostream>
using namespace std;
int main()
{
    int yas;
    cout << "yasinizi giriniz:";
    cin >> yas;

    if(yas<18)
        cout << "Ucret: 15 TL ";
    else if(yas<65)
        cout << "Ucret: 20 TL ";
    else
        cout << "Ucret: 10 TL";
    return 0;
}
```

5. Görev



1. Grup Ekran Çıktısı:

……

2. Grup Ekran Çıktısı:

……

3. Grup Ekran Çıktısı:

……

4. Grup Ekran Çıktısı:

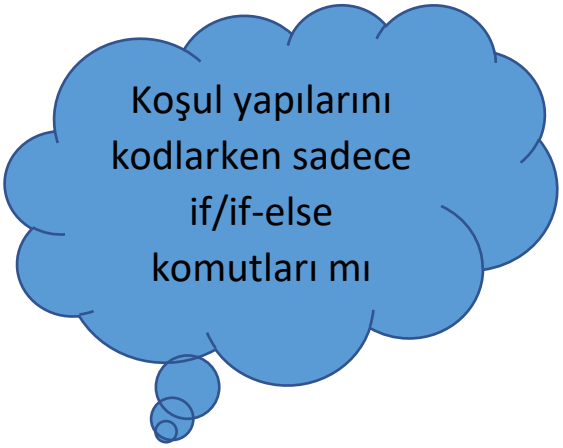
……

5. Grup Ekran Çıktısı:

……

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

O3.B3.1. SWITCH Yapısı Kullanımı



Cevap: Birden çok koşul olduğu durumlarda "switch" bloğunu kullanabiliriz. Gelin if ile switch kullanımını hesap makinesi yapımı için gereken kodlamayı yaparak kıyaslayalım.

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

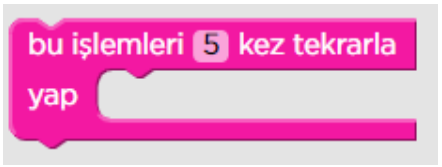
```
#include <iostream>
using namespace std;
int main()
{
    char islem;
    cin >> islem;
    switch(islem){
    case '+':
        cout << "Toplama islemi";
        break;
    case '-':
        cout << "Cikarma islemi";
        break;
    case '*':
        cout << "Carpma islemi";
        break;
 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

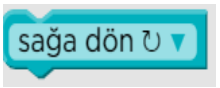
O3.B4.1. Döngüleri Tanıyalım 1

** 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?

O3.B4.2. Döngüleri Tanıyalım 2





** 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?

O3.B5.1. Döngü Çeşitleri Afişi

Resim 26. *Döngü çeşitleri afişi*

O3.B5.2. Döngüleri Neden Yazıyorum

Sol kısımda bulunan kodların hangi temel problem ve amaç için yazıldığını sağ sütunda belirtiniz.

Kod	Problem/Amaç
`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);`	

O3.B6.1. Döngü Görevlerini Kodlayalım

1

Merve adlı öğrenci 1'den 20'ye kadar olan tek sayıları bulan bir program yazıp ekranda göstermek istemektedir. Merve bunun için sizce nasıl bir kod yazmalı?

2

Ali kardeşi Buğra'nın 1'den 30'a kadar 3'erli olarak sayı saymasını istemektedir. Buradan hareketle kardeşinin doğru sayıp saymadığını kontrol etmesi için bilgisayarda 1'den 30'a kadar küçükten büyüğe olacak şekilde 3'e bölünebilecek sayıları ekranda göstermek istiyor. Bunun için nasıl bir kod yazmalı?

3

Caner öğretmen sınıfında bulunan 5 öğrencinin matematik dersinde aldığı notları klavyeden girerek bulunmasını isteyen bir program yapmak istiyor. Bunun için nasıl bir kod yazmalı.

4

Defne öğretmen aldığı 10 tane kitabı sınıfındaki öğrencilere çekiliş yoluyla dağıtmak istemektedir. Sınıftaki öğrencilerin okul numaraları 50 ile 100 arasındadır. Defne öğretmen bunun için 50 ile 100 arasında 10 tane rastgele sayı üreten bir program yazarak çıkan numaraya sahip öğrencilere kitabı vermeyi planlamaktadır. Bunun için nasıl bir kod yazmalıdır.

5

Ahmet okul kütüphanesindeki raflara herkesin kolayca kitapları bulabilmesi için sayı etiketleri yapıştırmak istiyor. Kütüphanede 30 tane raf olduğu düşünülürse
Ahmet'in 1 den 30'a kadar sayıları sıralayıp ekranda göstermesi gerekmektedir. Buradan hareketle Ahmet'in nasıl bir kod yazması gereklidir, bilgisayarımızda kodlayalım.

Hafta 3. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftaki çevrim içi ders içeriğinin amacı, C++'ta karar yapıları ve döngü yapılarını öğrenerek, farklı senaryolara uygun koşullu ifadeler ve tekrar eden işlemler tasarlamaktır. Gerçek hayattan örneklerle bu yapıların kullanımını kavramak ve Code::Blocks ortamında uygulamalar geliştirerek karar verme süreçleri ile döngü mantığını pekiştirmektir.

Etkinlik Uygulama Ortamı:

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Günün Rotası (5 dk.)

B. Öğrenme Görevleri

B1. Verilen Programın Tasarlanma Amacını Tahmin Ediyorum (10 dk.)

B2. Çalıştırmadan Tahmin Et: Karar Yapılarını Anlama (15 dk.)

B3. Akıştan Koda: Algoritma Dönüştürme Yarışması (20 dk.)

Ders Arası (10 dk.)

B4. Döngü Koşullarını İnceleyip Kodun Ekran Çıktısını Tahmin Etme (10 dk.)

B5. Rastgele Sayılarla Karşılaştırma: Büyük Sayıyı Bulalım (20 dk.)

B6. Belirli Kriterlere Uyan Öğrenci Numaralarını Bulma Programı (20 dk.)

A. Giriş: Günün Rotası

Süre: 5 dk.

Uygulama: Eğitmen, çevrim içi ders saati başladığında, öğrencilerin sanal sınıfa katılmasını beklerken arka planda odaklanmayı kolaylaştıracak hafif bir enstrümantal müzik açar. Yeterli katılım sağlandığında müziği yavaşça kısar ve öğrencileri sıcak bir şekilde selamlar. Ardından bugünkü dersin yol haritasını tek bir paragrafta net biçimde özetler: "Arkadaşlar, bugünkü temel amacımız, C++'ta bir programın belirli koşullara göre nasıl farklı yollar izleyebileceğini ve tekrarlayan işlemleri otomatik olarak nasıl gerçekleştirebileceğini öğrenmektir. Bu amaç doğrultusunda if, else if, else ve switch gibi karar yapıları ile for, while ve do-while döngülerini ele alacağız. Gerçek hayattan örneklerle, bir durum karşısında programın doğru kararı nasıl vereceğini ve tekrar eden görevleri en verimli şekilde nasıl yapacağını göreceğiz. Gün sonunda, bu yapıların mantığını kavrayarak Code::Blocks üzerinde koşullu ifadeler ve döngüler içeren küçük ama işlevsel programlar yazabiliyor olacağız.

B. Öğrenme Görevleri

B1. Verilen Programın Tasarlanma Amacını Tahmin Ediyorum

Süre: 10 dk.

Materyaller: OÇ3.B1.1. Öğrenci Yönerge Metni

OÇ3.B1.2. Kod Bloğu

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ3.B1.1. Öğrenci Yönerge Metni'ne göre verilen kodun neden yazıldığına yönelik çıkarımlarda bulunması istenecektir.

Aşağıda bir programcı tarafından yazılmış bir kod parçası verilmiştir. Bu kodu inceleyerek aşağıdaki soruları yanıtlayınız:

1. Programcı bu kodu yazarken neyi amaçlamış olabilir? (0-5 dk)
2. Kodun akışını (giriş, işlem, çıkış) kendi cümlelerinizle açıklayınız (0-5 dk).

B2. Çalıştırmadan Tahmin Et: Karar Yapılarını Anlama

Süre: 15 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda)

ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ3.B2.1. Öğrenci Yönerge Metni

OÇ3.B2.2. Yanıt

Uygulama: Öğrenciler, belirtilen OÇ3.B2.1. Öğrenci Yönerge Metni'nde yer alan kodun çıktısını tahmin etmesi istenecektir. Bu etkinliğine yönelik eğitmenin yapacağı zaman planı aşağıda gösterildiği gibi planlanacaktır.

Zaman Planı:

- **0-3 dakika:** Öğrenciler kodu dikkatle inceler, değişken değerlerini, karar yapısını ve işlem sırasını analiz eder.
- **3-5 dakika:** Her öğrenci, kodu çalıştırmadan önce ekranda ne çıkacağını bireysel olarak tahmin etmesi ve tahminlerini chat kısmından yazması istenir.
- **5-12 dakika:** Öğrenciler kodu Code::Blocks üzerinde çalıştırarak tahminleri ile gerçek sonucu karşılaştırır.
- **12-15 dakika:** Eğitmen, doğru tahmin yöntemlerini ve sık yapılan hataları sınıfla paylaşır, program akışını adım adım açıklayarak geri bildirim verir.

B3. Akıştan Koda: Algoritma Dönüştürme Yarışması

Süre: 20 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ3.B3.1. Öğrenci Yönerge Metni

OÇ3.B3.2. Kod Bloğu

Uygulama: Öğrenciler, OÇ3.B3.1. Öğrenci Yönerge Metni'nde verilen akış diyagramını inceleyerek adım adım C++ koduna dönüştüreceklerdir. Bu etkinliğine yönelik eğitmenin yapacağı zaman planı aşağıda gösterildiği gibi planlanacaktır.

Zaman Planı:

- **0-3 dakika:** Akış diyagramındaki başlangıç, işlem, karar ve bitiş adımlarının analiz edilmesi. Hangi veri tiplerinin kullanılacağına ve hangi kontrol yapılarının (if, else, for, while) gerektiğine karar verilmesi.
- **3-15 dakika:** C++ kodunun Code::Blocks ortamında yazılması. Girdi alma (cin), çıktı verme (cout) ve koşul/döngü yapılarını akış diyagramına uygun şekilde kullanma.
- **15-20 dakika:** Kodun test edilmesi, beklenen çıktılar ile karşılaştırılması ve varsa hataların düzeltilmesi. Eğitmen, doğru çözümleri vurgular ve geliştirme önerileri sunar.

B4. Döngü Koşullarını İnceleyip, Kodun Ekran Çıktısını Tahmin Etme

Süre: 10 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ3.B4.1. Öğrenci Yönerge Metni

OÇ3.B4.2. Yanıt

Uygulama: Öğrenciler, belirtilen OÇ3.B4.1. Öğrenci Yönerge Metni'nde yer alan kodun çıktısını tahmin etmesi istenecektir. Bu etkinliğine yönelik eğitmenin yapacağı zaman planı aşağıda gösterildiği gibi planlanacaktır.

Zaman Planı:

- **0-3 dakika:** Öğrenciler kodu dikkatle inceler, değişken değerlerini, karar yapısını ve işlem sırasını analiz eder.
- **3-5 dakika:** Her öğrenci, kodu çalıştırmadan önce ekranda ne çıkacağını bireysel olarak tahmin etmesi ve tahminlerini chat kısmından yazması istenir.
- **5-10 dakika:** Öğrenciler kodu Code::Blocks üzerinde çalıştırarak tahminleri ile gerçek sonucu karşılaştırır.

B5. Rastgele Sayılarla Karşılaştırma: Büyük Sayıyı Bulalım

Süre: 20 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ3.B5.1. Öğrenci Yönerge Metni

OÇ3.B5.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen OÇ3.B5.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Ahmet rastgele 2 sayı üretip büyük olanı ekrana yazdırmak istiyor. Ahmet'in nasıl bir kod yazması gerekmektedir?

Zaman Planı:

- **0-3 dakika:** Öğrenciler kodu dikkatle inceler, değişken değerlerini, karar yapısını ve kullanacağı döngünün işlem sırasını analiz eder.

- **3-15 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **15-20 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B6. Belirli Kriterlere Uyan Öğrenci Numaralarını Bulma Programı

Süre: 15 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ3.B6.1. Öğrenci Yönerge Metni

OÇ3.B6.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen OÇ3.B6.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir sınıftaki öğrencilerin numarası 5 ile 25 arasında değişmektedir. Sınıfa giren matematik öğretmeni Sercan, öğrenci numaralarından 3 ile tam bölünebilenleri bulduran bir program yazmak istediğini belirtmiştir.

Siz matematik öğretmeninize kodları yazarak nasıl yardımcı olurdunuz?

Zaman Planı:

- **0-3 dakika:** Öğrenciler kodu dikkatle inceler, değişken değerlerini, karar yapısını ve kullanacağı döngünün işlem sırasını analiz eder.
- **3-15 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **15-20 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

Hafta 3. Çevrim İçi: Ders Materyalleri

OÇ3.B1.1. Öğrenci Yönerge Metni

Göreviniz: Aşağıda bir programcı tarafından yazılmış bir kod parçası verilmiştir. Bu kodu inceleyerek aşağıdaki soruları yanıtlayınız:

1. Programcı bu kodu yazarken neyi amaçlamış olabilir?
2. Kodun akışını (giriş, işlem, çıkış) kendi cümlelerinizle açıklayınız.

OÇ3.B1.2. Kod Bloğu

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

OÇ3.B2.1. Öğrenci Yönerge Metni

Defne, Hüma'ya kısa bir C++ kodu yazmış ve bu kodu çalıştırmadan önce ekranda nasıl bir sonuç oluşacağını tahmin etmesini istemiştir. Defne'nin yazdığı kod aşağıda gösterildiği gibidir.

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

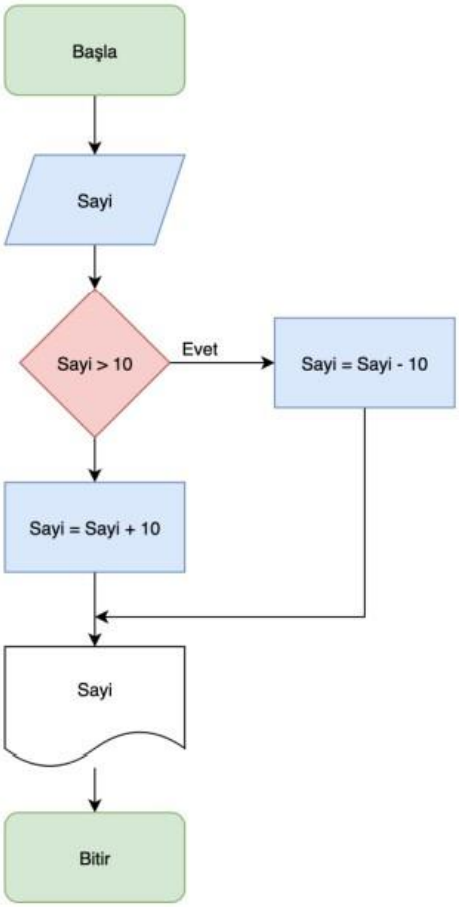
Hüma'nın tahmini doğru olduğuna göre sizce nasıl bir tahminde bulunmuştur?

OÇ3.B2.2. Yanıt

Cevap: 6

OÇ3.B3.1. Öğrenci Yönerge Metni

Akış diyagramı verilen algoritmayı bilgisayar ortamında koda çevirme yarışmasına katılan yarışmacılar, aşağıdaki akış diyagramını nasıl kodlamalıdır?



OÇ3.B3.2. Kod Bloğu

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

OÇ3.B4.1. Öğrenci Yönerge Metni

Göreviniz: Aşağıda bir programcı tarafından yazılmış bir kod parçası verilmiştir.

Bu kodu inceleyerek aşağıdaki soruları yanıtlayınız:

1. Programcı bu kodu yazarken neyi amaçlamış olabilir?

OÇ3.B4.2. Kod Bloğu

```
for (int i = 0; i>10; i++)
{
    cout << "merhaba" << endl;
}
```

OÇ3.B5.1. Öğrenci Yönerge Metni

Göreviniz: Ahmet rastgele 2 sayı üretip büyük olanı ekrana yazdırmak istiyor. Ahmet'in nasıl bir kod yazması gerekmektedir?

OÇ3.B5.2. Kod Bloğu

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
    int sayi1, sayi2;
    srand(time(0));

    sayi1 = rand();
    sayi2 = rand();
    if(sayi1>sayi2)
        cout << sayi1;
    else
        cout << sayi2;
}
```

OÇ3.B6.1. Öğrenci Yönerge Metni

Göreviniz: Bir sınıftaki öğrencilerin numarası 5 ile 25 arasında değişmektedir. Sınıfa giren matematik öğretmeni Sercan, öğrenci numaralarından 3 ile tam bölünebilenleri bulduran bir program yazmak istediğini belirtmiştir.

Siz matematik öğretmeninize kodları yazarak nasıl yardımcı olurdunuz?

OÇ3.B6.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int main() {
    cout << "3 ile tam bolunen ogrenci numaralari:" << endl;

    for (int numara = 5; numara <= 25; numara++) {
        if (numara % 3 == 0) {
            cout << numara << " ";
        }
    }
    cout << endl;
    return 0;
}
```

Hafta 4. Yüz Yüze: 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.
- K7. Diziler üzerinde arama yapabilir.
- K8. Diziler üzerinde basit sıralama yapabilir.

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. Diziler ve katarlar üzerinde arama ve basit sıralama yapar.

Önerilen Ders Akışı

A. Giriş: Ekip İşi (10 dk.)

B. Öğrenme Görevleri

B1. Dizileri Tanıyalım (20 dk.)

B2. Dizilere Değer Verelim (20 dk.)

Ders Arası (10 dk.)

B3. Döngülerle Diziler (25 dk.)

B4. Dizilerle Kodlayalım (25 dk.)

Ders Arası (10 dk.)

B5. Kodlama Ekibi (25 dk.)

B6. Farkı Bul (25 dk.)

Ders Arası (10 dk.)

B7. Arama Yapalım (25 dk.)

B8. Sıralayalım (25 dk.)

A. Giriş: Ekip İşi

Süre: 10 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.

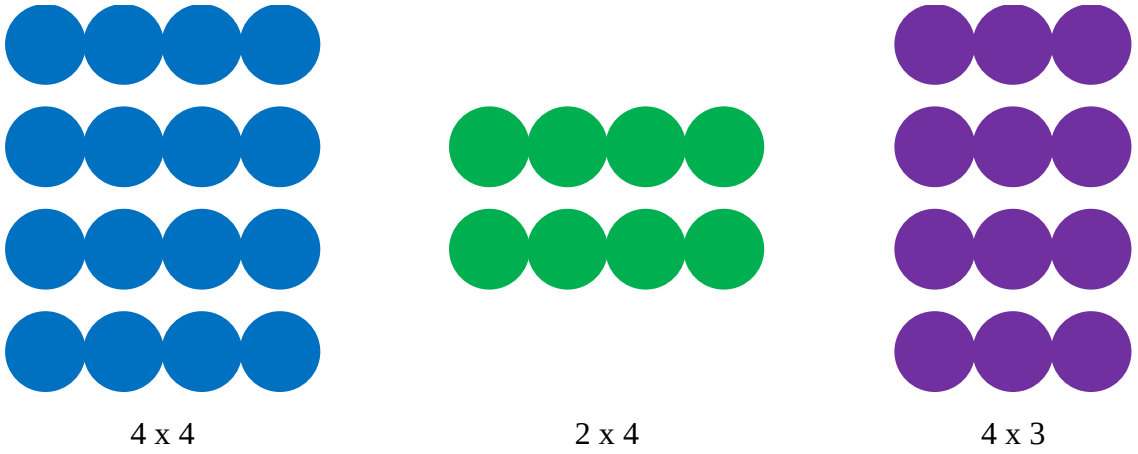
Materyaller: O4.B1.1. Destekleyici Bilgi: Dizileri Tanıyalım

Mavi, yeşil ve mor renkli oyun hamurları

Hazırlık: O4.B1.1. materyali ÖYS üzerinden erişime açılır. Eğitmen ayrıca farklı renklerde oyun hamurları getirir.

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 bitişini hatırlatmak için bir çan ya da zil kullanabilir. Daha sonra dış çemberdekilere sıra geçer 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 farklı renklerde ve çeşitli büyüklüklerde oyun hamurları dağıtılır. Öğrenci çiftlerinden bu hamurlar ile aşağıdaki gibi iki boyutlu diziler oluşturmaları istenir.



Resim 27. *Farklı dizi boyutları*

Eğitmen öğrencilerden oluşturdukları dizilerin boyutlarını küçük ve büyük şeklinde değiştirerek, bunları boyut bilgileri ve indis numaraları ile etiketlemelerini ister. Etkinlik sonunda eğitmen grup çalışmalarını takip eder ve özellikle hatalı diziler üzerinden örnekler alarak konuyu özetler.

Eğitmene Öneriler: Etkinlik materyali olarak oyun hamuru yerine farklı renk ve büyüklükte minik ponponlar, farklı şekillerde kesilmiş renkli kâğıtlar ya da madenî paralar ile de yürütülebilir.

B2. Dizilere Değer Verelim

Süre: 20 dk.

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

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

Materyaller: O4.B2.1. Destekleyici Bilgiler: Dizilere Değer Verelim

O4.B2.2. Görev Kartları: Dizilere Değer Verelim

Hazırlık: Öğrenciler B1 etkinliğindeki oturma düzeninde iç içe çember şeklinde oturmaya devam eder. Ancak bu sefer iç çemberdekiler çok boyutlu dizi, dış çemberdekiler ise tek boyutlu dizi temsilcisi olarak yer değiştirmektedir. O4.B2.2. materyali B1 etkinliğindeki gibi ikiye ayrılır. O4.B2.1. Destekleyici bilgiler ise ÖYS üzerinden erişime açılır.

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 bitişini hatırlatmak için bir çan ya da zil kullanabilir. Daha sonra dış çemberdekilere sıra geçer 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: 25 dk.

Kazanımlar: K4. Dizileri döngü içinde kullanır.

Materyaller: O4.B3.1. Destekleyici Bilgi: Döngülerle Diziler

O4.B3.2. Görev Kartı: Döngülerle Diziler

Hazırlık: Öğrenciler B2 etkinliğindeki oturma düzenindedir. Materyaller ÖYS sisteminde 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 birinci 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 1, iç çemberde yan yana oturan çiftlere ise Görev 2 üzerinde çalışmaları söylenir. Çiftlerin 5 dk. görev üzerinde çalışmaları beklenir. 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 5 dk. süre verilir. Süre sonunda eğitmen tahtaya görevlerin doğru yanıtlarını yazar. Eğitmen çiftlerin çalışmalarını gezerek takip eder.

Eğitmene Öneriler: Öğrencilerin yaptıkları görevlerin doğru yanıtları aşağıdaki gibidir:

Görev 1 Yanıtı

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

{

    int yaslar[5] = {15, 14, 17, 12, 16};

    int i;

    for(i=0; i<5; i++){

        cout << i+1 <<". arkadasimin
yasi: " << yaslar[i] << endl;

    }

    return 0;

}
```

Görev 2 Yanıtı

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

   int yaslar[1][5] = {{15, 14, 17, 12,
16}};

   int k = 1;

   for(int i=0; i<1;i++){

      for(int j=0; j<5; j++,k++){

        cout << k <<". arkadasimin yasi:
" << yaslar[i][j] << endl;

      }

   }

   return 0;

}
```

B4. Dizilerle Kodlayalım

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: O4.B4.1. Görev Kodu

Hazırlık: Öğrenciler B3 etkinliğindeki oturma düzeninde iç içe çember şeklinde oturmaya devam eder ve çiftler hâlinde çalışır. 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:

Görev Kodu 1: Bu görevde program iki dizinin karşılıklı olarak elemanlarını toplayıp, yeni diziye kaydetmeyi amaçlamaktadır.

Görev Kodu 2: Bu görevde program dizideki en büyük elemanı ve bu elemanın indis numarasını yazdırmayı amaçlamaktadır.

Görev Kodu 3: Bu görevde program dizide ardışık olarak tekrar eden elemanları bulmayı amaçlamaktadır.

Görev Kodu 4: Bu görevde program girilen dizide tek ve çift sayı adedini ekrana yazdırmayı amaçlamaktadır.

Görev Kodu 5: Bu görevde program girilen dizide 5'e bölünebilen sayı adedini ekrana yazdırmayı amaçlamaktadır.

B5. Kodlama Ekibi

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: O4.B5.1. Problem

O4.B5.2. Görev Kartı

Hazırlık: 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. 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, yeşil kod satırlarına iç çemberdekilerin, turkuaz kod satırlarına dış çemberdekilerin erişmesi beklenir. Sarı kod satırları ve diğer tamamlamalar ise kodlama ekibinin görevidir.

```
#include<iostream>
using namespace std;
int main()
{
    int i, j, satir=3, sutun=3, matris1[3][3], matris2[3][3], sonuc[3][3];
    for(i = 0; i < satir; i++) {
        for(j = 0; j < sutun; j++) {
            cout << "1. Matrisin [" <<i<< "," <<j<<"]. elemanini giriniz: ";
            cin >> matris1[i][j];
        }
    }
    cout << endl;
    for(i = 0; i < satir; i++) {
        for(j = 0; j < sutun; j++) {
            cout << "2. Matrisin [" <<i<< "," <<j<<"]. elemanini giriniz: ";
            cin >> matris2[i][j];
        }
    }
    cout<<"\nSonuc Matrisi: " << endl;
    for(i = 0; i < satir; i++) {
        for(j = 0; j < sutun; j++) {
            sonuc[i][j] = matris1[i][j] + matris2[i][j];
            cout << sonuc[i][j] << " ";
        }
        cout << endl;
    }
}
```

B6. Farklı Bul

Süre: 25 dk.

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

Materyaller: O4.B6.1. Farkı Bul

Hazırlık: Öğrenciler ikili gruplar hâlinde oturmaktadır. Öğrenme yönetim sisteminden 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.

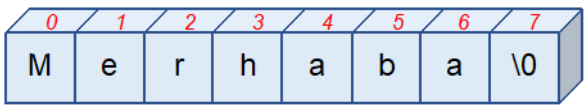
Programlamada metin türünde verilerimizi saklamak için kullanılan özel karakter dizileridir. Katarlar, null ('\0') karakter ile sonlandırılmış tek boyutlu karakter dizileri olarak tanımlanabilir. Aşağıda verilen örnekte, "Merhaba" sözcüğünden oluşan bir katar oluşturulmaktadır. Normalde verilen kelime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

```
char kelime[] = {'M', 'e', 'r', 'h', 'a', 'b', 'a', '\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 kelime[] = "Merhaba";
```

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.



B7. Arama Yapalım

Süre: 25 dk.

Kazanımlar: K7. Diziler üzerinde arama yapabilir.

Materyaller: O4.B7.1. Dizide Arayalım

Hazırlık: Eğitmen, ders öncesinde öğrencilerin ikili gruplar halinde çalışabilmesi için laboratuvardaki bilgisayarları hazırlar. Her bilgisayarda Code::Blocks programını açar ve O4.B7.1. Dizide Arayalım materyalindeki C++ kodunu programa yapıştırarak öğrencilerin kullanımına hazır hale getirir.

Uygulama: Eğitmen, etkinliği öğrenciler için somut bir senaryo ile başlatır: “Bir an için sınıf başkanı olduğunuzu ve yoklama aldığınızı düşünün. Elinizde tüm sınıfın okul numaralarının yazılı olduğu bir liste var. Derse geç kalan bir arkadaşınız kapıdan giriyor ve diyor ki: ‘Öğretmenim, ben geldim! Numaram 45, beni listede işaretler misiniz?’ Arkadaşınızın listede olup olmadığını nasıl kontrol edersiniz?”

Eğitmen, öğrencilerden gelen cevaplar üzerine, “Harika! Tıpkı sizin bir listede bir numarayı aramanız gibi, listedeki her numaraya en baştan başlayarak sırayla bakarız. İşte biz bugün, yazdığımız programa tam olarak bu işi yaptıracağız.” diyerek konuyu koda bağlar.

Eğitmen, öğrencileri ikili gruplara ayırır ve bilgisayarlarının başına geçmelerini söyler. İlk görevleri, ekranlarında açık olan O4.B7.1 kodunu incelemektir. Eğitmen gruplara şu yönergeyi verir: "Önünüzdeki kodu 10 dakika boyunca serbestçe inceleyin. Kodu çalıştırın, hem listede olan hem de olmayan farklı okul numaraları girerek ne yaptığını anlamaya çalışın. Arkadaşınızla kodun hangi bölümünün ne işe yaradığını tartışın."

Sürenin sonunda eğitmen, kodu çalıştırarak ve senaryoya geri dönerek tartışmayı başlatır: “Evet arkadaşlar, keşifleriniz neler? 45 numaralı öğrenciyi listenin başında bulduysak, listenin sonuna kadar bakmaya devam etmemiz gerekir mi? Kodun içinde bu işi durduran bir komut fark ettiniz mi?” sorusuyla break; komutunun verimliliğini tartışmaya açar. Ardından, ‘bool bulundu’ diye bir not defterimiz olmasaydı, aradığımız öğrencinin listede olmadığını program nasıl anlardı? sorusuyla öğrencilerin "bayrak" değişkeninin önemini kendi bulgularıyla paylaşmalarını sağlar. Eğitmen, bu değişkeni ve ilgili satırları geçici olarak silip kodu tekrar çalıştırarak, bu "bayrak" değişkeninin programın hafızası olarak nasıl çalıştığını uygulamalı olarak gösterir. Böylece tartışmayı yöneterek ana fikirleri pekiştirir.

Eğitmene Öneriler: Tartışma boyunca "yoklama listesi", "aranan öğrenci numarası", "listeyi kontrol etmek" gibi somut ifadeler kullanmak, ortaokul öğrencilerinin soyut kod kavramlarını anlamasını kolaylaştırır. Tahtaya küçük bir dizi (liste) çizin ve bir for döngüsünün her adımında i değişkeninin hangi indeksi (listedeki hangi sırayı) kontrol ettiğini oklarla gösterin. Bu, döngü mantığını somutlaştırır. Bu etkinlikteki temel amaç, bir dizi içinde sıralı arama (linear search) mantığını öğretmektir. C++ dilinin karmaşık syntax detaylarına girmek yerine, "döngü kur-karşılaştır-bulunca dur-bulamazsan haber ver" mantığına odaklanın.

B8. Sıralayalım

Süre: 25 dk.

Kazanımlar: K8. Diziler üzerinde basit sıralama yapabilir.

Materyaller: O4.B8.1. Dizi Sıralama

Hazırlık: Eğitmen, sınıfın ortasında öğrencilerin rahatça hareket edebileceği bir alan oluşturur. Projeksiyon cihazına bağlı bilgisayarında Code::Blocks programını açar ve O4.B8.1. Dizi Sıralama materyalindeki C++ kodunu programa yapıştırarak hazır hale getirir.

Uygulama: Eğitmen, etkinliği tüm sınıfı harekete geçirecek bir oyunla başlatır: "Haydi hep birlikte bir oyun oynayalım! Bana gönüllü 5 arkadaş lazım." Eğitmen, gönüllü öğrencileri sınıfın önüne karışık bir sırada dizer ve diğer öğrencilere döner: "Arkadaşlar, önünüzde karışık bir 'sayı dizisi' gibi duruyorlar. Amacımız, bu diziyi boy sırasına göre kısadan uzuna doğru sıralamak. Ama bir kuralımız var: Her seferinde sadece iki kişinin yerini değiştirebilirsiniz. Sınıf olarak, bu sıralamayı nasıl yapardınız? Haydi, en kısa arkadaşımızı bulup en başa getirelim, komutlar sizde!"

Eğitmen, sınıfın yönlendirmesiyle ("Arda'yı Karaca ile karşılaştır," "Defne en kısa, onu en baştaki Duru ile yer değiştir" gibi) ilk elemanı bulma ve yer değiştirme işlemini canlandırır. Bu işlemi ikinci ve üçüncü en kısa öğrenciler için de tekrarlatır. Bu eğlenceli canlandırma tamamlandıktan sonra eğitmen, "Harika bir iş çıkardınız! İşte az önce hep birlikte yaptığınız bu işleme bilgisayar dünyasında 'Seçmeli Sıralama' deniyor. Şimdi gelin, sınıfça yaptığımız bu oyunun C++ dilindeki karşılığına, yani koduna bakalım." diyerek projeksiyonda hazır olan O4.B8.1 kodunu gösterir.

Eğitmen, tartışmayı oyuna bağlayarak şu sorularla yönetir:

- "Bu kodda iç içe iki döngü var. Oyunda her seferinde en kısa kişiyi bulmak için bütün sırayı kontrol ediyorduk. Sizce kodun içindeki bu 'arama' işini hangi döngü yapıyor olabilir?" (İç döngünün amacını keşfettirir.)
- "Peki, en kısa kişiyi bulduktan sonra yaptığımız 'yer değiştirme' işlemini kodun hangi bölümü gerçekleştiriyor?" (Takas/swap bölümünü bulmalarını sağlar.)
- "Kodun içindeki o yer değiştirme bölümü hiç olmasaydı ne olurdu? Program yine de en küçük sayıyı bulur muydu? Ama dizi sıralanır mıydı?" (Takas işleminin önemini vurgular.)

Eğitmen, öğrencilerden farklı ve küçük sayılı (4-5 elemanlı) dizi örnekleri vermelerini ister ve bu dizilerle kodun adımlarını tahtada izleyerek hangi elemanın seçildiğini ve nasıl yer değiştirildiğini birlikte inceler.

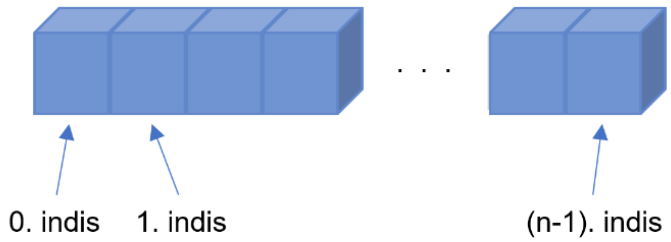
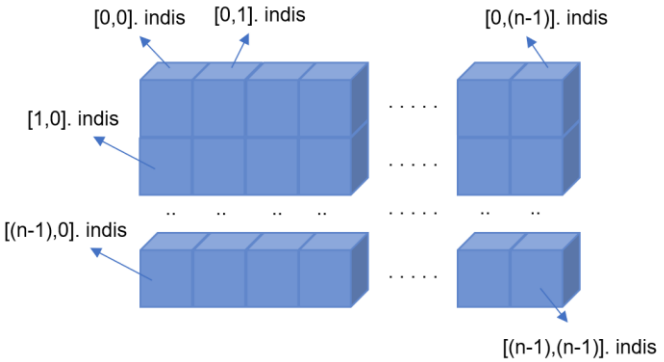
Eğitmene Öneriler: Bu fiziksel canlandırma, soyut bir sıralama algoritmasını ortaokul öğrencilerinin somut bir deneyimle ilişkilendirmesi için oldukça etkilidir. Etkinliği eğlenceli ve dinamik tutmak önemlidir. Kod incelemesi sırasında, öğrencilerden gelen farklı ve küçük dizi örneklerini tahtaya yazarak dış döngünün her adımında dizinin nasıl değiştiğini adım adım göstermek, algoritmanın anlaşılmasını çok kolaylaştıracaktır. Ortaokul seviyesinde minIndex, temp gibi değişkenlerin isimlerinin ne anlama geldiğini basitçe açıklamak yeterlidir. Önemli olan, öğrencilerin "en küçüğü bul ve başa koy, sonra kalandan devam et" temel mantığını kavramasıdır.

Hafta 4. Yüz Yüze: Ders Materyalleri

O4.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. 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  İndisler: **0** **1** **2** **3** **4**	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. 

O4.B2.1. Destekleyici Bilgi: Dizilere Değer Verelim

Tek Boyutlu Diziler: Tek bir veri türü içeren dizilerdir.	***Çok Boyutlu Diziler:*** Bir veya daha fazla dizi içeren dizilerdir.
veri_tipi dizi_adi[eleman_sayisi]; Beş adet öğrenciye ait öğrenci numarasını saklamak için oluşturulan "numaralar" dizisine değer atamak için: **int** numaralar[5]; **Uyarı!** Numaralar dizisi bir boyutludur ve 5 elamana sahiptir. Dizi içerisindeki herhangi bir elemana ulaşmak için dizi adı ve ardından eleman indis numarasını içeren köşeli parantezler kullanılır. Yukarıda tanımlanan numaralar dizisinin beşinci elemanına erişmek için şu şekilde bir ifade yazılır: numaralar[4]	veri_tipi dizi_adi[boyut1][boyut2]…[boyutN]; 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. Dizi içerisindeki herhangi bir elemana ulaşmak için dizi adı ve ardından eleman indis numarasını içeren köşeli parantezler kullanılır. sayilar [2][2] dizisi için eleman indis numarası; sayilar[0][0] : birinci satır ilk eleman sayilar[0][1] : birinci satır ikinci eleman sayilar[1][0] : ikinci satır birinci eleman sayilar[1][1] : ikinci satır ikinci eleman
veri_tipi dizi_adi[eleman_sayisi] = {değer1, değer2, değer3, …}; Beş adet öğrenci adlarının ilk harflerini saklamak için oluşturulan "isim" dizisine değer atamak için: **char** isim [5] = {'H', 'K', 'A', 'R', 'S'}; **char** isim [] = {'H', 'K', 'A', 'R', 'S'}; **Uyarı!** Dizinin ilk değerlerini bu şekilde atarsanız dizi eleman sayısı alanını boş bırakabilirsiniz. Derleyici, bu şekilde dizinin elemanları verildiği zaman dizi boyutunun ne olacağına kendisi karar verecektir.	veri_tipi dizi_adi[boyut1][boyut2]…[boyutN] = {değer1, değer2,..değerN}; 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.

```
int numara[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 [2][3][5];
```

Yukarıda tanımlanan *sayilar* dizisi 2 * 3 * 5 = 30 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.

O4.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[2][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!
 	
`char harfler[] = {'H', 'K', 'A', 'R', 'S'};`
 	
Örnekteki harfler dizisinde 2. indis'ten sonra gelen dizi elemanı hangisidir?
	Sıra Sizde!
	
	`char harfler[2] [5] = {{'H', 'K', 'A', 'R', 'S'},`
	`{'M', 'N', 'P', 'S', 'D'} };`
	
	Örnekteki harfler dizisinde (1,2) numaralı indis'ten (2.satır 3. sütun olmakta) önce gelen dizi elemanı hangisidir?

O4.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*

O4.B3.2. Görev Kartı: Döngülerle Diziler

Problem: 5 arkadaşın yaş bilgilerini değişkende saklama ve ekrana yazdırma

```
#include <iostream>
using namespace std;
int main(){
    int yas1 = 15;
    int yas2 = 14;
    int yas3 = 17;
    int yas4 = 12;
    int yas5 = 16;
    cout << "1. arkadasimin yasi: " << yas1 << endl;
    cout << "2. arkadasimin yasi: " << yas2 << endl;
    cout << "3. arkadasimin yasi: " << yas3 << endl;
    cout << "4. arkadasimin yasi: " << yas4 << endl;
    cout << "5. arkadasimin yasi: " << yas5 << endl;
    return 0;
}
```

Kodun Çıktısı:

```
1. arkadasimin yasi: 15
2. arkadasimin yasi: 14
3. arkadasimin yasi: 17
4. arkadasimin yasi: 12
5. arkadasimin yasi: 16
```

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.

O4.B4.1. Görev Kodu

```
#include <iostream>
using namespace std;
int main(){
    int dizi1[5], dizi2[5], dizi3[5];
    int i;
    for(i=0; i<5;i++){
        cout << "1. Dizinin " << (i+1) << ". elemanini giriniz: ";
        cin >> dizi1[i];
    }
    cout << endl;
    for(i=0; i<5;i++){
        cout << "2. Dizinin " << (i+1) << ". elemanini giriniz: ";
        cin >> dizi2[i];
    }
    cout << endl;
    for(i=0; i<5;i++){
        dizi3[i] = dizi1[i] + dizi2[i];
        cout << "3. Dizinin " << (i+1) << ". elemani: " << dizi3[i] << endl;
    }
    return 0;
}
```

Görev Kodu 1

```cpp
#include <iostream>
using namespace std;
int main()
{
    int sayilar[] = {19, 11, 21, 13, 15};
    int i, maks, yer, n = 5;
    cout << "Dizi: ";
    for(i=0; i < n; i++)
        cout << sayilar[i] << " ";
    maks = sayilar[0];
    yer = 0;
    for(i=1; i < n; i++){
        if(maks < sayilar[i]){
            maks = sayilar[i];
            yer = i;
        }
    }
    cout << "\nDizinin en buyuk elemani " << yer <<". indisteki " << maks;
  return 0;
}
```

Görev Kodu 2

```cpp
#include <iostream>
using namespace std;
int main()
{
    int dizi[] = {12, 67, 78, 45, 78, 78, 32, 16, 16, 57};
    int i, j, n = 10;
    cout << "Dizi: ";
    for(i = 0; i < n; i++)
        cout << dizi[i] << " ";

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

Görev Kodu 3

```cpp
#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;
}
```

Görev Kodu 4

```cpp
#include <iostream>
using namespace std;
int main()
{
    int sayilar[100];
    int i, n, 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]%5==0)
            bol5++;
    }
    cout << "\n5 ile bolunebilen eleman sayisi: " << bol5;
    return 0;
}
```

Görev Kodu 5

O4.B5.1. Problem

Haftalık oyun oynayan üç arkadaşın oyun sonunda aldıkları puanların ilk iki haftası aşağıda verilmiştir. Üçüncü hafta ise ilk iki haftanın puan toplamından oluşmalıdır. Buna göre bu üç arkadaşın üçüncü hafta puanlarını ekrana yazdıran programı tasarlayınız.

Hafta 1:

Oyuncu	Puan		
Ayşe	1	0	1
Burcu	0	0	0
Can	0	2	0

Hafta 2:

Oyuncu	Puan		
Ayşe	1	1	1
Burcu	2	0	0
Can	0	2	0

Hafta 3: ??

Oyuncu	Puan		
Ayşe			
Burcu			
Can			

O4.B5.2. Görev Kartı

İç Çember Görev Kartı

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

Dış Çember Görev Kartı

Problemde yer alan ikinci hafta skorlarını ekrana yazdıran kod satırlarını yazınız.

Kodlama Ekibi Görev Kartı

İlk hafta ve ikinci hafta skorlarını ekrana yazdıran kod satırlarını birleştirin. Bu iki skorların toplamını ekrana yazdıran yeni kod satırlarını oluşturun. Şimdi elinizde temel problemi çözecek kod satırı parçaları vardır. Bunları bir araya getirerek temel problemin çıktısını oluşturacak programı çalıştırın.

O4.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;
}
```

O4.B7.1. Dizide Arayalım

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

int main() {

    int sayilar[] = {10, 25, 30, 45, 50, 60};
    int diziBoyutu = 6;

    int aranan;
    cout << "Aranacak sayiyi girin: ";
    cin >> aranan;

    bool bulundu = false;

    for (int i = 0; i < diziBoyutu; ++i) {
        if (sayilar[i] == aranan) {
            cout << aranan << " bulundu! Konumu (indeksi): " << i << endl;
            bulundu = true;
            break;
        }
    }

    if (!bulundu) {
        cout << aranan << " dizide bulunamadi." << endl;
    }

    return 0;
}
```

O4.B8.1. Dizi Sıralama

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

int main() {
    int sayilar[] = {40, 10, 50, 20, 30};
    int n = 5;

    cout << "Sirasiz dizi: ";
    for (int i = 0; i < n; i++) {
        cout << sayilar[i] << " ";
    }
    cout << endl;

    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;

        for (int j = i + 1; j < n; j++) {
            if (sayilar[j] < sayilar[minIndex]) {
                minIndex = j;
            }
        }

        if (minIndex != i) {
            int temp = sayilar[i];
            sayilar[i] = sayilar[minIndex];
            sayilar[minIndex] = temp;
        }
    }

    cout << "Sirali dizi: ";
    for (int i = 0; i < n; i++) {
        cout << sayilar[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Hafta 4. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftanın çevrim içi dersinin amacı, öğrencilerin yüz yüze eğitimde öğrendikleri dizi ve katar yapılarını kullanarak problem çözme becerilerini pekiştirmektir. Bu kapsamda öğrencilerin; dizileri tanımlayıp elemanlarına erişmesi, döngü yapıları aracılığıyla diziler üzerinde işlem yapması, katar veri tipini etkin kullanması ve bu iki temel veri yapısı arasındaki işlevsel farkları uygulamalı olarak pekiştirmeleri hedeflenmektedir.

Etkinlik Uygulama Ortamı

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır. Bireysel çalışma ürünleri ise dijital tartışma panosu aracılığıyla toplanacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Isınma Turu (10 dk.)

B. Öğrenme Görevleri

B1. Bireysel Macera: Kodun Sırrını Çöz (15 dk.)

B2. İkili Görev: Eksik Parçayı Bul (25 dk.)

Ders Arası (10 dk.)

B3. Grup Meydan Okuması: Bereketli Tarla (40 dk.)

B4. Değerlendirme ve Kapanış (10 dk.)

A. Giriş: Isınma Turu

Süre: 10 dk.

Hazırlık: Eğitmen, ders öncesinde öğrencilerin derse katılımı sırasında çalmak üzere telifsiz, enerjik bir enstrümantal müzik bağlantısı hazırlar.

Uygulama: Dersin ilk 2-3 dakikasında, öğrenciler çevrim içi sınıfa katılırken eğitmen tarafından hazırlanan müzik çalınır. Bu, derse enerjik bir başlangıç yapılmasını sağlar ve tüm öğrencilerin katılması için zaman tanır. Müzik bittikten sonra eğitmen, güler yüzle öğrencileri karşılar ve dersin amacını onlara hitap eden bir dille açıklar: "Hoş geldiniz kod dedektifleri! Bugün, geçen hafta yüz yüze derste öğrendiğimiz dizilerle harika şeyler yapacağız. Tıpkı bir oyunun yeni seviyelerini açmak gibi, bildiklerimizi kullanarak eğlenceli kodlama maceralarına atılacağız!"

Ardından, eğitmen öğrencilere sohbet ekranını kullanacakları interaktif bir soru yöneltir: "Şimdi hızlı bir hafıza oyunu zamanı! Geçen haftaki yüz yüze dersimizden 'diziler' konusuyla ilgili aklınıza gelen ilk kelime veya emoji nedir? Haydi, sohbet ekranına yazın, görelim!" Öğrenciler cevaplarını yazarken 1 dakika kadar kısa bir süre tanınır. Süre sonunda eğitmen, gelen cevapları sesli bir şekilde ve pozitif bir tonla okur: "Harika cevaplar geliyor! 'Liste' diyen var, 'kutucuklar' diyen var, bir arkadaşımız tren emojisi koymuş, süper! Evet, hepsi dizileri hatırlatıyor." gibi yorumlarla öğrencilerin ön bilgileri pekiştirilir ve bir sonraki etkinliğe yumuşak bir geçiş yapılır.

B. Öğrenme Görevleri

B1. Bireysel Macera: Kodun Sırrını Çöz

Süre: 15 dk.

Materyaller: OÇ4.B1.1. Görev Kodu: Kodun Sırrı Ne?

Hazırlık: Eğitmen, OÇ4.B1.1 materyalindeki kod bloğunu ve anket sorusunu sunumuna veya paylaşacağı bir metin belgesine önceden hazırlar. Sanal sınıf platformunun anket özelliğini kullanıma hazır hale getirir.

Uygulama: Etkinlik, eğitmenin ekranını paylaşarak öğrencilere seslenmesiyle başlar: "Şimdi her biriniz birer kod dedektifi olacaksınız. Ekrana gizemli bir kod getireceğim ve sizden bu kodun sırrını çözmenizi, yani çalıştırıldığında ekrana ne yazacağını tahmin etmenizi isteyeceğim."

Eğitmen, OÇ4.B1.1 materyalindeki kodu ve soruyu ekrana yansıtır. Ardından sanal sınıfın anket aracını başlatır ve öğrencilere şu soruyu yöneltir:

Anket Sorusu: Gördüğünüz bu C++ kodu çalıştırıldığında ekrana hangi çıktıyı yazdırır?

A) 78 bulundu! Konumu (indeksi): 4

B) 78 bulundu! Konumu (indeksi): 5

C) bulundu! Konumu (indeksi): 4

D) Kod hata verir, çalışmaz.

Öğrencilere düşünmeleri ve anketi yanıtlamaları için 2-3 dakika süre tanınır.

Anket süresi dolduktan sonra eğitmen sonuçları kısaca değerlendirir ve ardından IDE üzerinde kodu çalıştırarak doğru çıktıyı gösterir. Sonrasında en önemli kısım olan açıklamaya geçer:

"Harikasınız dedektifler! Doğru cevap A seçeneğiydi! Peki, neden 5 değil de 4 olduğunu hatırlayan var mı? Çünkü bilgisayarlar saymaya her zaman sıfırdan başlar! Bir dizideki ilk kutunun adresi 0, ikincinin 1, üçüncünün 2 diye devam eder. Bu yüzden listemizde 5. sırada duran 78 sayısının adresi, yani indisi 4'tür."

Eğitmen, etkinliğin kalan süresini bu "sıfır tabanlı indeksleme" kuralını pekiştirmek ve öğrencilerin sorularını yanıtlamak için kullanır.

B2. İkili Görev: Eksik Parçayı Bul

Süre: 25 dk.

Materyaller: OÇ4.B2.1. Görev Tanımı: En Yüksek Puan Kimde?

Hazırlık: Eğitmen, çevrim içi platformunda 2 kişilik çalışma odaları hazırlar. Öğrencilerin tamamladıkları kodları gönderecekleri bir Padlet panosu oluşturur ve paylaşım linkini hazır bulundurur.

Uygulama: Eğitmen, bir önceki etkinliğin ardından öğrencilere yeni görevlerini duyurur: "Kod dedektifleri, şimdi bir takım arkadaşıyla güçlerinizi birleştirme zamanı! Size bir bilgisayar oyunundaki en yüksek puanı bulan bir kod vereceğim ama... en önemli parçası eksik! Göreviniz, takım arkadaşınızla birlikte bu eksik parçayı bulup kodu tamamlamak."

Eğitmen, ekranını paylaşarak OÇ4.B2.1 materyalindeki eksik kod bloğunu ve senaryoyu öğrencilere gösterir. Problemi kısaca açıklar:

> *"Bir grup arkadaş oyun oynamış ve puanları bir listede (dizide) tutuluyor. Amacımız, bu listedeki en yüksek puanı bulmak. Kodumuz ilk puanı 'en yüksek' kabul ederek başlıyor, sonra diğerlerini tek tek kontrol ediyor. Sizin göreviniz ise bu kontrolü yapacak olan if koşulunu yazmak!"*

Ardından eğitmen, öğrencileri 2 kişilik çalışma odalarına gönderir ve görev tanımını sohbet üzerinden paylaşır. Öğrencilere, odalarında bir arkadaşlarının ekranını paylaşarak Code::Blocks üzerinde birlikte çalışmaları için 15 dakika süre verir.

Süre sonunda eğitmen, öğrencilerden tamamladıkları kodun ekran görüntüsünü veya sadece eksik if bloğunu Padlet panosuna yüklemelerini ister. Tüm öğrenciler ana odaya döndükten sonra, eğitmen Padlet'e yüklenen birkaç farklı çözümü ekrana yansıtır, doğru çözümü gösterir ve kodun çalışma mantığını basitçe açıklar.

Eğitmene Öneriler: Bu etkinlik, öğrencilerin bir algoritmanın mantığını takım arkadaşlarıyla tartışarak çözmesini hedefler. Süreci daha verimli hale getirmek için aşağıdaki noktalara dikkat edebilirsiniz:

- **Yönlendirici Sorular:** Öğrenciler çalışma odalarında zorlanırsa, onlara doğrudan cevabı vermek yerine şu gibi sorularla yol gösterebilirsiniz:
    - "Döngünün içindeyken, o anki puanı (puanlar[i]) ne ile karşılaştırmamız gerekiyor?"
    - "Eğer o anki puan, şimdiye kadarki en yüksek puandan daha büyükse ne yapmalıyız?"
    - "Yeni şampiyonu enYuksekPuan kutusuna nasıl atarız?"
- **Padlet Kullanımı:** Süre sonunda Padlet'e gönderilen farklı cevapları (doğru veya yanlış) ekrana yansıtarak öğrencilerle birlikte "Neden bu kod çalışır?" veya "Neden bu kod çalışmaz?" diye tartışın. Bu, hatalar üzerinden öğrenmeyi sağlar ve çok etkilidir.
- **Doğru Cevap ve Açıklaması:** Etkinlik sonunda tüm öğrencilerin anladığından emin olmak için doğru cevabı ve mantığını net bir şekilde açıklayın.

Görevin Doğru Cevabı (Eksik if Bloğu):

```
if (puanlar[i] > enYuksekPuan) {
   enYuksekPuan = puanlar[i];
}
```

Cevabın Açıklaması:

Bu kodun mantığını öğrencilere şu şekilde açıklayabilirsiniz:

1. **Karşılaştırma:** if (puanlar[i] > enYuksekPuan) satırı ile bilgisayara diyoruz ki: "Hey, şu an baktığın puanlar[i] (listedeki sıradaki puan), bizim şimdilik şampiyon ilan ettiğimiz enYuksekPuan'dan daha büyük mü?"
2. **Güncelleme:** Eğer cevap "Evet, daha büyük!" ise, { } parantezlerinin içindeki enYuksekPuan = puanlar[i]; komutu çalışır. Bu komut da şunu yapar: "O zaman eski şampiyonu unut, artık yeni şampiyonumuz bu puan!"
3. **Devam Etme:** Eğer cevap "Hayır, daha büyük değil" ise, if bloğu tamamen atlanır ve döngü bir sonraki puanı kontrol etmek için devam eder.

B3. Grup Meydan Okuması: Bereketli Tarla

Süre: 40 dk.

Materyaller: OÇ4.B3.1. Görev Kartı: Bereketli Tarla

Hazırlık: Eğitmen, çevrim içi platformunda 4 kişilik çalışma odaları hazırlar. Grupların tamamladıkları kod projelerini gönderecekleri bir Padlet panosu oluşturur ve paylaşım linkini hazır bulundurur.

Uygulama: Ders arasından sonra eğitmen, dersin final görevini yeni ve keyifli bir senaryo ile duyurur: “Final görevimiz için 4 kişilik takımlara ayrılacaksınız. Göreviniz genç bir çiftçiye yardım etmek! Çiftçimiz, tarlasından 5 gün boyunca topladığı çilek sepetlerini bir listeye kaydetmiş. Haftanın ne kadar verimli geçtiğini anlamak için bizden bu listenin analizini yapmamızı istiyor. Haydi, takımlar halinde çiftçimize yardım edelim!”

Eğitmen, öğrencileri 4 kişilik çalışma odalarına gönderir ve OÇ4.B3.1 görev kartını sohbet üzerinden paylaşır. Gruplara, aralarından bir kişinin ekranını paylaşarak Code::Blocks üzerinde ortaklaşa çalışmaları için 30 dakika süre verir.

Süre sonunda gruplardan tamamladıkları kodun ekran görüntüsünü Padlet panosuna yüklemeleri istenir ve tüm öğrenciler ana odaya geri döner.

Eğitmene Öneriler: Bu senaryo, programlamanın sadece oyunlarda değil, tarım gibi gerçek hayat alanlarında da veri analizi için nasıl kullanılabileceğini göstermek için iyi bir fırsattır. Eğitmen öğrencileri teşvik amaçlı çalışma odalarını ziyaretinde yönlendirici sorular sorabilir.

- "GÖREV 1 için for döngüsünü nasıl başlatmalı, nerede bitirmeli ve nasıl artırmalıyız?"
- "GÖREV 2'deki toplama satırını nasıl yazarız? Hangi değişkeni kullanacağız?"
- "GÖREV 3'te ortalamayı hesaplarken ondalıklı sonuç almak için neye dikkat etmeliyiz?"

İskelet Kodun Tamamlanmış Hali: Etkinlik sonunda tüm sınıfla birlikte incelenecek doğru çözüm aşağıdaki gibidir.

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

int main() {
    int cilekHasadi[] = {25, 32, 28, 40, 35};
    int gunSayisi = 5;
    int toplamHasiat = 0;
    int enIyiGun = cilekHasadi[0];
    float ortalamaHasiat = 0;

    // --- GOREV 1: Tamamlandi ---
    for (int i = 0; i < gunSayisi; i++) {
        // --- GOREV 2: Tamamlandi ---
        toplamHasiat += cilekHasadi[i];
        if (cilekHasadi[i] > enIyiGun) {
            enIyiGun = cilekHasadi[i];
        }
    }

    // --- GOREV 3: Tamamlandi ---
    ortalamaHasiat = (float)toplamHasiat / gunSayisi;

    cout << "Toplam Hasilat: " << toplamHasiat << " sepet" << endl;
    cout << "Gunluk Ortalama Hasilat: " << ortalamaHasiat << " sepet" <<
endl;
    cout << "En Iyi Gunun Hasilati: " << enIyiGun << " sepet" << endl;

    return 0;
}
```

B4. Değerlendirme ve Kapanış

Süre: 10 dk.

Uygulama: "Bereketli Tarla" görevinin tamamlanmasının ardından tüm öğrenciler ana odaya döner ve eğitmen kapanış bölümünü başlatır.

Eğitmen, Padlet panosunu ekranına yansıtır ve coşkulu bir ses tonuyla, "Harika iş çıkardınız çiftçiler! Padlet'e yüklediğiniz projelere bakıyorum da, hepsi birbirinden başarılı. Şimdi aranızdan gönüllü olan 1-2 takım, çözümünü bize kısaca anlatmak ister mi? Nasıl başardınız?" diyerek sözü öğrencilere bırakır.

Gönüllü takımlar, çözümlerini kısaca paylaştıktan sonra eğitmen tüm grupları tebrik eder ve dersi özetler: “Bugün hep birlikte harika bir iş başardık! Dizilerin içine bilgileri nasıl kaydedeceğimizi, bir döngüyle bu bilgileri tek tek nasıl gezeceğimizi ve toplama, ortalama bulma gibi hesaplamaları nasıl yapacağımızı öğrendik. Hatta aralarındaki en verimli günü bile bulduk! Artık elinizde çok güçlü bir kodlama aracı var.”

Dersi, bir sonraki haftanın konusunu merak uyandıracak şekilde duyurarak bitirir: “Haftaya daha da heyecanlı bir konuya geçiyoruz: Fonksiyonlar! Kendi sihirli komutlarımızı yazarak kodlarımızı süper güçlerle donatmayı öğreneceğiz. Haftaya görüşmek üzere, kodlama maceracıları!”

Hafta 4. Çevrim İçi: Ders Materyalleri

OÇ4.B1.1. Görev Kodu: Kodun Sırrı Ne?

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

int main() {
    // Okul numaralarinin oldugu bir liste (dizi)
    int okulNumaralari[] = {17, 29, 44, 65, 78, 40, 53, 12};

    // Aradigimiz ogrencinin numarasi
    int arananNumara = 78;

    // Dongu ile listedeki numaralari bastan sona kontrol et
    for (int i = 0; i < 8; ++i) {
        // Eger aranan numara, listenin o anki elemanina esitse
        if (okulNumaralari[i] == arananNumara) {
            // Ekrana numaranin bulundugunu ve adresini (indisini) yazdir
            cout << arananNumara << " bulundu! Konumu (indeksi): " << i
<< endl;
        }
    }
    return 0;
}
```

OÇ4.B2.1. Görev Tanımı: En Yüksek Puan Kimde?

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

int main() {
    // Arkadas grubunun oyunda aldigi puanlari iceren bir dizi
    int puanlar[] = {1250, 2100, 1800, 950, 2300, 1500};
    int oyuncuSayisi = 6;

    // En yuksek puani saklamak icin bir degisken olusturalim.
    // Simdilik ilk oyuncunun puanini en yuksek kabul edelim.
    int enYuksekPuan = puanlar[0];

    // Diger oyuncularin puanlarini kontrol etmek icin bir dongu kuralim.
    for (int i = 1; i < oyuncuSayisi; ++i) {

        // GOREV: Buraya o anki oyuncunun puani (puanlar[i]) ile
        // enYuksekPuan degiskenini karsilastirip, gerekirse enYuksekPuan
        // degiskenini guncelleyecek 'if' kosulunu yaziniz.

        // --- KODUNUZU BURAYA YAZIN ---


        // ---------------------------------
    }

    // Dongu bittikten sonra bulduğumuz en yuksek puani ekrana yazdir
    cout << "Bu turdaki en yuksek puan: " << enYuksekPuan << endl;

    return 0;
}
```

OÇ4.B3.1. Görev Tanımı: En Yüksek Puan Kimde?

Senaryo: Genç bir çiftçi, tarlasından 5 gün boyunca topladığı mis kokulu çilekleri sepetlere doldurup bir listeye kaydetti. Çiftçimiz, haftanın ne kadar bereketli geçtiğini görmek için sizden yardım istiyor.

Göreviniz: Aşağıda verilen **iskelet kodu** takımınızla birlikte tamamlayarak programı çalışır hale getirin! Yorum satırları (// --- GÖREV ... ---) ile belirtilen eksik kısımları doldurmanız yeterli.

Beklenen Ekran Çıktısı:

```
Toplam Hasilat: 160 sepet

Gunluk Ortalama Hasilat: 32 sepet

En Iyi Gunun Hasilati: 40 sepet
```

Tamamlanacak İskelet Kod:

```
#include <iostream>
using namespace std;
int main() {
    // Verilerimiz burada hazir
    int cilekHasadi[] = {25, 32, 28, 40, 35};
    int gunSayisi = 5;

    // Sonuclari saklayacagimiz bos degiskenler
    int toplamHasilat = 0;
    int enIyiGun = cilekHasadi[0]; // Ilk gunu en iyi kabul ediyoruz
    float ortalamaHasiat = 0;

    // --- GOREV 1: Diziyi bastan sona gezecek FOR dongusunu buraya
yazin! ---
    // for ( ... ) {

    // --- GOREV 2: Dongunun icinde yapilacak islemleri buraya yazin! ---
        // 1. Toplam Hasilati artirin.
        // 2. O anki hasilatin en iyi gunden daha iyi olup olmadiigini
kontrol edin.

    // } // for dongusu burada bitiyor

    // --- GOREV 3: Ortalama Hasilati burada hesaplayin! ---
    // ortalamaHasilat = ...

    // Sonuclari ekrana yazdiran hazir kod
    cout << "Toplam Hasilat: " << toplamHasilat << " sepet" << endl;
    cout << "Gunluk Ortalama Hasilat: " << ortalamaHasiat << " sepet" <<
endl;
    cout << "En Iyi Gunun Hasilati: " << enIyiGun << " sepet" << endl;
    return 0;
}
```

Hafta 5. Yüz Yüze: Fonksiyonlar

Kazanımlar

- K1. C++ programlama dilinde fonksiyon tanımlayabilir.
- K2. C++ programlama dilinde fonksiyon oluşturmayı bilir.
- K3. C++ programlama dilinde fonksiyon çağırmayı bilir.

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 (50 dk.)

   Ders Arası (10 dk.)

   B3. Fonksiyonları Kullanarak Kodlama Yapalım (50 dk.)

   Ders Arası (10 dk.)

   B4. Gelişmiş Fonksiyon Örnekleri Kodluyorum (50 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. C++ programlama dilinde fonksiyon tanımlayabilir.

Materyaller: O5.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.

Gerçek hayatta karmaşık bir işi daha kolay yönetebilmek için onu küçük parçalara bölmek oldukça etkili bir yöntemdir. Bu yaklaşım, bilgisayar programlamada da aynı şekilde geçerlidir. Büyük ve karmaşık bir programı küçük, anlamlı parçalara ayırmak hem geliştirme sürecini kolaylaştırır hem de hataların daha hızlı bulunmasını ve giderilmesini sağlar.

Programlamada bu parçalama işlemi fonksiyonlar aracılığıyla gerçekleştirilir. Fonksiyonlar, belirli bir görevi yerine getiren kod bloklarıdır. Özellikle birden fazla kez kullanılacak işlemleri bir fonksiyon içinde tanımlamak, kodun tekrar tekrar yazılmasını önler ve yazılımın genel yapısını daha derli toplu hale getirir.

Fonksiyon kullanmanın sağladığı bazı önemli avantajlar şunlardır:

Program Takibi Kolaylaşır: Bir program fonksiyonlar sayesinde bölümlere ayrıldığında, her bir bölüm ne iş yaptığını açıkça gösterir. Bu da programın genel yapısını anlamayı kolaylaştırır. Fonksiyonlar, programın mantıksal akışını daha okunabilir hale getirir ve özellikle büyük projelerde kodun yönetimini ciddi anlamda kolaylaştırır.

Hataların Çözümü Daha Kolay Hale Gelir: Her fonksiyon tek bir işi yapacak şekilde tasarlandığı için bir hata oluştuğunda tüm programı incelemek yerine yalnızca ilgili fonksiyonu kontrol etmek yeterli olur. Bu sayede hata ayıklama süreci hızlanır ve daha az çaba ile sorunlar çözülebilir.

Tek Noktadan Değişiklik Yapılabilir: Eğer bir işlem programın birçok yerinde yapılıyorsa ve bu işlemde bir değişiklik yapılması gerekiyorsa, fonksiyon kullanılmadığı takdirde tüm yerlerin tek tek güncellenmesi gerekir. Ancak bu işlem bir fonksiyon içinde tanımlanmışsa, yalnızca fonksiyonun içeriğini değiştirmek yeterlidir. Böylece kodun bakım süreci kolaylaşır ve hata yapma riski azalır.

Kod Tekrarını Azaltır, Daha Az Satır Kod ile Aynı İş Yapılır: Fonksiyonlar sayesinde aynı işlemi tekrar tekrar yazmak gerekmez. Örneğin, programın 10 farklı yerinde aynı 5 satırlık işlemi yapmanız gerekiyorsa, bu işlemi bir fonksiyon haline getirerek yalnızca bir kez yazarsınız. Fonksiyon çağrıları ile bu işlemi istediğiniz yerde tekrar edebilirsiniz.

Bu durumda kodun toplam satır sayısı da önemli ölçüde azalır. Fonksiyon kullanılmazsa:

10 farklı yerde x 5 satır = 50 satır kod gerekir.

Fonksiyon kullanıldığında ise:

Fonksiyonun kendisi için 5 satır,

10 yerde fonksiyon çağrısı için 10 satır olmak üzere,

Toplam 15 satır kod yeterli olur.

Bu örnekten de görüldüğü gibi, fonksiyonlar hem kod tekrarını önler hem de yazılımı daha kısa, okunabilir ve yönetilebilir kılar.

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

1) Hataları daha kolay tespit edebiliriz. Fonksiyonlar belirli görevleri yerine getiren bağımsız yapılar olduğundan, hata oluştuğunda yalnızca ilgili fonksiyonu inceleyerek sorunu izole edebilir ve çözebiliriz. (Materyaldeki dördüncü örnek.)
2) Daha doğru ve sağlam çözümler üretebiliriz. Kodun mantıksal bloklara ayrılması ve her fonksiyonun belirli bir işlevi yerine getirmesi, yazılımın genel yapısını güçlendirir ve hatasız çalışmasına katkı sağlar. (Materyaldeki beşinci örnek.)
3) İhtiyaç duyduğumuzda belli görevleri yerine getirmesi için çağırabiliriz. Fonksiyonlar, belirli bir işlemi gerçekleştirmek üzere tanımlanır ve ihtiyaç duyulan her noktada kolayca çağrılabilir. (Materyaldeki birinci örnek.)

4) Tanımladığımız görevlerin tekrar kullanılmasını sağlayabiliriz. Fonksiyonlar bir kez yazılır ve farklı yerlerde, farklı verilerle tekrar tekrar kullanılabilir. Bu, hem verimliliği artırır hem de kod tekrarını önler. (Materyaldeki üçüncü örnek.)
5) Daha az satır kod yazar, programın yönetimini kolaylaştırırız. Aynı işlemler için yeniden kod yazmak yerine, fonksiyonları kullanarak kısa, anlaşılır ve yönetilebilir bir yapı elde ederiz. (Materyaldeki ikinci ve beşinci örnek)
6) Kod tekrarını önler, istenilen yerlerde kolayca kullanabiliriz. Fonksiyonlar sayesinde aynı kod bloğunu farklı yerlerde yeniden yazmak yerine sadece çağırmak yeterlidir. Bu da yazılımın sürdürülebilirliğini artırır. (Materyaldeki ikinci ve beşinci örnek)

B2. Fonksiyonların Nasıl Kullanıldığını Keşfediyorum

Süre: 50 dk.

Kazanımlar: K2. C++ programlama dilinde fonksiyon oluşturmayı bilir.

Materyaller: O5.B2.1 C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri

O5.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 "O5.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 (O5.B2.2) öğrencilerle paylaşılır. Öğrencilerin ihtiyaç duyduğu aşamada eğitmenler konu ile ilgili yöntemsel bilgiyi öğrenciye verir.

Eğitmene Öneriler: Yukarıdaki etkinlikleri öğrenciler bitirdikten sonra fonksiyonlar aşağıdaki gibi özetlenir.

C++ programlama dilinde fonksiyonlar, belirli bir görevi yerine getiren ve gerektiğinde bir değer döndürebilen bağımsız kod bloklarıdır. Fonksiyonlar sayesinde programın yapısı daha modüler, okunabilir ve sürdürülebilir hale gelir. C++ dilinde bir fonksiyon tanımlarken üç temel bileşen bulunur:

1. Dönüş Tipi (Return Type): Fonksiyonun işlemi tamamladıktan sonra geriye hangi türde bir değer döndüreceğini belirtir. Bu değer, int, float, double, bool, string gibi veri türlerinden biri olabilir. Eğer fonksiyon işlem sonunda herhangi bir değer döndürmeyecekse void anahtar kelimesi kullanılır.

Örneğin:

int → tamsayı döndürür

bool → doğru ya da yanlış (true/false) döndürür

void → hiçbir değer döndürmez

2. Fonksiyon İsmi: Fonksiyonun ne işe yaradığını belirten tanımlayıcı bir isimdir. Fonksiyon isimleri anlamlı ve fonksiyonun görevini yansıtan şekilde seçilmelidir. Bu, hem kodun okunabilirliğini artırır hem de bir başkasının fonksiyonu anlamasını kolaylaştırır.

İsimlendirme yaparken genellikle küçük harf ve alt çizgi (_) kullanılır (ortalama_al, fiyat_hesapla) ya da PascalCase ya da camelCase gibi farklı stil yaklaşımları tercih edilebilir (OrtalamaAl, fiyatHesapla).

3. Parametreler: Fonksiyonun çalışabilmesi için ihtiyaç duyduğu dış veriler parametreler aracılığıyla fonksiyona aktarılır. Parametreler, fonksiyon isminin yanında parantez içinde belirtilir ve türleriyle birlikte tanımlanır. Fonksiyon birden fazla parametre alabilir ve parametreler virgül ile ayrılır.

Örneğin:

```
void mesaj_goster(string mesaj, int tekrar_sayisi);
```

Fonksiyonun genel yazım şekli şu şekildedir:

```
dönüş_tipi fonksiyon_ismi(parametreler)
{
   // Fonksiyon içinde yapılacak işlemler
   // Gerekirse return ifadesiyle değer döndürülür
}
```

Örneğin, bir sosyal medya uygulamasında, örneğin Instagram'da, her fotoğrafın benzersiz bir numarası (ID) vardır. Bu numaralar sayesinde belirli bir fotoğraf üzerinde işlem yapılabilir. Bir fotoğrafa yorum yapmak istiyorsak, bu işlemi bir fonksiyonla gerçekleştirebiliriz. Bu durumda fonksiyonun yapısı şöyle olabilir:

Fonksiyon ismi: yorum_yap

Parametreler: Fotoğrafın ID numarası ve yorum metni

Dönüş tipi: bool — Yorumun başarıyla gönderilip gönderilmediğini belirtmek için

Örnek kod:

```
bool yorum_yap(int foto_id, string yorum)
{
   // Yorum gönderme işlemleri
```

```
    // İşlem başarılıysa true, başarısızsa false döndürülür
    return true;
}
```

Bu örnek, gerçek dünyadaki problemleri programlama yoluyla nasıl çözdüğümüzü açıkça gösterir.

Fonksiyon isimleri, ne işe yaradıklarını açıkça belli edecek şekilde tanımlanmalıdır. Fonksiyonu kullanan başka bir geliştirici, sadece ismine bakarak fonksiyonun ne yaptığını anlayabilmelidir. Bu yazılımın okunabilirliğini ve sürdürülebilirliğini artırır. Verilen sayıların ortalamasını alan bir fonksiyon için:

ortalama_al() veya OrtalamaAl()

Kullanıcılara e-posta gönderen bir fonksiyon için:

mail_at() veya MailAt()

C++ dilinde fonksiyon yazmak, sadece teknik bir gereklilik değil, iyi bir yazılım tasarımı için temel bir yaklaşımdır. Fonksiyonlar kodu parçalayarak:

- Kod tekrarını azaltır
- Bakımı ve geliştirmesi kolay hale getirir
- Hata ayıklamayı kolaylaştırır
- Okunabilirliği artırır

B3. Fonksiyonları Kullanarak Kodlama Yapalım

Süre: 50 dk.

Kazanımlar: K1. C++ programlama dilinde fonksiyon tanımlayabilir.

K2. C++ programlama dilinde fonksiyon oluşturmayı bilir.

K3. C++ programlama dilinde fonksiyon çağırmayı bilir.

Materyaller: O5.B3.1. Fonksiyonları Kullanarak Kodlama Yapma

O5.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 etkinliğindeki uygulamaları, öğrencilerin kodlamasını ister. 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 sayı 5 ile tam bölünüyor ise ekrana "tam bölünür." aksi durumda kalanı yazdıran fonksiyonu yazalım.

Cevap 3:

```
#include <iostream>
using namespace std;
void bes_ile_bolme(int sayi)
{
    if(sayi % 5 == 0)
        cout << "tam bolunur."<<endl;
    else
        cout << sayi%5<<endl;
}
int main()
{
    bes_ile_bolme(11);
    bes_ile_bolme(15);
    bes_ile_bolme(12);
}
```

Görev 4: Fonksiyona gönderilen iki sayıdan küçük olanı geriye döndüren fonksiyonu yazalım.

Cevap 4:

```
#include <iostream>
using namespace std;
int hangisi_buyuk(int sayi1, int sayi2)
{
```

```
    if(sayi1>sayi2)
        return sayi1;
    else
        return sayi2;
}
int main()
{
    int sayi = hangisi_buyuk(10,20);
    cout << sayi <<endl;

    sayi = hangisi_buyuk(22,11);
    cout << sayi <<endl;
}
```

B4. Gelişmiş Fonksiyon Örnekleri Kodluyorum

Süre: 50 dk.

Kazanımlar: K2. C++ programlama dilinde fonksiyon oluşturmayı bilir.

K3. C++ programlama dilinde fonksiyon çağırmayı bilir.

Materyaller: O5.B4.1. Gelişmiş Fonksiyonlar Yazıyorum

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 "Gelişmiş Fonksiyonlar Yazıyorum" etkinliğindeki görevlerden üç tanesinin kodlamasını sağlamalıdır. Öğrenciler kodlama esnasında verilen görevler için hazırlanan sunu destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmen sınıfı dolaşarak ve kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Fonksiyona gönderilen tam sayı tipindeki dizinin en büyük sayısını ekrana yazan fonksiyonu yazalım.

Cevap 1:

```
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 2: İki dizi içerisindeki en büyük iki sayının toplamını bulan programı fonksiyon kullanarak yazalım.

Cevap 2:

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 3: Arda, kendisine verilen 2x2 boyutundaki bir matrisin tüm elemanlarının toplamını bulan bir fonksiyon yazmak istemektedir. Bu işlemi gerçekleştirmek için diziyi fonksiyona göndererek toplamı hesaplayan programı nasıl yazmalıdır?

Cevap 3:

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

int matris_toplami(int matris[2][2]) {
    int toplam = 0;

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            toplam += matris[i][j];
        }
    }

    return toplam;
}

int main() {
    int matris[2][2] = {
        {1, 2},
        {3, 4}
    };

    int toplam = matris_toplami(matris);

    cout << "Matrisin elemanlari toplami: " << toplam << endl;

    return 0;
}
```

Hafta 5. Yüz Yüze: Ders Materyalleri

O5.B1.1. Fonksiyonların Özelliklerini Keşfediyorum

1

Evimize katı meyve sıkacağı alıyoruz ve biz bunu canımız her ne zaman meyve suyu çektiğinde meyve sıkacağını kullanıyoruz. Yani bu makinenin görevi biz istediğimizde meyve suyu yapmak.

2

İnstagram da milyonlarca fotoğraf paylaşan insan var ve bu fotoğraflar on binlerce insan tarafından beğeniliyor. İnstagramı yazanlar her beğenilen fotoğraf için ayrı ayrı kodlar mı yazdılar acaba.

3

Her gün sabah uyanıp ekmek almaya gitmekten yoruldum. Keşke bir yardımcı robotum olsaydı onu her çağırdığımda gelip benim yerime ekmek alsaydı.

4

Türkçe öğretmenimin bana verdiği kompozisyon ödevini 20 sayfa yazarak bitirdim. Öğretmenim "yalnız" kelimesini yanlış yazdığımı söyledi. Şimdi tüm sayfalara teker teker gidip bu yanlışları düzeltmem gerekecek. Keşke yanlış yazdığım kelimenin birisini düzelttiğimde diğer yanlışlarımda düzelseydi.

5

Kodlamak istediğim web sitesinde sisteme her giriş yapan kullanıcıya Merhaba "kullanıcının ismi yazmak istiyorum. Ne yani binlerce kişi web siteme girerse her isim için ayrı ayrı merhaba mı diyeceğim.

O5.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 oluşturacak olan 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?

O5.B2.2. Fonksiyonların Kullanım Afişi

Resim 28. *Fonksiyon tanımlama afişi*

O5.B3.1. Fonksiyonları Kullanarak Kodlama Yapma

1

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

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?

3

Fonksiyona gönderilen sayı 5 ile tam bölünüyor ise ekrana "tam bölünür." aksi durumda kalanı yazdıran fonksiyonu yazalım.

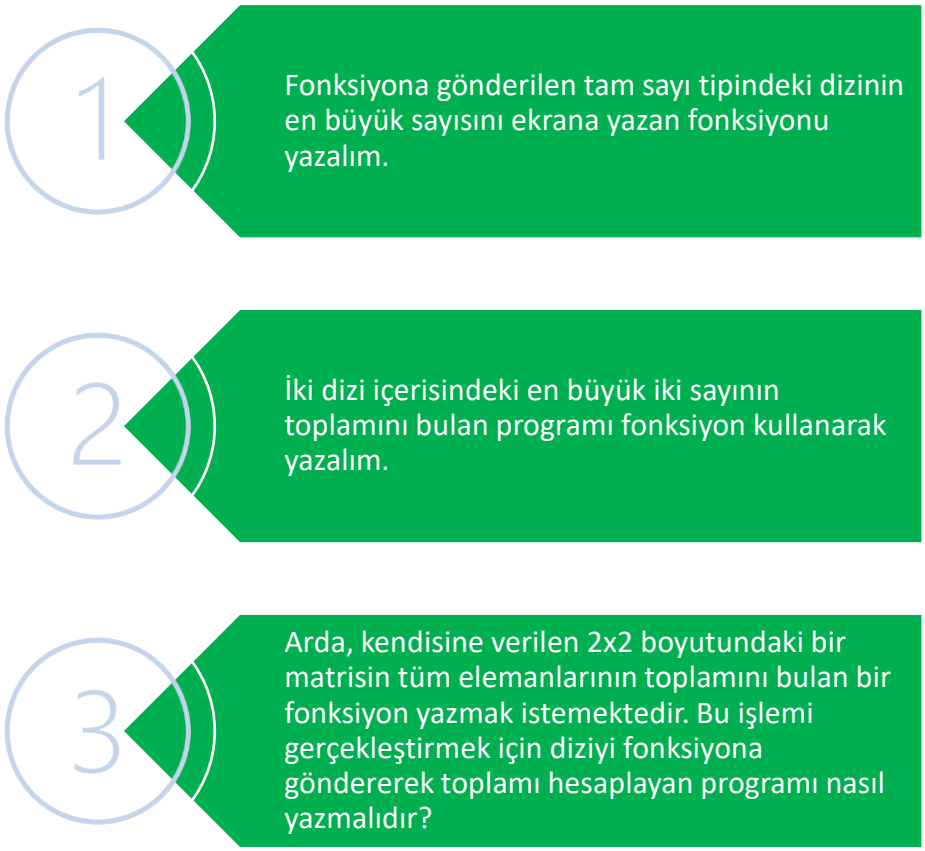
4

Fonksiyona gönderilen iki sayıdan küçük olanı geriye döndüren fonksiyonu yazalım.

O5.B3.2. Fonksiyon Kodlama ile İlgili Destekleyici Bilgi Sunusu

Yukarıda verilen görevlerin her biri için destekleyici bilgi hazırlanmıştır. Bu destekleyici bilgilere ulaşmak için O5.B3.2 adlı sunumu açınız. Bu sunu öğrencilerin bilgisayarlara gönderilerek veya her öğrenci grubu için çıktı alınarak kullanılabilir.

O5.B4.1. Gelişmiş Fonksiyonlar Yazıyorum



Hafta 5. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftaki çevrimiçi ders içeriğinin amacı, C++'ta fonksiyon kavramını öğrenerek, farklı senaryolara uygun şekilde parametre alan, değer döndüren veya void türünde fonksiyonlar tasarlamaktır. Gerçek hayattan örneklerle fonksiyonların kullanımını kavramak ve Code::Blocks ortamında uygulamalar geliştirerek programlarda tekrar eden işlemleri daha düzenli, anlaşılır ve yeniden kullanılabilir biçimde yazmayı pekiştirmektir.

Etkinlik Uygulama Ortamı:

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrimiçi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Günün Rotası (5 dk.)

B. Öğrenme Görevleri

B1. Fonksiyon Kullanarak Yazılmış Bir Kodun, Programın Çıktısını Tahmin Etme (20 dk.)

B2. Parametre Olarak Alınan İki Sayının Toplamını Döndüren Fonksiyonun Yazılması (25 dk.)

Ders Arası (10 dk.)

B3. Parametrelerle Karakter Tekrar Fonksiyonu Oluşturma (25 dk.)

B4. Sosyal Medya Fotoğraflarının Beğeni Toplamını Hesaplayan Fonksiyon (25 dk.)

A. Giriş: Günün Rotası

Süre: 5 dk.

Uygulama: Eğitmen, çevrimiçi ders saati başladığında, öğrencilerin sanal sınıfa katılmasını beklerken arka planda odaklanmayı kolaylaştıracak hafif bir enstrümantal müzik açar. Yeterli katılım sağlandığında müziği yavaşça kısar ve öğrencileri sıcak bir şekilde selamlar. Ardından bugünkü dersin yol haritasını tek bir paragrafta net biçimde özetler: Arkadaşlar, bugünkü temel amacımız, C++'ta fonksiyonların nasıl tanımlandığını ve program içinde nasıl kullanılabileceğini öğrenmektir. Bu amaç doğrultusunda, parametre alan ve almayan fonksiyonların, değer döndüren ve döndürmeyen fonksiyonların nasıl yazıldığını inceleyeceğiz. Gerçek hayattan örneklerle, bir programda tekrar eden işlemleri fonksiyonlar sayesinde nasıl daha düzenli ve anlaşılır hale getirebileceğimizi göreceğiz. Gün sonunda, fonksiyonların mantığını kavrayarak Code::Blocks üzerinde kendi fonksiyonlarımızı tanımlayabilecek ve bu fonksiyonları çağırarak küçük ama işlevsel programlar yazabiliyor olacağız.

B. Öğrenme Görevleri

B1. Fonksiyon Kullanarak Yazılmış Bir Programın Çıktısını Tahmin Etme

Süre: 20 dk.

Materyaller: OÇ5.B1.1. Öğrenci Yönerge Metni

OÇ5.B1.2. Yanıt

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ5.B1.1. Öğrenci Yönerge Metni'nde yer alan kodun çıktısını tahmin etmesi istenecektir. Bu etkinliğine yönelik eğitmenin yapacağı zaman planı aşağıda gösterildiği gibi planlanacaktır.

Zaman Planı:

- **0-3 dakika:** Öğrenciler kodu dikkatle inceler, fonksiyon tanımlarını, parametreleri, geri dönüş değerlerini ve fonksiyonun program içindeki kullanımını analiz eder."
- **3-5 dakika:** Her öğrenci, kodu çalıştırmadan önce ekranda ne çıkacağını bireysel olarak tahmin etmesi ve tahminlerini chat kısmından yazması istenir.
- **5-15 dakika:** Öğrenciler kodu Code::Blocks üzerinde çalıştırarak tahminleri ile gerçek sonucu karşılaştırır.
- **15-20 dakika:** Eğitmen, doğru tahmin yöntemlerini ve sık yapılan hataları sınıfla paylaşır, program akışını adım adım açıklayarak geri bildirim verir.

B2. Parametre Olarak Alınan İki Sayının Toplamını Döndüren Fonksiyonun Yazılması

Süre: 25 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ5.B2.1. Öğrenci Yönerge Metni

OÇ5.B2.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen OÇ5.B2.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Parametre olarak gönderilen iki sayının toplamını geriye döndüren bir fonksiyonu oluşturmak isteseydiniz nasıl bir kod yazardınız?

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi değişkenlerin kullanılacağını, nasıl bir fonksiyon yazılması gerektiğine yönelik analizlerin yapılması.
- **5-20 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **20-25 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B3. Parametrelerle Karakter Tekrar Fonksiyonu Oluşturma

Süre: 25 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ5.B3.1. Öğrenci Yönerge Metni

OÇ5.B3.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen OÇ5.B3.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Parametre olarak gönderilen harfi, yine parametre olarak gönderilen sayı kadar ekrana yazdıran bir fonksiyon tanımlamak isteseydiniz nasıl kodlardınız?

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi değişkenlerin kullanılacağını, nasıl bir fonksiyon yazılması gerektiğine yönelik analizlerin yapılması.

- **5-20 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **20-25 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B4. Sosyal Medya Fotoğraflarının Beğeni Toplamını Hesaplayan Fonksiyon

Süre: 25 dk.

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Materyaller: OÇ5.B4.1. Öğrenci Yönerge Metni

OÇ5.B4.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen OÇ5.B4.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir kullanıcının sosyal medyada paylaştığı fotoğraflardaki toplam beğeni sayılarının bir dizide saklandığı düşünülürse; paylaşılan fotoğraflardaki beğenilerin toplamını geriye döndüren bir fonksiyonu nasıl kodlardınız?

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi değişkenlerin kullanılacağını, nasıl bir fonksiyon yazılması gerektiğine yönelik analizlerin yapılması.
- **5-20 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **20-25 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

Hafta 5. Çevrim İçi: Ders Materyalleri

OÇ5.B1.1. Öğrenci Yönerge Metni

Göreviniz: Aşağıdaki programın ekran çıktısı nedir? Kodu bilgisayarınıza yazarak cevabınızı kontrol ediniz.

```
#include <iostream>
using namespace std;
int sayi=2;
void fonksiyon1()
{
    sayi = 5;
}
void fonksiyon2()
{
    int sayi = 7;
}
int main()
{
    fonksiyon1();
    fonksiyon2();
    cout << sayi;
}
```

Arda'nın tahmini doğru olduğuna göre sizce nasıl bir tahminde bulunmuştur?

OÇ5.B1.2. Yanıt

Cevap: 5

OÇ5.B2.1. Öğrenci Yönerge Metni

Göreviniz: Parametre olarak gönderilen iki sayının toplamını geriye döndüren bir fonksiyonu oluşturmak isteseydiniz nasıl bir kod yazardınız?

OÇ5.B2.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int topla(int sayi1, int sayi2)
{
    int toplam = sayi1 + sayi2;
    return toplam;
}
int main()
{
    int sonuc = topla(5,4);
    cout<< sonuc;
}
```

OÇ5.B3.1. Öğrenci Yönerge Metni

Göreviniz: Parametre olarak gönderilen harfi, yine parametre olarak gönderilen sayı kadar ekrana yazdıran bir fonksiyon tanımlamak isteseydiniz nasıl kodlardınız?

OÇ5.B3.2. Kod Bloğu

```
#include <iostream>
using namespace std;
void tekrar_yaz(char a, int adet)
{
    for (int i=0;i<adet;i++)
        cout << a;
}
int main()
{
    tekrar_yaz('z',5);
}
```

OÇ5.B4.1. Öğrenci Yönerge Metni

Göreviniz: Bir kullanıcının sosyal medyada paylaştığı fotoğraflardaki toplam beğeni sayılarının bir dizide saklandığı düşünülürse; paylaşılan fotoğraflardaki beğenilerin toplamını geriye döndüren bir fonksiyonu nasıl kodlardınız?

OÇ5.B4.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int dizi_topla(int dizi[5])
{
    int toplam = 0;
    for (int i=0;i<5;i++)
        toplam = toplam + dizi[i];
    return toplam;
}
int main()
{
    int sayilar[5] = {5,6,9,3,2};
    int sonuc = dizi_topla(sayilar);
    cout << sonuc;
}
```

Hafta 6. Yüz Yüze: Nesne Yönelimli Programlama

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.
- K8. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.
- K9. Yapıcı ve yıkıcı fonksiyonları kullanarak program geliştirir.

Amaç

Bu haftanın amacı, nesne yönelimli programlamanın temel felsefesini, yordamsal programlamadan farkını, avantaj ve dezavantajları ile nesne yönelimli programlamada sınıf, nesne ve fonksiyon tanımlama işlemlerini uygulamaktır. Ayrıca yapıcı, yıkıcı ve sabit fonksiyonları kullanmak, sınıf dosyaları ile büyük boyutlu projeler hazırlamayı anlamaktır.

Önerilen Ders Akışı

A. Öğrenme Görevleri

A1. Nesneni Çiz (25 dk.)

A2. Sınıfının Özelliklerini Tanı (25 dk.)

Ders Arası (10 dk.)

A2. Sınıfının Özelliklerini Tanı (25 dk.)

A3. Kendi Sınıfını Afişle (25 dk.)

Ders Arası (10 dk.)

A3. Kendi Sınıfını Afişle (25 dk.)

A4. Yarış Benimle (25 dk.)

Ders Arası (10 dk.)

A5. Kod Satırlarını Tamamla (50 dk.)

A. Öğrenme Görevleri

A1. Nesneni Çiz

Sure: 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.

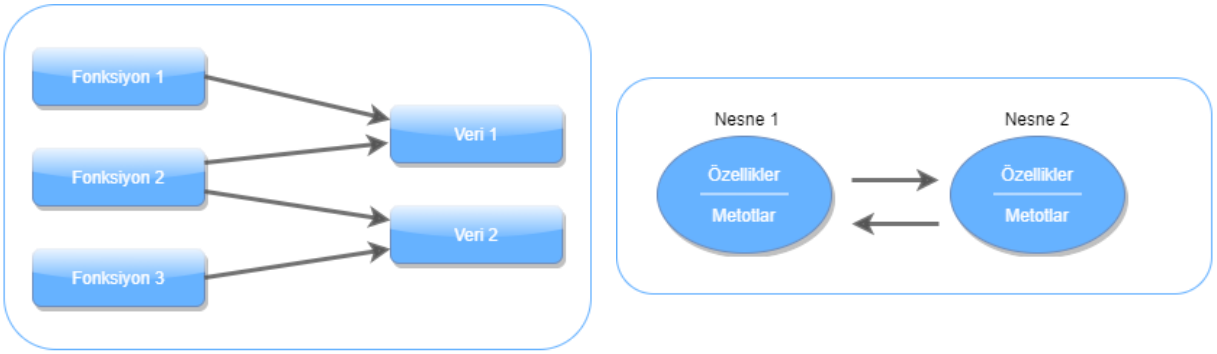
Materyaller: Bir fanus ya da kura torbası

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: C++ programlama dili, temelde güçlü ve düşük seviyeli bir dil olan C programlama diline nesne yönelimli programlama özelliklerini eklemek amacıyla geliştirilmiştir. C dili, sistem programlama ve donanım seviyesinde oldukça etkili olmasına rağmen, nesne yönelimli yapıları doğal olarak desteklemez. C++, bu eksikliği gidermek ve daha modüler, yeniden kullanılabilir ve bakımı kolay yazılımlar geliştirmeyi sağlamak için tasarlanmıştır.

Nesne yönelimli programlama, yazılımları, birbiriyle etkileşim kuran nesnelerin oluşturduğu sistemler olarak tasarlayan bir yaklaşımdır. Bu yöntem, bir programın işleyişini yalnızca işlemler (fonksiyonlar) üzerinden değil, aynı zamanda veri ve bu veriler üzerinde çalışan fonksiyonları bir arada tutan "nesneler" üzerinden tanımlar.



Resim 29. *Yordamsal ve nesne yönelimli programlama*

Yordamsal programlama, veriler üzerinde işlemler gerçekleştiren fonksiyonlar yazmakla ilgiliyken, nesne yönelimli programlama ise hem verileri (özellikler) hem de fonksiyonları içeren nesneler oluşturmakla ilgilidir. Yordamsal programlama geleneksel programlama yöntemlerinden biridir ve işlemleri sıralı adımlar hâlinde ele alır. Bu yaklaşımda:

- Veriler ve fonksiyonlar ayrı ayrı tanımlanır.
- Veriler üzerinde işlem yapılacaksa, bu veriye hangi fonksiyonların etki ettiğini takip etmek zor olabilir.
- Program büyüdükçe, kodun yönetimi, bakımı ve genişletilmesi zorlaşabilir.

Nesne yönelimli programlama ise bu dezavantajları ortadan kaldırmak amacıyla geliştirilmiş bir yaklaşımdır. Nesne yönelimli programlamada:

- Veriler (özellikler) ve bu verilere ait işlemler (fonksiyonlar / metotlar) birlikte tanımlanır ve nesne adı verilen yapılarda birleştirilir.
- Bu yapı sayesinde, her nesne kendi görevini bilir ve diğer nesnelerle düzenli, kontrollü bir şekilde iletişim kurar.
- Kod daha okunabilir, sürdürülebilir ve genişletilebilir hale gelir.

Nesne yönelimli programlamanın, yordamsal programlamaya göre birçok avantajı vardır:

- Daha hızlıdır 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.

Prosedürel Programlama	Nesne Yönelimli Programlama

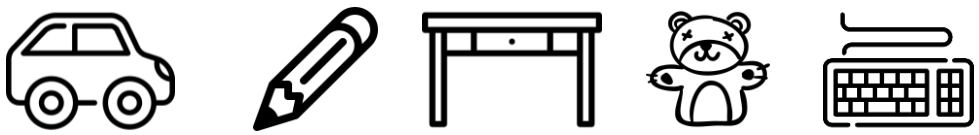
Program, fonksiyon adı verilen küçük parçalara bölünmüştür.	Program, nesneler adı verilen küçük parçalara ayrılır.
Yeni veri ve fonksiyon eklemek kolay değildir.	Yeni veri ve fonksiyon eklemek koladır.
Verileri gizlemek için uygun bir yol yoktur, bu nedenle daha az güvenlidir.	Daha güvenli olmak için 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.

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 (Object) Kavramı

Nesne yönelimli programlama yaklaşımında, temel yapı taşı nesne (object) kavramıdır. Bir nesne, belirli özelliklere (veri/öz nitelikler) ve davranışlara (fonksiyonlar/metotlar) sahip, yazılım içinde tanımlanabilir bir varlıktır. Gerçek dünya varlıkları gibi düşünülebilir: Her nesne kendine özgü bir durumu ve gerçekleştirdiği eylemleri barındırır.

Nesneler, hem verileri (özellikler) hem de bu verilere ait işlemleri (davranışlar) bir arada tutan yapılardır. Bu özellik, nesne yönelimli programlamanın prosedürel programlamaya göre en büyük farkıdır: Veriler ve işlemler artık ayrı ayrı değil, bir bütün olarak tek bir birim (nesne) içerisinde tanımlanır. Örneğin: araba, kalem, masa, oyuncak ve klavye gibi.



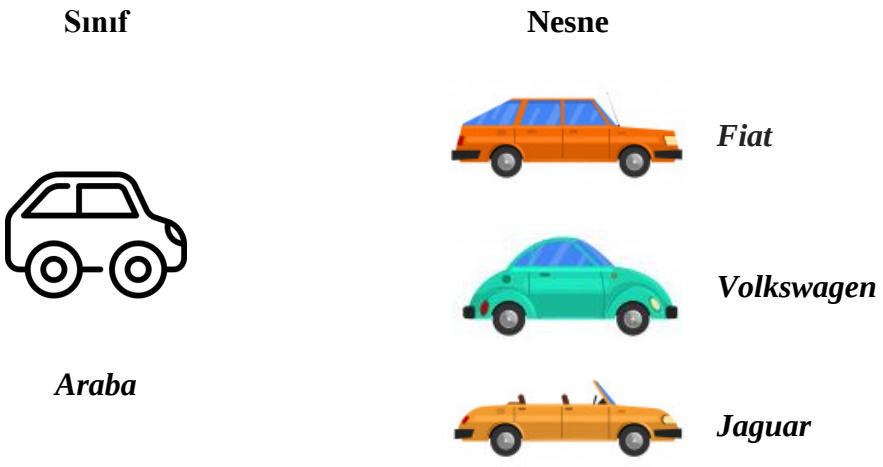
Resim 30. *Nesne örnekleri*

Nesne soyut bir veri örneğidir. Sınıf tanımlandığında, bellek ayrılmaz, ancak örnek oluşturulduğunda (yani bir nesne oluşturulduğunda) bellek ayrılır.

Sınıf (Class) Kavramı

Nesne yönelimli programlamada, bir sınıf (class), bir nesnenin sahip olabileceği tüm özellikleri (verileri) ve gerçekleştirebileceği işlemleri (metotları) tanımlayan bir şablon ya da taslaktır. Yani sınıf, bir nesnenin ne olduğunu, neleri içereceğini ve onunla ne tür işlemler yapılabileceğini belirleyen bir plandır. Bir sınıf tanımladığınızda henüz ortada somut bir nesne

yoktur. Ama bu sınıf, o sınıftan üretilecek nesnelerin nasıl görüneceğini ve nasıl davranacağını belirler. Araba bir sınıf ise bundan üretilen farklı araba çeşitleri de nesneleri ifade eder.



Resim 31. *Sınıf ve nesne kavramı*

Sınıf, veri üyeleri ve üye fonksiyonlarına sahip kullanıcı tanımlı bir veri türüdür. Veri üyeleri, veri değişkenleridir. Üye fonksiyonları ise bu değişkenleri işlemek için kullanılan fonksiyonlardır. Bunlar birlikte bir sınıftaki nesnelerin özelliklerini ve davranışını tanımlar.

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. Sınıfın Özelliklerini Tanı

Süre: 25 dk. + 25 dk.

Kazanımlar: 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.

Materyaller: O6.A2.1. Grup Afişleri

Hazırlık: Materyaller her gruba 2'şer adet dağıtılmak üzere çıktı alınır.

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 oluşturma* ve *nesne oluşturma* terimlerini tartışmaları istenir. Gruplara 5 dk. inceleme süresinin ardından, 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. Tüm grupların dağılma sürecinin ardından eğitmen öğrencilerden bu kavramlar hakkında ne öğrendiklerini özetleyen bir grup kâğıdı oluşturmalarını ve bunları tahtaya asmalarını ister. Eğitmen bu kâğıtlar üzerinden her bir kavramı açıklamaya çalışır.

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.

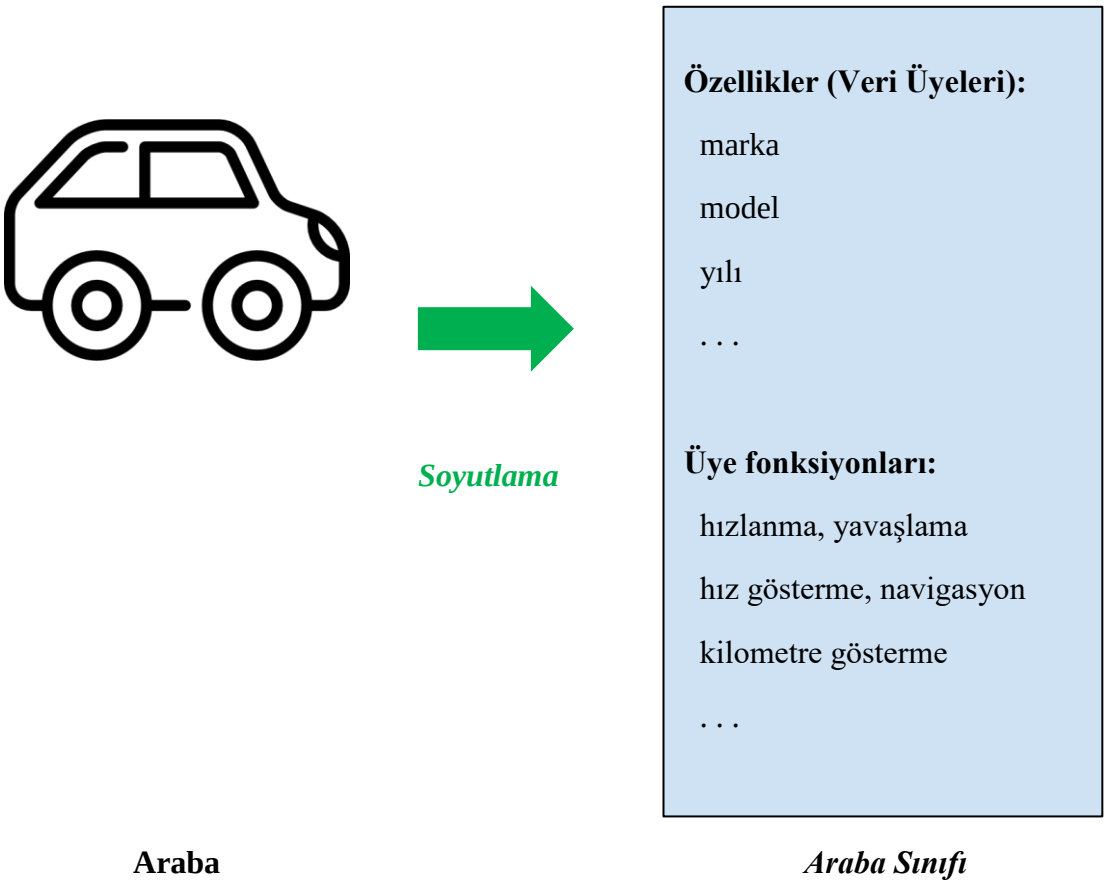
Konu İçeriği: Sınıf Tanımlama

Sınıflar, nesne yönelimli programlamanın en önemli yapı elemanıdır. Bir sınıf, bir nesnenin özelliklerini ve kapasitelerini tanımlar. Pek çok farklı sınıf tanımı oluşturabilirsiniz. Bu bölümden itibaren artık nesne yönelimli programlama tasarımına giriş yapıyoruz. Başlangıçta biraz zorlu olacağı öngörülebilir fakat ne kadar değerli bir şey öğrendiğimizi unutmayalım. Kavramları anlamaya başladığınızda ve kod yazmayı ilerledikçe, nesne yönelimli

programlamanın ilk bakışta göründüğü kadar zor olmadığını fark edeceksiniz. İlk aşamada sınıf kavramını anlamak için bir gerçek dünya nesnesi olan arabayı düşünebiliriz.

UYARI:

Her sınıf modeline yalnızca bir kavram (nesne bütünlüğü) sunmalıdır. Örneğin, Araba sınıfını Sürücü sınıfından ayrı tutun.



Resim 32. *Araba sınıfı örneği*

Yukarıda gördüğünüz üzere arabaya ait bir sınıf oluşturmak istediğimiz zaman bu sınıfa ait özellikleri ve fonksiyonları belirlememiz gerekmektedir. Özellikler sınıfın ayırt edici bilgilerinden oluşan marka, model ve yılı gibi verileri içermektedir. Fonksiyonlar ise sınıfın hızlanma, yavaşlama ve hız ve kilometre gösterme 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
{
public:
```

```
        üyeListesi
};
```

Burada **class**, sınıf tanımına ait anahtar kelime, **public** üye listesinin erişim belirteci, **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:

Bu eğitim kapsamında sınıfları ve sınıf üyelerini ayırt etmek için standart adlandırma kurallarına uyacağız. Sınıf adları büyük harfle ve üye adları küçük harfle başlatılacaktır. Bu kural sadece C++ 'da değil, aynı zamanda diğer tüm nesne yönelimli programlama dillerinde de kullanılır.

üyeListesi, üye bildirimlerini içeren listeden oluşur. Bunlar üye veri veya üye 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.

```
char a;
int b;
double c;
```

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 araba sınıfında markanın, modelin ve yılın ne olduğu bu veri üyeleri tarafından saklanır. Bir sınıfın veri üyeleri, sınıf özelliklerini tanımlar ve açıklar. Bu nedenle, her bir araba nesnesinin markası, o nesnenin bir özelliğidir. Aşağıda araba sınıfına ait marka, model, yıl, renk ve fiyat veri üyelerinin sınıf içerisinde nasıl tanımlandığını görebilirsiniz.

```
class Araba
{
public:
    char marka[30];
```

```
    char model[30];
    int yıl;
    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 sahip olabileceği farklı erişim belirteçlerinden ikisi:

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.

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;
    int imei_no;
};
```

Yukarıda verilen sınıf tanımlamasında fiyat ve imei no veri üyeleri ile birlikte marka ve yıl veri üyelerinin de "private" olduğunu unutmayınız. Diğer taraftan model ve renk "public" olarak tanımlanmıştır. Sınıf tanımlamalarınızda istediğiniz kadar erişim belirteciniz olabilir, ancak belirteçleri tek bir grup altında toplamak sınıfın anlaşılırlığını artıracaktır.

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 tanımlanan bu fonksiyonlara yalnızca sınıfın bir nesnesi aracılığıyla erişilebilir.

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
}
```

Tanımlanan fonksiyon içinde, sınıfa ait tüm üyelere adları kullanılarak erişilebilir. Sınıf üyeliği otomatik olarak sağlanır ve aynı sınıfa ait fonksiyonlar birbirini doğrudan çağırabilir. Belirli bir nesne için bir fonksiyon çağrıldığında, ilgili fonksiyon bu nesnenin verilerini işleyebilir. Artık Araba sınıfının hızlanma ve yavaşlama fonksiyonlarını uygulayabiliriz.

```
class Araba
{
public:
    char marka[30];
    char model[30];
    float fiyat;
    int hiz;
```

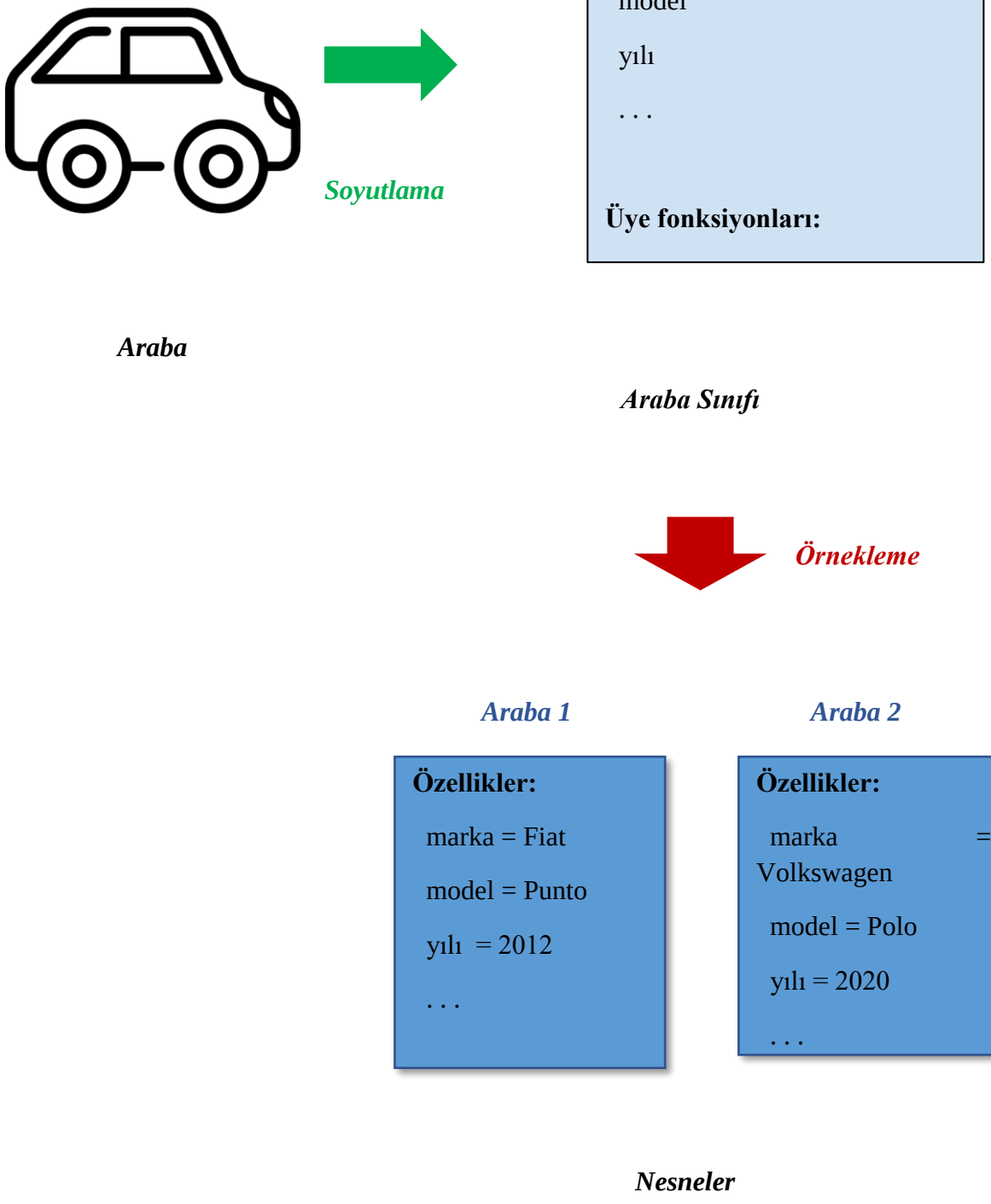
```
    void hizlanma();
    void yavaslama();
};

void Araba::hizlanma()
{
    hiz = hiz + 10;
    cout << "Araba hizlaniyor." << endl;
}
void Araba::yavaslama()
{
    hiz = hiz - 10;
    cout << "Araba yavasliyor." << endl;
}
```

UYARI:

Her yöntemin ve sınıfın başına, kullanıcıya tanımlanan üye veya sınıfın ne iş yaptığını belirten bir yorum satırı ekleyin.

Nesne Tanımlama

Nesne, sınıfın bir örneğidir. Bir sınıf tanımlandığında, bellek tahsis edilmez, ancak bir nesne oluşturulduğunda (somutlaştırıldığında) bellek ayrılır. Sınıfta tanımlanan verileri kullanmak ve fonksiyonlara erişmek için nesneler oluşturmamız gerekir. Daha önce, bir sınıf oluşturmanın bir veri türü oluşturmaya benzediğini belirtmiştik. Bu ilişki, ilerleyen aşamalarda nesneleri daha iyi öğrenmeye başladığınızda belirginleşecektir. Bir sınıfın adını temel bir veri türünün adı gibi (int, char, float) kullanabilirsiniz. Bir nesne değişkeni, belirli bir sınıfın nesnesini depolayan bir değişkendir. Aşağıdaki görselde araba sınıfı kullanılarak birden fazla nesnenin nasıl oluşturulduğunu görebilirsiniz.



Resim 33. *Araba sınıfı ve nesneleri*

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 belirtiniz. Nesne oluşturmak için aşağıda verilen sözdizimlerinden birini kullanabilirsiniz:

```
SınıfAdı nesneDeğişkeni (parametre_listesi);

SınıfAdı nesneDeğişkeni = SınıfAdı (parametre_listesi);
```

A3. Kendi Sınıfını Afişle

Süre: 25 dk. + 25 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. Öğrenciler burada A1 etkinliğinde çizdikleri nesne ve sınıflardan birine karar verebilirler. 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 öğretmen 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. Bu noktada afişin revize edilebilir dosyası öğrencilerle paylaşılabilir. Böylece afiş üzerindeki kodlar ve açıklamalar üzerinde öğrenciler değişiklik yapabilir. Diğer bir alternatif ise, Word ortamında afişi oluşturmaları istenebilir.

A4. Yarış Benimle

Süre: 25 dk.

Kazanımlar: K8. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.

Materyaller: O6.A4.1. Yarışma Kartları

O6.A4.2. Kod Örneği

Hazırlık: Bir kura torbasında yarışma kartları kesikli çizgilerden kesilerek kuraya hazırlanır. Kod örneği ÖYS'den öğrenci erişimine açılır ya da 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.

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 atamak 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 SınıfAdi
{
SınıfAdi (parametre listesi){
       yapıcı fonksiyon gövdesi
}
~SınıfAdi (){
       yıkıcı fonksiyon gövdesi
}
};
```

Yapıcı ve yıkıcı fonksiyonların tanımlamaları daha sonra yapılacaksa aşağıdaki sözdizimi kullanılır.

```
class SınıfAdi
{
       SınıfAdi (parametre listesi);
       ~SınıfAdi ();
};
```

```
SınıfAdi::SınıfAdi (parametre listesi){
        yapıcı fonksiyon gövdesi
}
SınıfAdi::~SınıfAdi (){
        yıkıcı fonksiyon gövdesi
}
```

Yapıcı fonksiyonun 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.

```
class Ceptelefonu
{
public:
    char model[30];
    float fiyat;
    bool aramaDurum;
    bool mesajDurum;

    void arama();
    void mesaj_gonder();

    Ceptelefonu(){
        aramaDurum = false;
        mesajDurum = false;
    }
    ~Ceptelefonu(){
        cout << "Nesne yok edildi." << endl;
    }
};
```

A5. Kod Satırlarını Tamamla

Süre: 50 dk.

Kazanımlar: K9. Yapıcı ve yıkıcı fonksiyonları kullanarak program geliştirir.

Materyaller: O6.A5.1. Grup Görev Kartları

Hazırlık: Materyallerin birer çıktısı alınır ve gruplara dağıtmak için kesilir.

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";

}
```

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 hizlan(int x);

};

int Araba::hizlan(int x) {

  return hiz + x;

}

int main() {

    Araba araba1; // Araba nesnesini olusturun

    strcpy(araba1.marka, "Opel");

    strcpy(araba1.model, "Astra");
```

```
    araba1.hiz = 120;
    cout << "Araba: " << araba1.marka << " " << araba1.model << " \n";
    cout << "Yeni Hiz: " << araba1.hizlan(30); // Fonksiyonu parametre ile cagirin
}
```

Kodun Çıktısı:

```
Araba: Opel Astra
Yeni Hiz: 150
```

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 fonksiyonu 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";
```

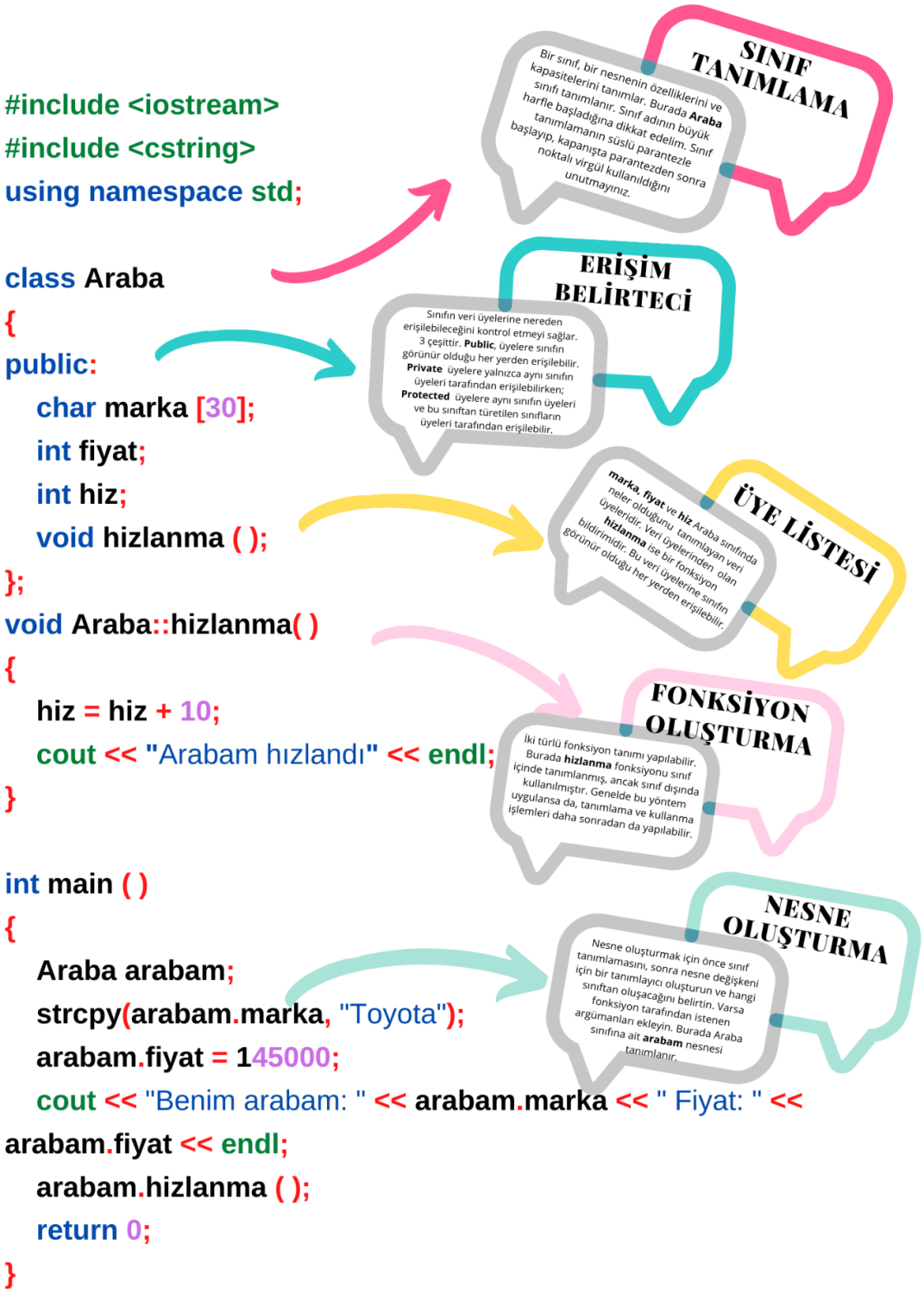
```
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil << " " << araba2.fiyat << "\n";

}
```

Kodun Çıktısı:

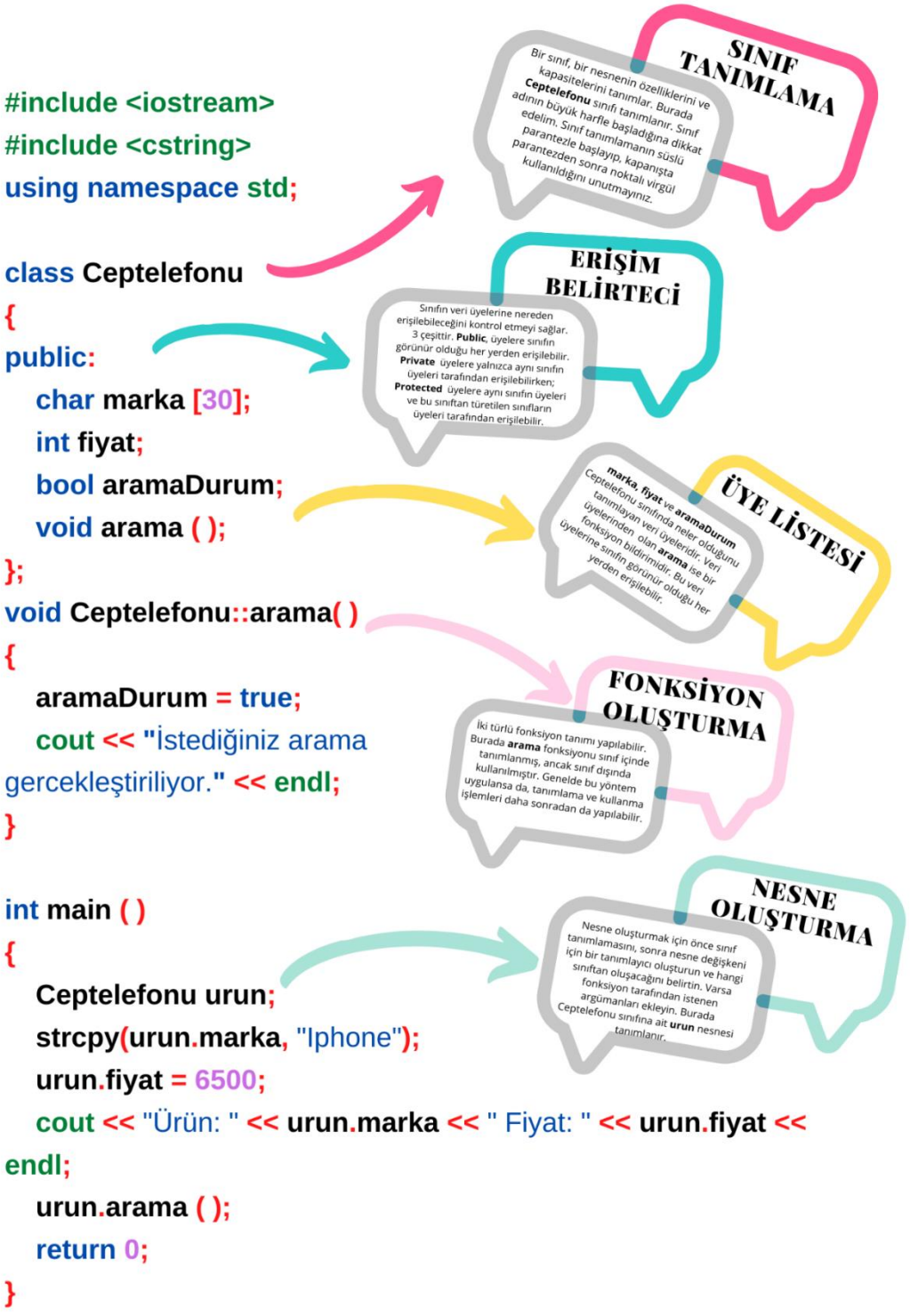
```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

Hafta 6. Yüz Yüze: Ders Materyalleri

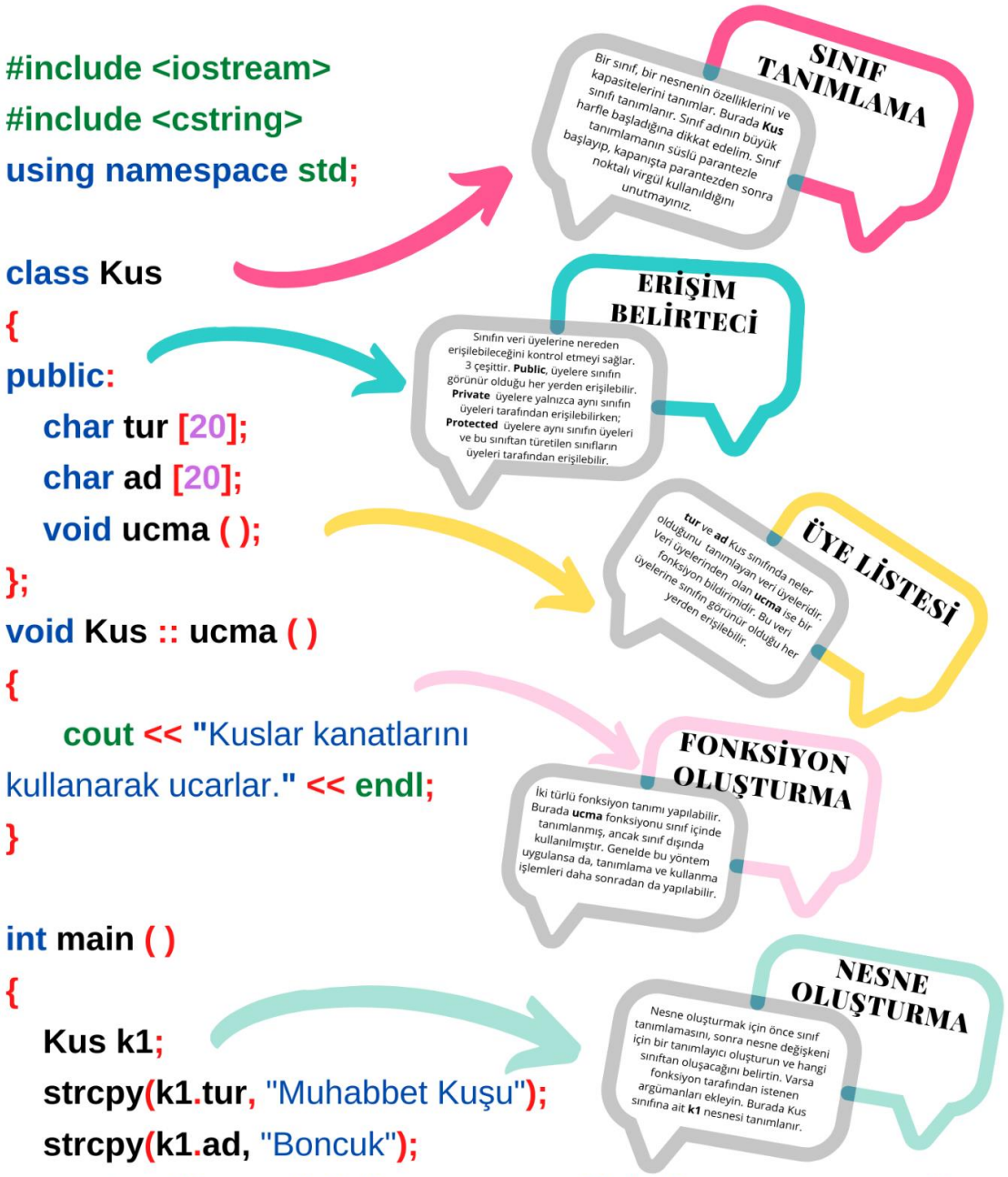
O6.A2.1. Grup Afişleri



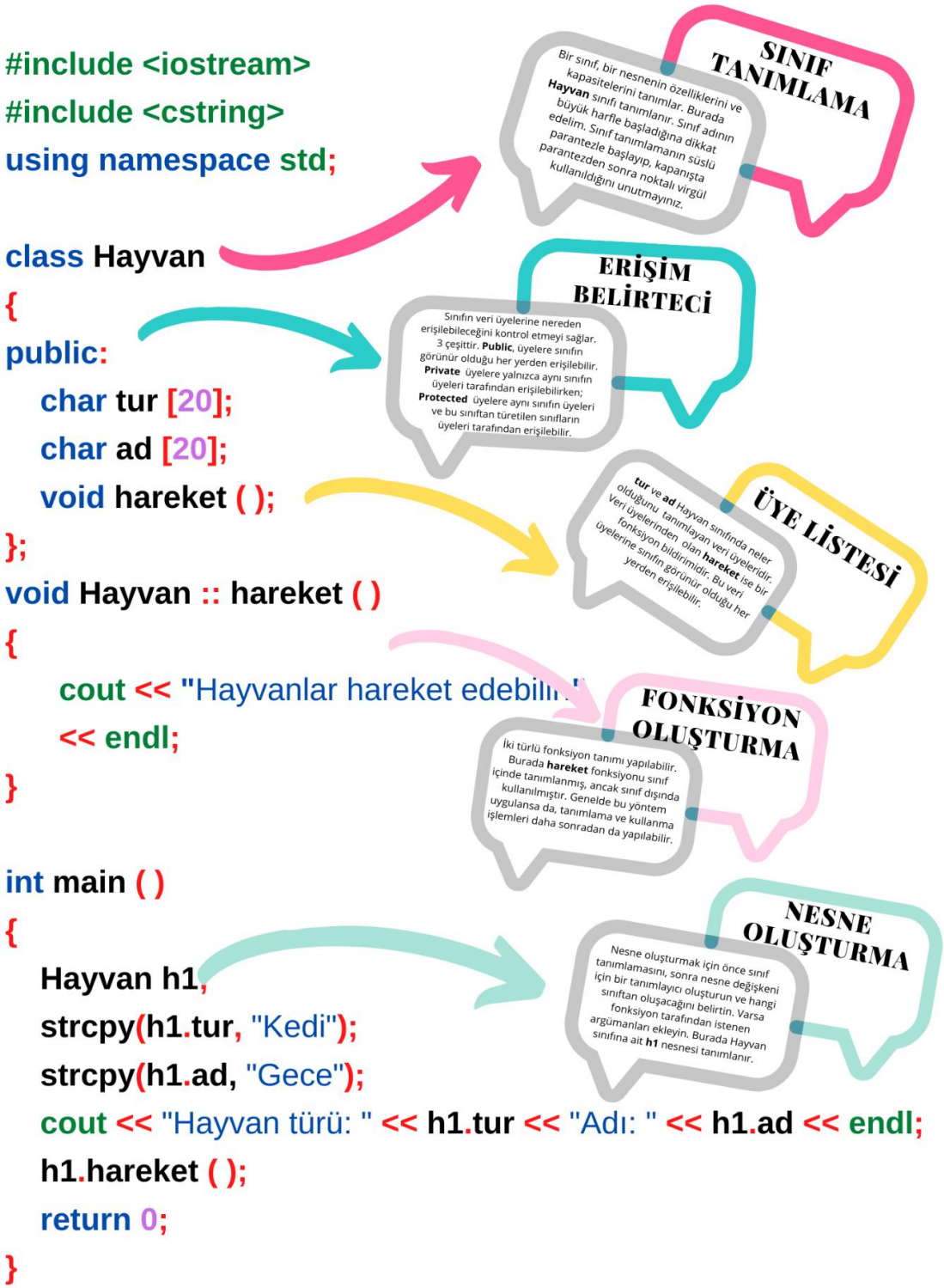
Resim 34. *Araba sınıfı afişi*



Resim 35. *Cep telefonu sınıfı afişi*

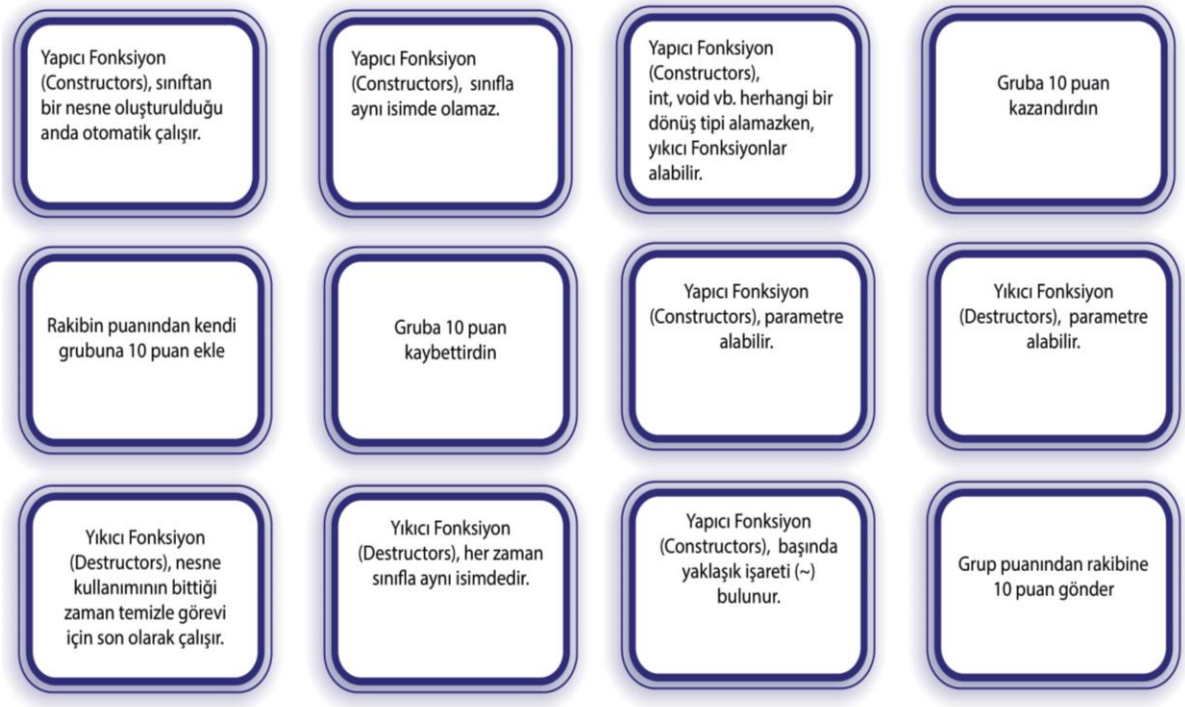


Resim 36. *Kuş sınıfı afişi*

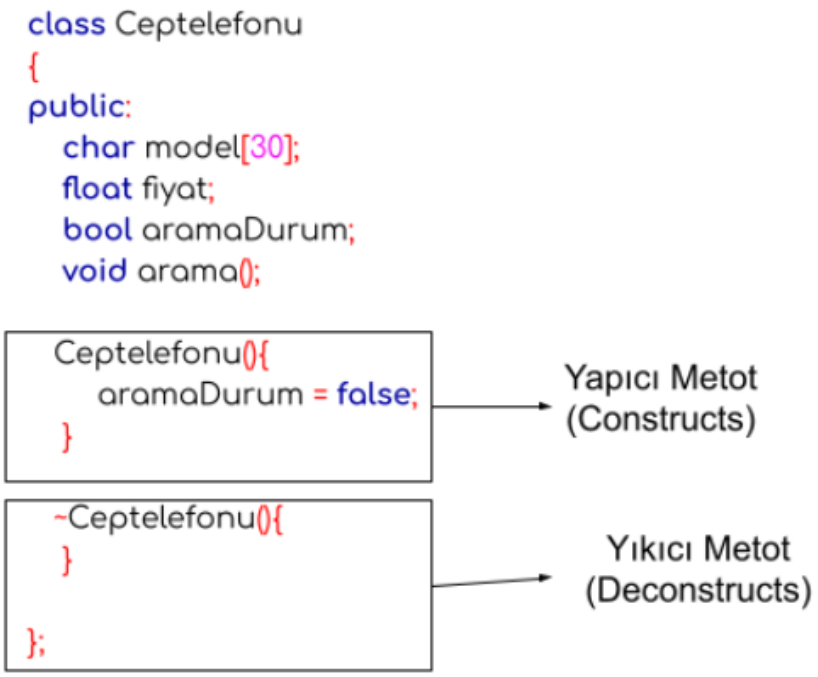


Resim 37. *Hayvan sınıfı afişi*

O6.A4.1. Yarışma Kartları



O6.A4.2. Kod Örneği



O6.A5.1. 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(30); // Fonksiyonu parametre ile
cagirin
    return 0;
}
```

Kodun Çıktısı:

```
Araba: Opel Astra

Hiz: 150
```

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.

Hafta 6. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftaki çevrimiçi ders içeriğinin amacı, C++'ta nesne yönelimli programlama (NYP) kavramlarını öğrenerek, farklı senaryolara uygun şekilde sınıf (class) ve nesne (object) tanımlamayı, kapsülleme (encapsulation), kalıtım (inheritance) ve çok biçimlilik (polymorphism) ilkelerini etkin biçimde kullanmaktır. Gerçek hayattan örneklerle NYP yaklaşımının nasıl uygulandığını kavramak ve Code::Blocks ortamında uygulamalar geliştirerek programların daha düzenli, modüler, güvenilir ve yeniden kullanılabilir biçimde yazılmasını pekiştirmek amaçlanmıştır.

Etkinlik Uygulama Ortamı

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır. Bireysel çalışma ürünleri ise dijital tartışma panosu aracılığıyla toplanacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Günün Rotası (5 dk.)

B. Öğrenme Görevleri

B1. Nesne Yönelimli Programlama ile Futbolcu Bilgilerinin Yönetimi (45 dk.)

Ders Arası (10 dk.)

B2. İki Boyutlu Noktaların Nesne Yönelimli Programlama ile Modellenmesi (50 dk.)

A. Giriş: Günün Rotası

Süre: 5 dk.

Uygulama: Eğitmen, çevrimiçi ders saati başladığında, öğrencilerin sanal sınıfa katılmasını beklerken arka planda odaklanmayı kolaylaştıracak hafif bir enstrümantal müzik açar. Yeterli katılım sağlandığında müziği yavaşça kısar ve öğrencileri sıcak bir şekilde selamlar. Ardından bugünkü dersin yol haritasını tek bir paragrafta net biçimde özetler: Arkadaşlar, bugünkü temel amacımız, C++'ta Nesne Yönelimli Programlama (OOP) kavramlarını öğrenmektir. Bu amaç doğrultusunda sınıf (class) ve nesne (object) tanımlamayı, kapsülleme (encapsulation), kalıtım (inheritance) ve çok biçimlilik (polymorphism) ilkelerinin programlamada nasıl kullanıldığını inceleyeceğiz. Gerçek hayattan örneklerle, örneğin bir öğrenci bilgi sistemi ya da basit bir kütüphane yönetim sistemi tasarlayarak OOP'nin sağladığı düzen, güvenilirlik ve yeniden kullanılabilirlik avantajlarını keşfedeceğiz. Gün sonunda, sınıf ve nesne mantığını kavrayarak Code::Blocks üzerinde kendi OOP tabanlı programlarımızı geliştirebilecek ve modüler, sürdürülebilir uygulamalar tasarlayabiliyor olacağız.

B. Öğrenme Görevleri

B1. Nesne Yönelimli Programlama ile Basketbolcu Bilgilerinin Yönetimi

Süre: 45 dk.

Materyaller: OÇ6.B1.1. Öğrenci Yönerge Metni

OÇ6.B1.2. Kod Bloğu

OÇ6.B1.3. Kod Çıktısı

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ6.B1.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir futbol takımının futbol direktörü, oyuncularının bilgilerini bilgisayarda saklamak istemektedir. Bunun için bir Futbolcu adlı sınıf (class) tasarlamanız beklenmektedir.

Bu sınıf içerisinde:

- oyuncunun ad-soyadını (string),
- forma numarasını (int),
- attığı gol sayısını (int)

tutan üye değişkenler bulunmalıdır.

Ayrıca, sınıfın içinde:

- oyuncu bilgilerini ekrana yazdıran bir fonksiyon,

yer almalıdır. Programınızda bu sınıftan en az 2 futbolcu nesnesi tanımlayınız. Ardından her oyuncunun bilgilerini ekrana yazdırınız.

Zaman Planı:

- **0-10 dakika:** Senaryonun okunması; hangi sınıfların oluşturulacağına, bu sınıfların hangi özellik ve davranışlara sahip olacağına, sınıflar arasında hangi ilişkilerin kurulacağına ve nesneler üzerinden hangi işlemlerin (oluşturma, güncelleme, veri görüntüleme) yapılması gerektiğine yönelik analizlerin gerçekleştirilmesi.
- **10-35 dakika: Kodun öğrenciler tarafından yazılması.** Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **35-45 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B2. İki Boyutlu Noktaların Nesne Yönelimli Programlama ile Modellenmesi

Süre: 50 dk.

Materyaller: OÇ6.B2.1. Öğrenci Yönerge Metni

OÇ6.B2.2. Kod Bloğu

OÇ6.B2.3. Kod Çıktısı

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ6.B2.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: 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

Zaman Planı:

- **0-10 dakika:** Senaryonun okunması; hangi sınıfların oluşturulacağına, bu sınıfların hangi özellik ve davranışlara sahip olacağına, sınıflar arasında hangi ilişkilerin kurulacağına ve nesneler üzerinden hangi işlemlerin (oluşturma, güncelleme, veri görüntüleme) yapılması gerektiğine yönelik analizlerin gerçekleştirilmesi.

- **10-40 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **40-50 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

Hafta 6. Çevrim İçi: Ders Materyalleri

OÇ6.B1.1. Öğrenci Yönerge Metni

Göreviniz: Bir futbol takımının futbol direktörü, oyuncularının bilgilerini bilgisayarda saklamak istemektedir. Bunun için bir Futbolcu adlı sınıf (class) tasarlamanız beklenmektedir.

Bu sınıf içerisinde:

- oyuncunun ad-soyadını (string),
- forma numarasını (int),
- attığı gol sayısını (int)

tutan üye değişkenler bulunmalıdır.

Ayrıca, sınıfın içinde:

- oyuncu bilgilerini ekrana yazdıran bir fonksiyon,

yer almalıdır. Programınızda bu sınıftan en az 2 futbolcu nesnesi tanımlayınız. Ardından her oyuncunun bilgilerini ekrana yazdırınız.

OÇ6.B1.2. Kod Bloğu

```
#include <iostream>
#include <string>
using namespace std;
class Futbolcu
{
public:
    string ad_soyad;
    int forma_no;
    int gol_sayisi;
    Futbolcu(string x_ad_soyad, int x_forma_no, int x_gol_sayisi){
        ad_soyad = x_ad_soyad;
        forma_no = x_forma_no;
        gol_sayisi = x_gol_sayisi;
    }
};
int main()
{
    Futbolcu f1("Arda Alp" , 21 , 13);
    Futbolcu f2("Burakcem", 23, 17);
    cout << f1.ad_soyad << " " << f1.forma_no << " " << f1.gol_sayisi <<
"\n";
    cout << f2.ad_soyad << " " << f2.forma_no << " " << f2.gol_sayisi <<
"\n";
    return 0;
}
```

OÇ6.B1.3. Kod Çıktısı

```
Arda Alp 21 13

Burakcem 23 17
```

OÇ6.B2.1. Öğrenci Yönerge Metni

Göreviniz: 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

OÇ6.B2.2. Kod Bloğu

```
#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;
}
```

OÇ6.B2.3. Kod 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 7. Yüz Yüze: 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 (40 dk.)

Ders Arası (10 dk.)

B2. Kütüphanelerdeki Bazı Fonksiyonları Kullanarak Kodluyorum (50 dk.)

Ders Arası (10 dk.)

B3. Neden Dosyalama İşlemleri Yaparız? (30 dk.)

B4. C++ Dilinde Dosyalama İşlemleri Yapıyorum (20 dk.)

Ders Arası (10 dk.)

B5. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum (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: 40 dk.

Kazanımlar: K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: O7.B1.1. C++ Programlama Dilinde Kütüphaneler

O7.B1.2. Karakter Kütüphanesi

O7.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 "O7.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. Kütüphanelerdeki Bazı Fonksiyonları Kullanarak Kodluyorum

Süre: 50 dk.

Kazanımlar: K1. Program içinde karakter kütüphanesi kullanarak sonucu test eder.

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: O7.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: Eğitmenler C++ programlama dilindeki bazı kütüphane ve hazır fonksiyonları kullanıp uygulaması için hazırlanan görev etkinliğinde en az iki 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 matematik, katar ve karakter kütüphaneleri ilgili materyalleri destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmen sınıfı dolaşarak ve kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler: Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Klavyeden karakterleri sırasıyla girilen “Arda” ismini “a” değişken adıyla karakter dizisinde, “Duru” ismini ise “b” değişken adıyla katarda tutarak ekrana yazdıran kodu oluşturunuz.

Cevap 1:

```
#include<iostream>
using namespace std;
int main()
{
    char a[4];
    char b[5];
    int i;
    cout << "Ilk ismin karakterlerini giriniz: " << endl;
    for(i=0; i < 4; i++) {
        cin >> a[i];
    }
    cout << "Ikinci ismin karakterlerini giriniz: " << endl;
    for(i=0; i < 4; i++) {
        cin >> b[i];
```

```
    }
    b[4] = '\0';
    cout << "Ilk isim: ";
    for(i=0; i < 4; i++) {
        cout << a[i];
    }
    cout << "\nIkinci isim: ";
    cout << b;
    return 0;
}
```

Görev 2: Büyük küçük karışık hâlde yazılmış bir cümleyi her harfi büyük olacak şekilde yazdıralım. “BuGun Hava Cok GUZEl!” cümlesini “Bugun Hava Cok Guzel!” şekline getirelim.

Cevap 2:

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char mesaj[] = "BuGun Hava Cok GUZEl!";

    for(int i=0; i<strlen(mesaj);i++)
    {
        if(i==0 || mesaj[i-1] ==' ')
            mesaj[i] = toupper(mesaj[i]);
        else
            mesaj[i] = tolower(mesaj[i]);
    }
    cout << mesaj;
    return 0;
}
```

Görev 3: Yazdığınız bir programda kullanıcıya bir karakter katarı içerisinde, kaç tane kelimeden oluştuğunu sayabilecek bir program yazınınız.

Cevap 3:

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char mesaj[] = "Bugun hava cok guzel!";
    int kelimeSayisi = 0;
    for(int i=0; i<strlen(mesaj);i++)
    {
        if(mesaj[i] == ' ')
            kelimeSayisi++;
    }
    cout << "Bu cumlede " << kelimeSayisi+1 << " kelime bulunmaktadir.";
    return 0;
}
```

B3. Neden Dosyalama İşlemleri Yaparız?

Süre: 30 dk.

Kazanımlar: K3. Dosyalama işleminin gerekliliğini açıklar.

K4. Dosya kütüphanesi kullanarak program geliştirir.

Materyaller: O7.B3.1. Dosyalama İşlemleri Görev Yaprağı

O7.B3.2. Dosyalama İşlemleri Afişi 1

O7.B3.3. Dosyalama İşlemleri Afişi 2

O7.B3.4. Dosyalama İşlemleri Afişi 3

O7.B3.5. 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

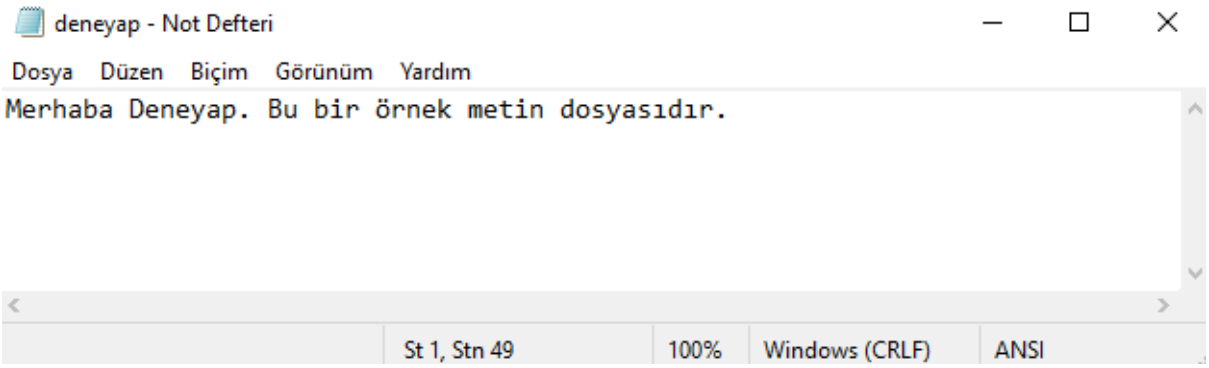
"O7.B3.1. Dosyalama İşlemleri Görev Yaprağı" adlı görev kâğıdını her gruba bir tane verecek şekilde 5 adet hâlinde çıktı alarak derse gelir.

Uygulama: Eğitmen öncelikle her bir öğrenciye "O7.B3.1. 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.	Bir dosyanın açılıp/açılmadığı "is_open()" fonksiyonu ile kontrol edilir. Eğer hatalı bir durum olduysa "0" değeri döndürecektir. Bu şekilde dosyanın düzgün biçimde	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.

fstream ise varsayılan modu **ios::in ve ios::out** modlarıdır.	açıldığı kontrol edilebilir.		

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!";

B4. C++ Dilinde Dosyalama İşlemleri Yapıyorum

Süre: 20 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: O7.B3.5. Dosyalama İşlemleri

O7.B4.1. Kodlar Arasındaki Farkı Bulma

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan "O7.B3.5. 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 "O7.B4.1. Kodlar Arasındaki Farkı Bulma" adlı görev kâğıdını öğrencilerle 5 grubun her birine birer adet olacak şekilde getirir.

Uygulama: Eğitmenler sınıfı dörder kişilik beş gruba ayrılır. Her grubun görebileceği şekilde bu ders için hazırlanan afişi sınıfın belirli bölgelerine asar ve afişi projeksiyon üzerinden yansıtır. Her gruba "O7.B4.1. 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: İlk 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.

B5. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

Süre: 50 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: O7.B3.2. Dosyalama İşlemleri Afişi 1

O7.B3.3. Dosyalama İşlemleri Afişi 2

O7.B3.4. Dosyalama İşlemleri Afişi 3

O7.B3.5. Dosyalama İşlemleri

O7.B5.1. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan afişleri ÖYS üzerinden gönderir. Eğitmen dersin bu bölümü için hazırlanan materyal olan "O7.B5.1 Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum" adlı görev kâğıdını her iki öğrenciye bir çıktı 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 destekleyici bilgi olarak kullanır. Sunu öğrencilere paylaşılır ve ÖYS üzerinden istedikleri görev sayfasındaki destekleyici bilgiyi görmesi sağlanır. Öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmen sınıfı dolaşarak ve kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

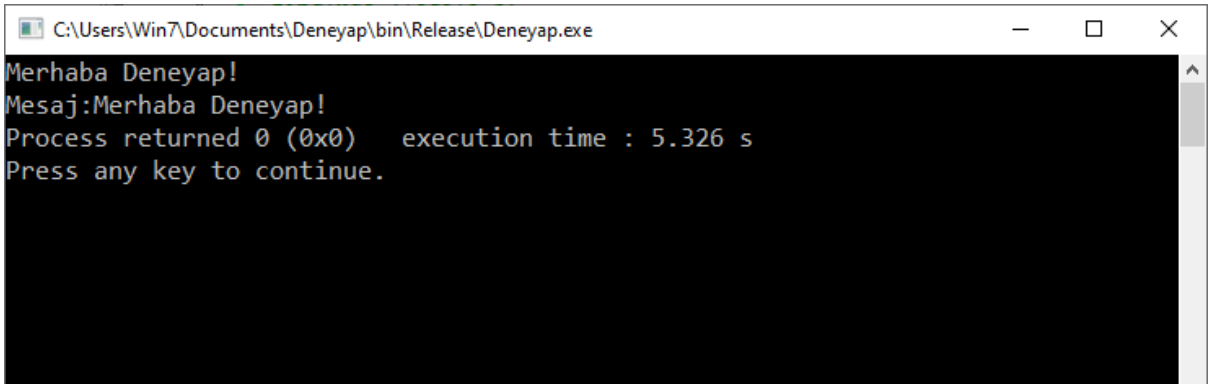
Dosyalama işlemleri ile ilgili verilen görevlerin cevapları aşağıdaki gibidir:

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 38. *Ekran çıktısı*

```
#include <iostream>
#include <fstream>
```

```
#include <string>
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.

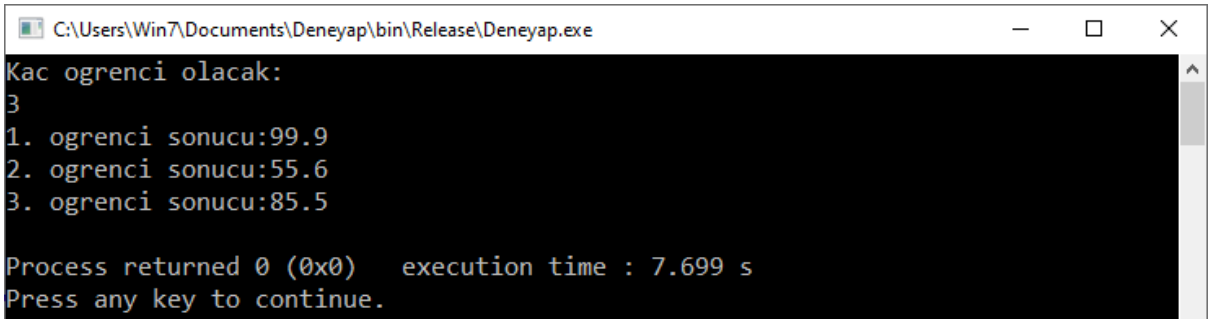
Görev 2: Klavyeden girilen 10 sayıyı dosyaya yazalım.

Cevap 2:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ofstream dosya("deneyap.txt");

    for(int i=0;i<10;i++)
    {
        int sayi;
        cin >> sayi;
        dosya << sayi << endl;
    }

    dosya.close();
}
```

Görev 3: Klavyeden girilen öğrenci sayısı kadar sınav notlarını klavyeden okuyup dosyaya yazdıran programı oluşturunuz.

Cevap 3:



Resim 39. *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();
}
```

Hafta 7. Yüz Yüze: Ders Materyalleri

O7.B1.1. C++ Programlama Dilinde Kütüphaneler

Dikkat!

C++ programlama dili içerisinde geliştiricilerin kullanımına sunulmuş birçok fonksiyonların kütüphaneler içerisinde yer aldığını biliyor muydunuz?

Örneğin; Bu haftaya kadar kullandığımız ***cout*** ve ***cin*** fonksiyonları ***iostream*** isimli kütüphane içerisinde bulunmaktadır.

Dikkat!

Şimdi gelin bu derste C++ programlama dilinde kullanabileceğimiz diğer kütüphaneleri ve bu kütüphanelerdeki bazı fonksiyonları öğrenelim.

O7.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’)	…..

O7.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)	……..

O7.B2.1. C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

Görev 1

Klavyeden karakterleri sırasıyla girilen "Arda" ismini "a" değişken adıyla karakter dizisinde, "Duru" ismini ise "b" değişken adıyla katarda tutarak ekrana yazdıran kodu oluşturunuz.

Görev 2

Büyük küçük karışık halde yazılmış bir cümleyi her harfi büyük olacak şekilde yazdıralım. "BuGun Hava Cok GUZEl!" cümlesini "Bugun Hava Cok Guzel!" şekline getirelim.

Görev 3

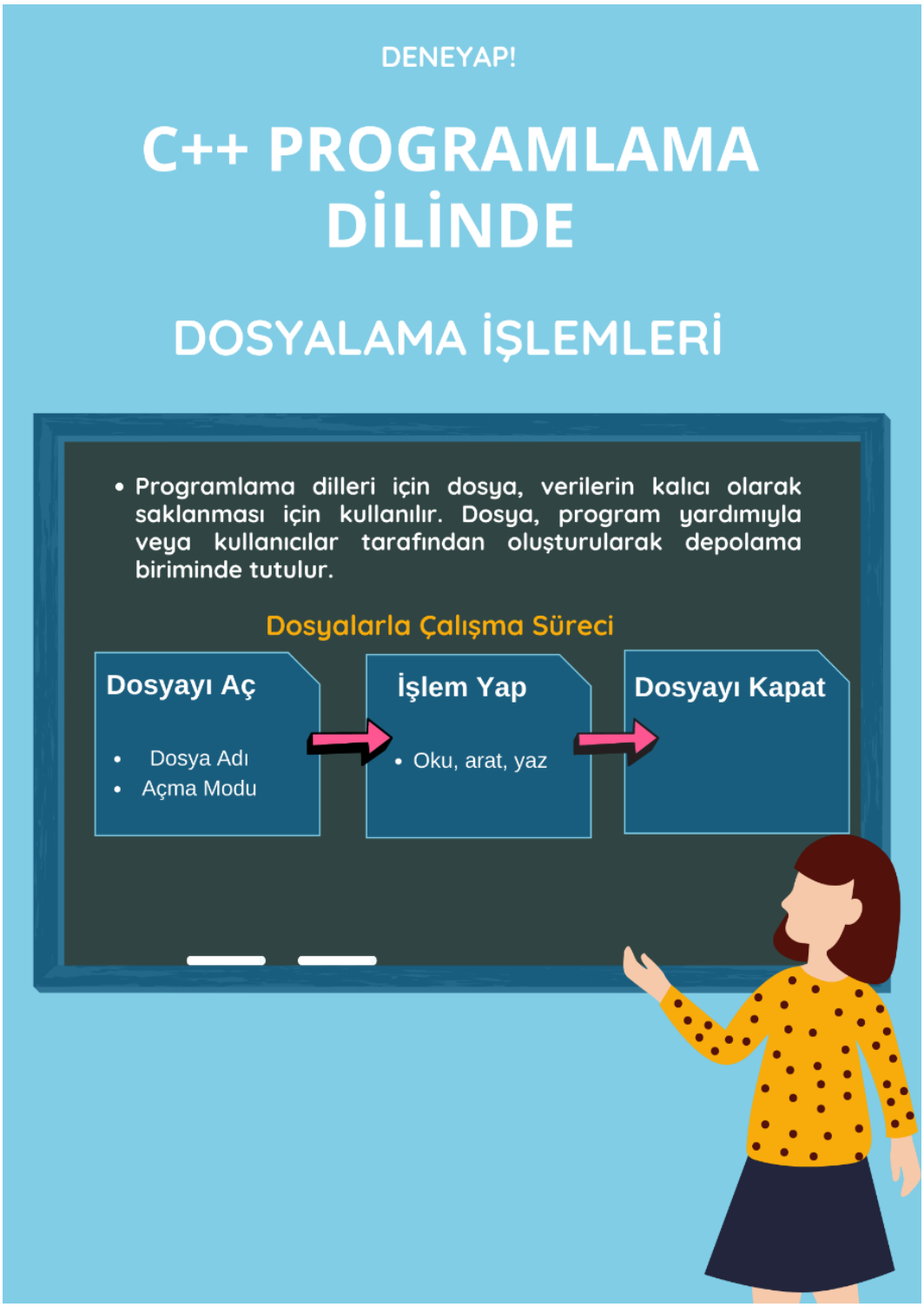
Yazdığınız bir programda kullanıcıya girişte karşılamak isteyeceğiniz mesajın, kaç tane kelimeden oluştuğunu sayabilecek bir program yazınınız.

07.B3.1. 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.

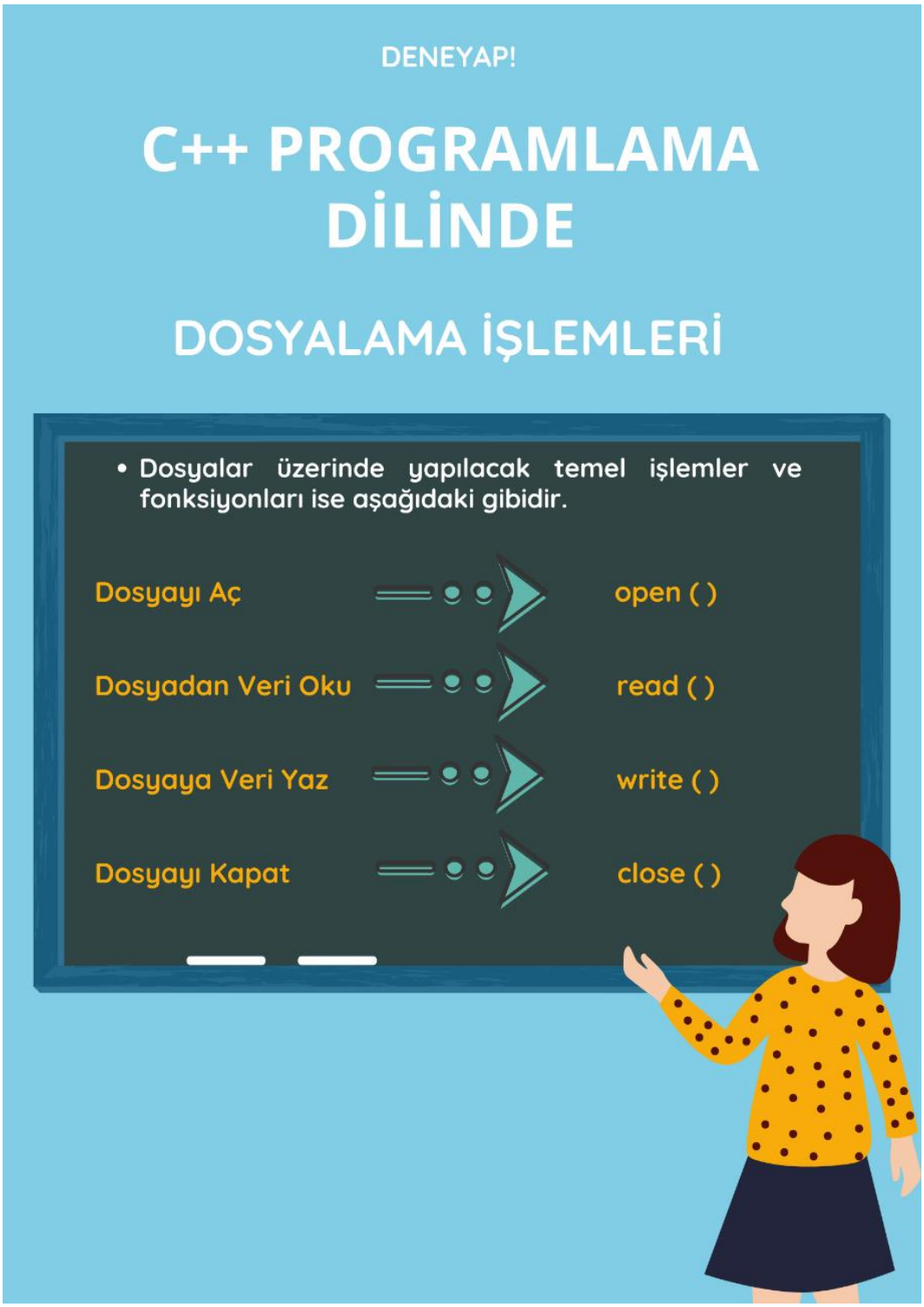
Örnek Kod:
`fstream dosya;`
 	
`dosya.open("deneyap.txt", ios::in | ios::out)`
	Örnek Kod:
	`if(!dosya.is_open())`
	`cout << "Dosya acilamadi!";`
			Örnek Kod:
			`dosya.open("deneyap.txt", ios::nocreate)`
			`if(!dosya.is_open())`
			`cout << "Dosya mevcut değil!";`

O7.B3.2. Dosyalama İşlemleri Yapıyorum Afişi 1



Resim 40. *Dosyalama işlemleri afişi 1*

O7.B3.3. Dosyalama İşlemleri Yapıyorum Afişi 2

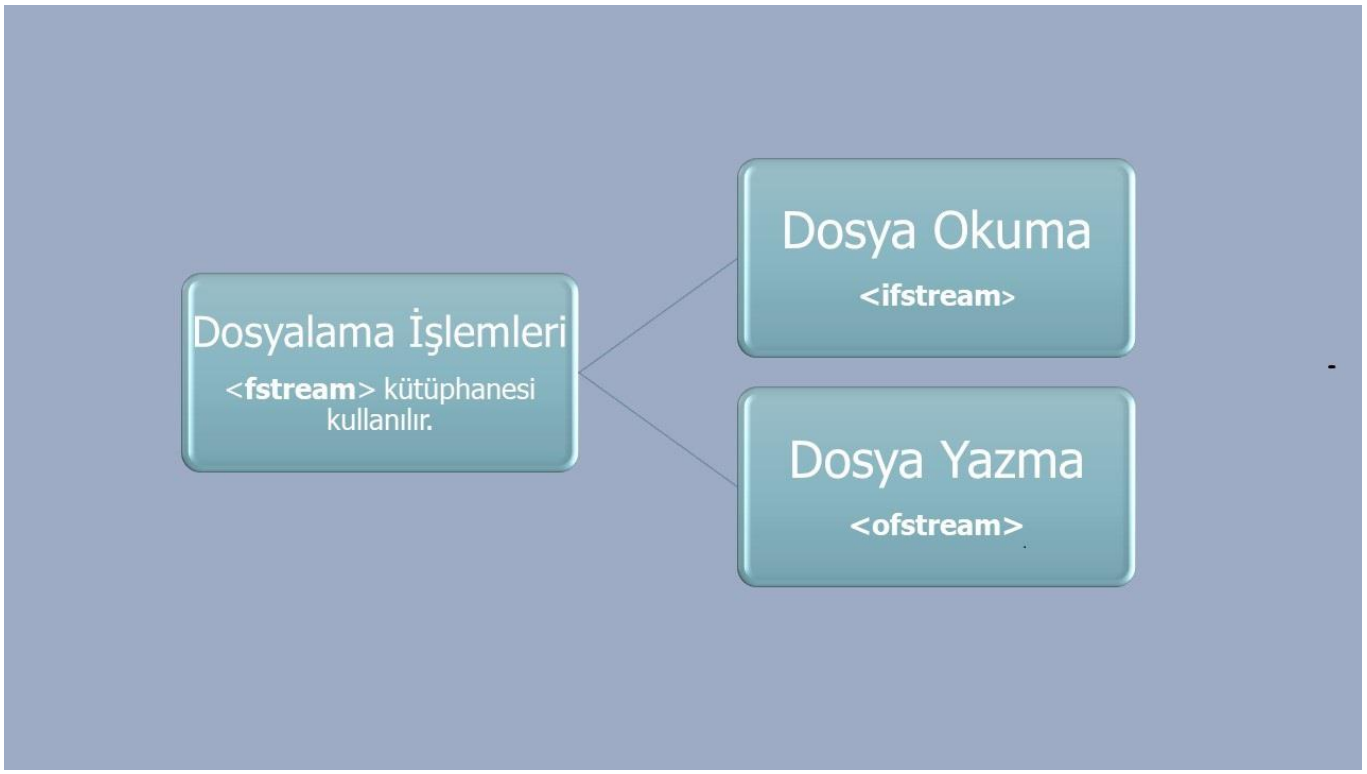


Resim 41. *Dosyalama işlemleri afişi 2*

O7.B3.4. Dosyalama İşlemleri Yapıyorum Afişi 3

Resim 42. *Dosyalama işlemleri afişi 3*

O7.B3.5. Dosyalama İşlemleri



Resim 43. *Dosyalama işlemleri*

O7.B4.1. Kodlar Arasındaki Farkı Bulma

İşlem

```
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
    ofstream dosya;
    dosya.open("deneyap.txt");
    dosya << "Merhaba Deneyap!" << endl;
    dosya.close();
}
```

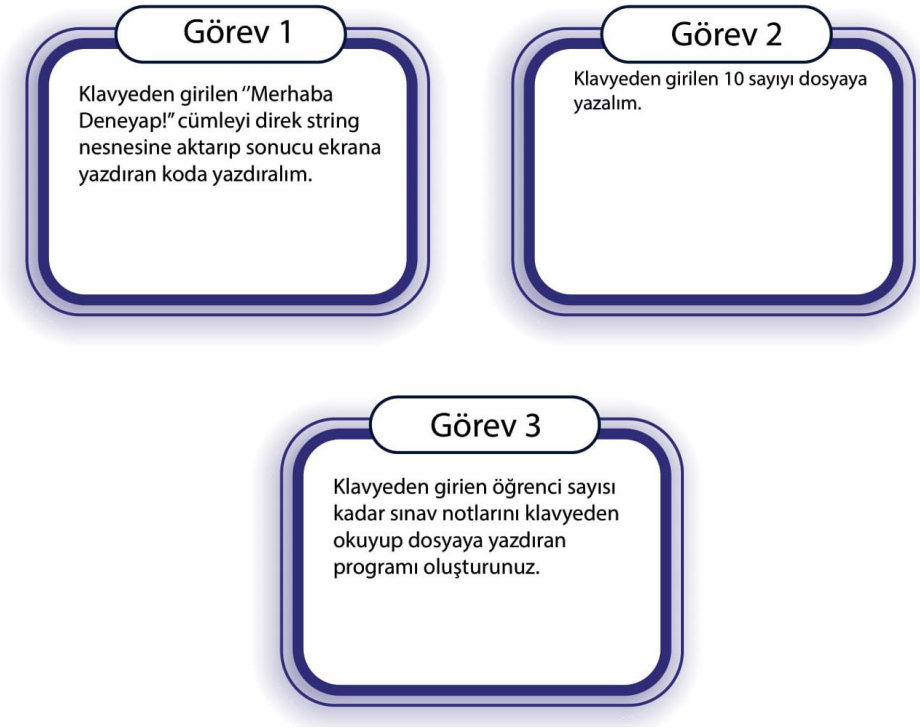
İşlem

```
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
    ofstream dosya;
    dosya.open("deneyap.txt",ios::app);
    dosya<< "Merhaba Deneyap!" << endl;
    dosya.close();
}
```

Dikkat!

Yukarıda verilen kodları dikkatlice gözlemleyerek aralarında ne gibi bir fark olduğunu kodları bilgisayarda yazarak bulmaya çalışalım.

O7.B5.1. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum



Hafta 7. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftaki çevrimiçi ders içeriğinin amacı, C++'ta dosyalama (file handling) ve kütüphane kullanımı kavramlarını öğrenerek, farklı senaryolara uygun şekilde dosyalardan veri okuma, dosyalara veri yazma ve standart kütüphaneleri etkin biçimde kullanmaktır. Gerçek hayattan örneklerle dosya işlemlerinin nasıl uygulandığını kavramak ve Code::Blocks ortamında uygulamalar geliştirerek programlarda verilerin saklanmasını, düzenlenmesini ve okunmasını daha sistematik, güvenilir ve yeniden kullanılabilir biçimde yazmayı pekiştirmek amaçlanmıştır.

Etkinlik Uygulama Ortamı:

1. Eğitim ÖYS ile entegre çalışan ve "breakout rooms" çalışma odaları oluşturma özelliği bulunan Big Blue Button çevrim içi öğrenme platformu aracılığıyla gerçekleştirilir.
2. Sorumlu eğitmen öğrencilerin çalışma odalarını ziyaret ederek, öğrenme ortamına katılım sağlayacaktır.
3. Çalışma odalarında yer alacak öğrenciler uygulama üzerinden rastlantısal olarak gruplara dağıtılacaktır. Eğitmen tarafından belirlenen grup çalışma süresi sonunda ana odaya geri dönen grupların ürünleri, dijital tartışma panosu aracılığıyla toplanacaktır. Bu rehberde dijital tartışma panosu olarak padlet.com kullanılmaktadır.
4. Big Blue Button içerisinde yer alan katılımcı durum özelliklerinin değiştirilmesi, anket sorusu iletme, paylaşılan notlar ve sohbet araçları da eğitimde aktif şekilde kullanılmaktadır.

Önerilen Ders Akışı

A. Giriş: Günün Rotası (5 dk.)

B. Öğrenme Görevleri

B1. Resmi Belgelerde İsim Düzenleme (45 dk.)

Ders Arası (10 dk.)

B2. Klavyeden Girilen Belirli Harflerin Tekrar Sayısını Bulma (25 dk.)

B3. Spor Salonu Günlük Tekrar Takibi (25 dk.)

A. Giriş: Günün Rotası

Süre: 5 dk.

Uygulama: Eğitmen, çevrimiçi ders saati başladığında, öğrencilerin sanal sınıfa katılmasını beklerken arka planda odaklanmayı kolaylaştıracak hafif bir enstrümantal müzik açar. Yeterli katılım sağlandığında müziği yavaşça kısar ve öğrencileri sıcak bir şekilde selamlar. Ardından bugünkü dersin yol haritasını tek bir paragrafta net biçimde özetler: Arkadaşlar, bugünkü temel amacımız, C++'ta dosyalama işlemlerini ve kütüphanelerin nasıl kullanılacağını öğrenmektir. Bu amaç doğrultusunda, dosyalardan veri okuma, dosyalara veri yazma ve standart kütüphanelerin programlarımızı daha işlevsel hale getirmede nasıl kullanıldığını inceleyeceğiz. Gerçek hayattan örneklerle, alışveriş verilerinin okunup işlenmesi gibi senaryolar üzerinden çalışacağız. Gün sonunda, dosyalama ve kütüphane mantığını kavrayarak Code::Blocks üzerinde kendi dosya okuma-yazma işlemlerimizi gerçekleştirebilecek ve farklı kütüphanelerden yararlanarak küçük ama işlevsel programlar geliştirebiliyor olacağız.

B. Öğrenme Görevleri

B1. Resmi Belgelerde İsim Düzenleme

Süre: 45 dk.

Materyaller: OÇ7.B1.1. Öğrenci Yönerge Metni

OÇ7.B1.2. Yanıt

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ7.B1.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Birçok çevrimiçi formda veya resmî belgede kişinin soyadının tamamının büyük harfle yazılması istenir. Kullanıcıdan ad ve soyad bilgisini alan bir program yazarak, soyadını otomatik olarak büyük harfe çeviriniz.

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi dosyaların açılacağına, hangi kütüphanelerin kullanılacağına ve dosya üzerinde hangi işlemlerin (okuma, yazma, güncelleme) yapılması gerektiğine yönelik analizlerin gerçekleştirilmesi.
- **5-25 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **25-35 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B2. Klavyeden Girilen Belirli Harflerin Tekrar Sayısını Bulma

Süre: 25 dk.

Materyaller: OÇ7.B2.1. Öğrenci Yönerge Metni

OÇ7.B2.2. Yanıt

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ7.B2.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Klavyeden girilen cümledeki 'a' veya 'A' karakterlerini sayan programı yazınız.

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi dosyaların açılacağına, hangi kütüphanelerin kullanılacağına ve dosya üzerinde hangi işlemlerin (okuma, yazma, güncelleme) yapılması gerektiğine yönelik analizlerin gerçekleştirilmesi.
- **5-20 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **20-25 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B3. Spor Salonu Günlük Tekrar Takibi

Süre: 25 dk.

Materyaller: OÇ7.B3.1. Öğrenci Yönerge Metni

OÇ7.B3.2. Yanıt

Hazırlık: Etkinliğin uygulanabilmesi için tüm öğrencilerin bilgisayarlarında Code::Blocks programının kurulu olması gerekmektedir. Eğitmenin, ders öncesinde bireysel paylaşımlara ve yorumlara olanak tanıyan bir Padlet panosu oluşturması (duvar ya da görsel taslak şablonunda) ve paylaşım linkini hazır bulundurması beklenir. Bu etkinlik için sunulacak görev öğrencilere sunum üzerinden gösterilir.

Uygulama: Öğrenciler, belirtilen OÇ7.B3.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir spor salonunda, antrenör bir öğrenciye 100 günlük bir antrenman takvimi oluşturmuştur. Antrenör bazı hareketleri öğrencinin antrenmanda geçirilen gün sayısı kadar tekrar etmesi istenmiştir. Daha sonra antrenör öğrencilerinin tek sayılı günlerde (1, 3, 5, …, 99) yaptıkları tekrarların toplamını kaydetmek istemektedir. Bu amaçla 1 ile 100 arasındaki tek sayıların toplamını hesaplayıp bir dosyaya yazan bir program tasarlayınız.

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi dosyaların açılacağına, hangi kütüphanelerin kullanılacağına ve dosya üzerinde hangi işlemlerin (okuma, yazma, güncelleme) yapılması gerektiğine yönelik analizlerin gerçekleştirilmesi.
- **5-25 dakika:** Kodun öğrenciler tarafından yazılması. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **25-30 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

Hafta 7. Çevrim İçi: Ders Materyalleri

OÇ7.B1.1. Öğrenci Yönerge Metni

Göreviniz:

Birçok çevrimiçi formda veya resmî belgede kişinin soyadının tamamının büyük harfle yazılması istenir. Kullanıcıdan ad ve soyad bilgisini alan bir program yazarak, soyadını otomatik olarak büyük harfe çeviriniz.

OÇ7.B1.2. Kod Bloğu

```
#include <iostream>
#include <cstring>
using namespace std;
int main(){
    char mesaj[] = "Ahmet Yilmaz";
    bool bosluk = false;
    for(int i=0; i<strlen(mesaj);i++) {
        if(bosluk)
            mesaj[i] = toupper(mesaj[i]);
        if(mesaj[i] == ' ')
            bosluk = true;
    }
    cout << mesaj ;
    return 0;
}
```

OÇ7.B2.1. Öğrenci Yönerge Metni

Göreviniz:

Klavyeden girilen cümledeki ‘a’ veya ‘A’ karakterlerini sayan programı yazınız.

OÇ7.B2.2. Kod Bloğu

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char mesaj[100];
    cin.getline(mesaj,100);
    int sayac = 0;
    for(int i=0;i<strlen(mesaj);i++)
    {
        if(mesaj[i]=='a' || mesaj[i]=='A')
            sayac++;
    }
    cout << "Bu cumlede "<< sayac <<" adet a vardir.";
    return 0;
}
```

OÇ7.B3.1. Öğrenci Yönerge Metni

Göreviniz:

Bir spor salonunda, antrenör bir öğrenciye 100 günlük bir antrenman takvimi oluşturmuştur. Antrenör bazı hareketleri öğrencinin antrenmanda geçirilen gün sayısı kadar tekrar etmesi istenmiştir. Daha sonra antrenör öğrencilerinin tek sayılı günlerde (1, 3, 5, …, 99) yaptıkları tekrarların toplamını kaydetmek istemektedir. Bu amaçla 1 ile 100 arasındaki tek sayıların toplamını hesaplayıp bir dosyaya yazan bir program tasarlayınız.

OÇ7.B3.2. Kod Bloğu

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int toplam = 0;
    for(int i=1;i<100;i++)
        toplam += i;
    ofstream dosya("sonuc.txt");
    if(dosya.is_open())
    {
        dosya << toplam;
    }
    else
    {
    cout << "Dosya Okunamadi!";
    }
}
```

Dosya İçeriği Görünümü: