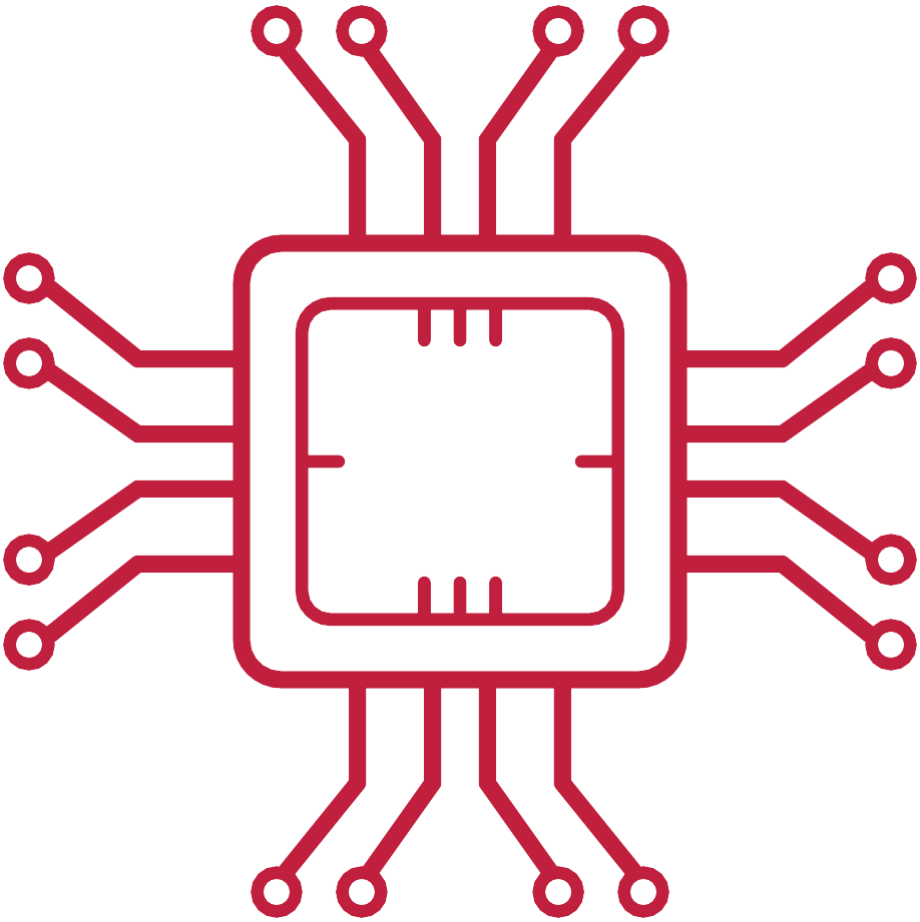
#MİLLİ TEKNOLOJİ HAMLESİ

DENEYAP TÜRKİYE

YAZILIM TEKNOLOJİLERİ

LİSE



TÜBİTAK

TÜRKİYE TEKNOLOJİ TAKIMI

DENEYAP

Teknoloji Atölyeleri

YAZILIM TEKNOLOJİLERİ

LİSE

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

TÜBİTAK

TÜBİTAK Deneyap Kitapları 8

YAZILIM TEKNOLOJILERI
LİSE

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 978-605-312-443-6
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 gençlerimizin adaptasyon sürecinde hazırlanan öğretim programlarının şekli ve uygulayıcıların kullandıkları öğrenme öğretme yöntemleri kilit noktadır.

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

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

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

Sizlere ve ülkemiz öğrencilerine faydalı olması dileğiyle!

Yazılım Teknolojileri Dersi Öğretim Planı Uygulama Kılavuzu

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

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

Yazılım Teknolojileri Dersi Bilgi Paketi

Dersin Amacı

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

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

Dersin Çıktıları

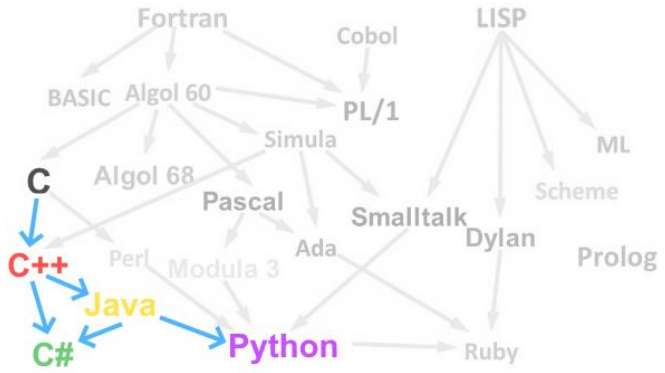
Bu dersi alan öğrenciler;

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

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

Neden C++ Programlama Dili

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



Resim 1. *Programlama dillerine ait aile ağacı*

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

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

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

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

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

Ders Haftalık Planlaması

- **Hafta 1:**

  Yüz Yüze: Programlamaya Giriş, Programlama Terimleri, Matematiksel İşlemler, Karşılaştırma İşlemleri, Mantıksal İşlemler, Sayı Sistemleri, Algoritma Tasarımı, Operatörler ve Kavramlar, 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, Veri Tipi Dönüşümleri, Kaçış Dizgeleri, Operatörlerin Kullanımı, Bit İşlemler

  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, Veri Soyutlama, Kapsülleme, Kalıtım, Polimorfizm, Aşırı Yükleme, Geçersiz Kılma

  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.

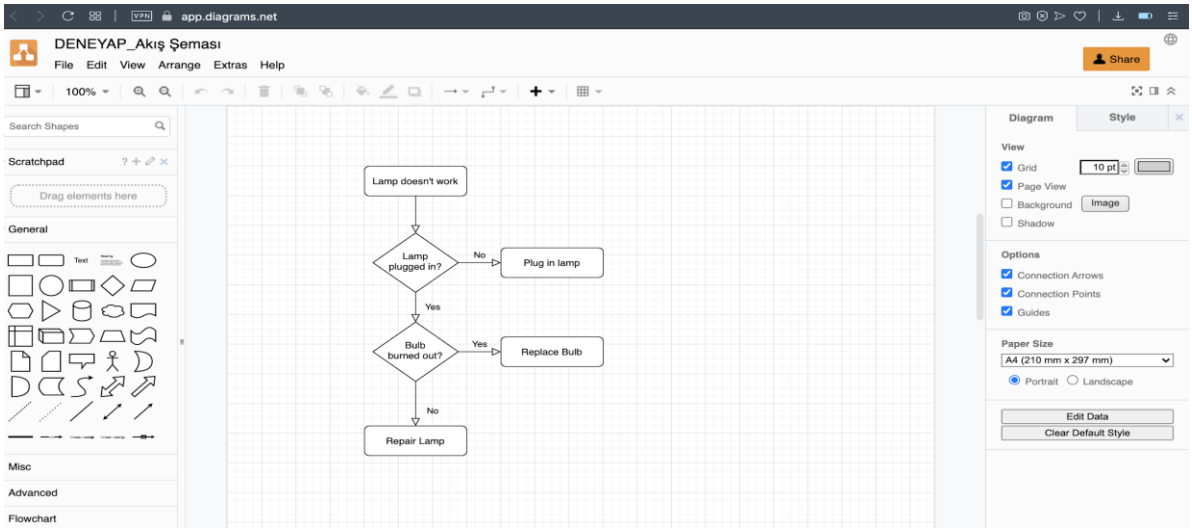
Sunulan bu yapılandırılmış öğretim modeli, 4C/ID'nin temel felsefesini hayata geçirerek öğrenme ve değerlendirme süreçlerini birbirinden ayrılmaz bir bütün haline getirir. Yüz yüze oturumlarda sunulan destekleyici bilgiler (Bileşen 2), öğrencileri senkron çevrim içi oturumlardaki hibrit kısmi görev pratiğine (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:

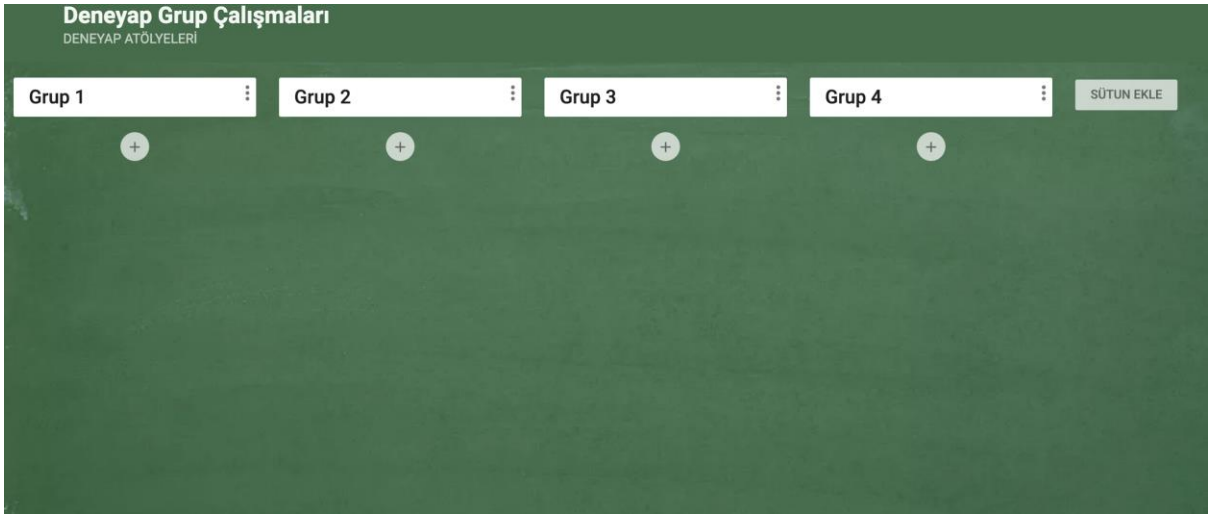
- Dersle ilgili genel duyuruların iletilmesi
- Öğrencilere sunulacak öğrenme materyallerinin paylaşılması
- Öğrenme görevi sırasında kullanılacak diğer teknolojik araçların iletilmesi
- Kısmi öğrenme görevleri gibi bireysel görevlilerin teslimi
- Grup olarak oluşturulacak ürün ya da tamamlanan görevlerin 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.

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. Programlama türlerini ayırt eder.
- K3. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.
- K4. Değişkenlere farklı türlerde değer ataması yapar.
- K5. Algoritma temel kavramlarını bilir ve analiz eder.
- K6. Bir algoritmayı akış şemasında gösterir.
- K7. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını bilgisayar ortamında oluşturur.
- K8. Bir problemin çözümünde değişkenleri kullanır.
- K9. Verilen algoritmanın akış diyagramını sözde koda dönüştürür ve değişkenleri ayırt eder.

Amaç

Bu haftanın amacı, programlamaya dünyasına giriş yapılarak öğrencilerin temel düzeyde bildikleri matematiksel, karşılaştırma ve mantıksal işleçleri (operatörleri) kullandırmak ve sayı sistemlerinin keşfedilmesidir. Ayrıca kâğıt üzerinde bir problemi çözmek için kod yazmada gerekli algoritmik düşünme yeteneğini geliştirmek ve problemin çözümünü bilgisayar kullanarak aktarmak hedeflenir.

Önerilen Ders Akışı

A. Giriş: Çizgi-Nokta Oyunu (15 dk.)

B. Öğrenme Görevleri

B1. Nasıl Anlatırsın? (20 dk.)

B2. Programlama Türlerini Karşılaştıralım (15 dk.)

Ders Arası (10 dk.)

B3. Beni Bul ve Değiştir (30 dk.)

B4. Farklı Atama Türlerini Tanıyalım (20 dk.)

Ders Arası (10 dk.)

B5. Algoritmayı Tanıyalım (30 dk.)

B6. Algoritmaları Eşleştirelim (20 dk.)

Ders Arası (10 dk.)

B7. Çiftleri Toplayalım (15 dk.)

B8. Değişkenlerin Önemi (15 dk.)

B9. Algoritmayı Test Edelim (20 dk.)

A. Giriş: Çizgi-Nokta Oyunu

Süre: 15 dk.

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

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

B. Öğrenme Görevleri

B1. Nasıl Anlatırsın?

Süre: 20 dk.

Kazanımlar: K1. Temel programlama terimlerini açıklar.

Materyaller: Bilgisayar ya da akıllı tahta

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

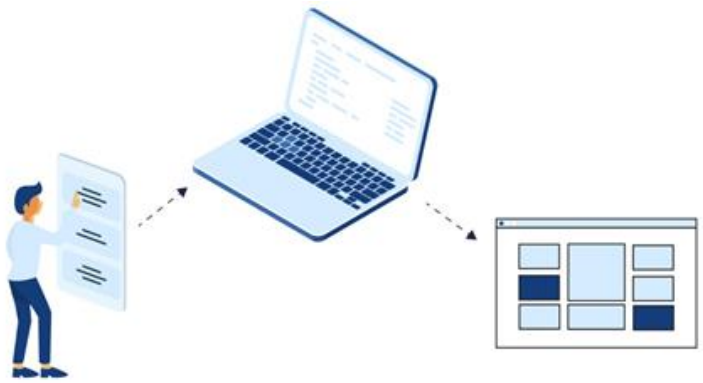
kavramları beyin fırtınası yoluyla özetlenir. Sonuç olarak öğrencilerin aşağıdaki içeriğe ulaşmaları sağlanır.

Konu içeriği:

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. Programlamanın tanımı, "çeşitli görevleri ya da işlemleri gerçekleştirmek için bilgisayara talimat verme" olarak verilebilir. Bilgisayarlar güçlü ve hızlı sistemler olmalarına rağmen insanlar gibi akıllı varlıklar olmadıkları için ne yapacaklarını bildiren bazı talimatlara ihtiyaç duyarlar. Programlama, bir bilgisayara belirli görevleri yapmasını sağlamak amacıyla mantıksal talimatlar dizisi (algoritmalar) yazma sürecidir. Bu talimatlar, programlama dilleri kullanılarak yazılır ve bilgisayarın anlayabileceği bir biçime (genellikle makine koduna) dönüştürülür. Bilgisayarlar çok hızlı işlem yapabilen elektronik cihazlardır ancak kendi başlarına düşünemez, karar veremez ya da bir işi nasıl yapacaklarını bilemezler. Bu nedenle, ne yapacaklarını, hangi verileri işleyeceklerini ve hangi koşullarda nasıl davranacaklarını bir insan tarafından açık ve kesin şekilde tarif etmek gerekir. İşte bu tarif, programlama ile yapılır.



Resim 5. *Programlama süreci*

Tıpkı insanların farklı doğal dilleri konuşması gibi (Türkçe, İngilizce, Fransızca...), bilgisayarlar da farklı programlama dillerini "konuşabilir". Ancak burada önemli bir fark vardır. İnsan dilleri esnek, bağlama göre anlam kazanan dillerdir. Bilgisayar dilleri ise katı ve kesin kurallara sahiptir; en küçük yazım hatası bile kodun çalışmasını engeller. Programlama,

aslında bilgisayara "ne yapacağını, ne zaman yapacağını ve nasıl yapacağını" adım adım anlatmaktır. Bu talimatlar genellikle üç tür işlem içerir:

1. Girdi alma (örneğin: kullanıcıdan bir sayı isteme)
2. İşleme (örneğin: bu sayının karesini hesaplama)
3. Çıktı verme (örneğin: sonucu ekrana yazdırma)

Programlama sayesinde bilgisayarlara basit görevler (örneğin: 5 + 3'ü hesapla) veya karmaşık görevler (örneğin: bir hava tahmin sistemi, bir sosyal medya uygulaması, bir oyun motoru) yaptırmak mümkündür. Bu görevler tek bir komutla da yapılabilir, binlerce satırlık kod içeren sistemler de gerektirebilir.

Daha önce de belirtildiği gibi bilgisayarlar, belirli bir sözdizimsel kurala (syntax) göre yazılmış talimatları anlayabilir. Bu kurallar bütününe programlama dili adı verilir. Programlama dili, programcının bir görevi ifade etmesine ve bilgisayarın bu ifadeyi anlayarak yürütmesine olanak sağlar. Günümüzde farklı amaçlara hizmet eden birçok programlama dili bulunmaktadır. Bunlardan bazıları:

- C: Sistem programlama ve donanım seviyesine yakın uygulamalar için kullanılır.
- C++: Nesne yönelimli programlama özelliklerine sahip, C diline dayalı güçlü bir dildir.
- Python: Basit sözdizimi ve geniş kütüphane desteği ile popüler bir yüksek seviyeli dildir.
- Java: Platformdan bağımsız uygulamalar geliştirmeye olanak tanıyan nesne yönelimli bir dildir.

Bu ders kapsamında, özellikle C++ programlama dili kullanılacaktır. C++, hem yapısal hem de nesne yönelimli programlama paradigmalarını desteklemesi sayesinde hem başlangıç düzeyi hem de ileri seviye uygulamalar için tercih edilen güçlü bir dildir. C++ diline ait temel yapılar, sözdizimi ve kullanım örnekleri ilerleyen bölümlerde ayrıntılı olarak ele alınacaktır.

B2. Programlama Türlerini Karşılaştıralım

Süre: 15 dk.

Kazanımlar: K2. Programlama türlerini ayırt eder.

Materyaller: L1.B2.1. Bilgi Kartları

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

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

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

1. Yapısal Programlama: Yapısal programlama, bir programın adım adım ve sıralı şekilde çalışacak biçimde tasarlandığı bir programlama yaklaşımıdır. Genellikle programlamaya yeni başlayanların tercih ettiği bu yöntem, basit ve anlaşılır kod yazmayı teşvik eder. Temel özellikleri:

- Program bir bütün olarak ele alınır.
- Kodlar genellikle başlangıçtan sona doğru sıralı olarak çalıştırılır.
- Alt programlar (fonksiyonlar) kullanılarak kod tekrarından kaçınılır.
- Okunabilirliği ve bakım kolaylığını artırır.
- Kodun anlaşılması kolaydır, küçük çaplı uygulamalar için idealdir.

2. Modüler Programlama: Modüler programlama, yazılımın farklı işlevlerinin modüller (birimler) hâlinde yazıldığı bir programlama tekniğidir. Her modül, belirli bir görevi yerine getirir ve bağımsız şekilde geliştirilebilir. Temel özellikleri:

- Program, birden fazla bağımsız modüle bölünür.
- Her modül kendi içinde çalışabilir, ancak gerektiğinde diğer modüllerle veri alışverişi yapabilir.
- Kodun test edilmesi, yeniden kullanımı ve bakımı kolaylaşır.
- Büyük ve karmaşık yazılımların yönetimini kolaylaştırır.

3. Nesne Yönelimli Programlama: Nesne yönelimli programlama, yazılımı nesneler ve bu nesneler üzerindeki işlemler (metotlar) üzerinden modelleyen bir programlama yaklaşımıdır. Gerçek dünya kavramları (araba, öğrenci, banka hesabı vb.) programda doğrudan temsil edilebilir. Temel özellikleri:

- Veriler ve bu veriler üzerinde yapılacak işlemler bir sınıf altında tanımlanır.
- Bu sınıflardan oluşturulan örneklere nesne denir.
- Nesneler birbirlerine mesaj göndererek iletişim kurar.
- Yeniden kullanılabilirliği ve genişletilebilirliği yüksektir.
- Gerçek dünya problemlerini modellemek kolaydır; büyük projelerde esneklik sağlar.

B3. Beni Bul ve Değiştir

Süre: 30 dk.

Kazanımlar: K3. Problem çözme sürecinde aritmetik, mantıksal ve karşılaştırmalı işleçleri kullanır.

Materyaller: L1.B3.1. Temel İşlemler Tablosu

L1.B3.2. İşleç Bilgi Kartları

L1.B3.3. Örnek Olay Sunum Kartı

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

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

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

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

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

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

Konu içeriği:

Aritmetik İşlemler

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

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

Tablo 2. *Matematiksel ifadeler ve bilgisayar kodlamaları*

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

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

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

***Tablo 3**. Aritmetik işleçler*

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

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

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

```
a = 5

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


c = 2

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


a = 5
```

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

c = 2

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

Karşılaştırma İşlemleri

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

Tablo 4. *Karşılaştırma işleçleri*

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

Mantıksal İşlemler

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

Tablo 5. *Mantıksal işleçler*

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

Mantıksal işleçlerin sonucu şu şekilde belirlenmektedir. Eğer x&&y kullanılıyor ise her iki değişkenin değeri "1" olursa sonuç "1" olur, aksi hâlde sonuç "0" olur. Eğer x||y kullanılıyor ise her iki değişkenin değeri "0" olursa sonuç "0" olur, aksi hâlde sonuç "1" olur. Eğer !x

kullanılıyor ise değişkenin değeri “1” ise sonuç “0”, “0” ise sonuç “1” olacaktır. Mantıksal VE için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 6. *Mantıksal VE kullanımı*

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

Mantıksal VEYA için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 7. *Mantıksal VEYA kullanımı*

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

Mantıksal DEĞİL için aşağıdaki tabloyu inceleyip çalışma prensibini kontrol edebilirsiniz.

Tablo 8. *Mantıksal DEĞİL kullanımı*

İşleç	İşlenen 2	Sonuç
!	Sıfırdan farklı	0
!	0	1

B4. Farklı Atama Türlerini Tanıyalım

Süre: 20 dk.

Kazanımlar: K4. Değişkenlere farklı türlerde değer ataması yapar.

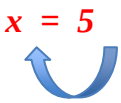
Materyaller: L1.B4.1. Yer Değiştirme Görev Kartları

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

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

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

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



x değişkenine 5 değeri atanıyor.

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

Konu İçeriği:

Farklı tür atama operatörleri aşağıda verilmektedir.

Tablo 9. *Atama operatörleri*

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

*=	Bu işleç '*' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değerle çarpar ve ardından sonucu soldaki değişkene atar.	x*=y (x=x*y)
/=	Bu işleç '/' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere böler ve ardından sonucu soldaki değişkene atar.	x/=y (x=x/y)
%=	Bu işleç '%' ve '=' operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere göre modunu alır ve ardından sonucu soldaki değişkene atar.	x%=y (x=x%y)

B5. Algoritmayı Tanıyalım ve Yazalım

Süre: 30 dk.

Kazanımlar: K5. Algoritma temel kavramlarını bilir ve analiz eder.

K6. Bir algoritmayı akış şemasında gösterir.

Materyaller: L1.B5.1. Doğru Algoritma Hangisi?

L1.B5.2. Akış Diyagramları Bilgi Afişi

L1.B5.3. App.diagrams.net Kullanım Kılavuzu

Hazırlık: Ders materyali ÖYS üzerinden öğrenci erişimine açılır.

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

Konu İçeriği:

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.

Algoritma yazarken, bilgileri uygun biçimde işleyebilmek için bazı temel kavramlar ve yapılar kullanılır. Bu kavramların doğru anlaşılması, algoritmanın hem doğru hem de okunabilir şekilde yazılmasını sağlar.

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

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

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

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

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

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

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

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

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

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

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

- *Bir bitiş noktası olmalı*

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

Eğitmene Öneriler: App.diagrams.net programında akış şemaları çizim özellikleri öğrencilere açıklanmalıdır.

B6. Algoritmaları Eşleştirelim

Süre: 20 dk.

Kazanımlar: K5. Algoritma temel kavramlarını bilir ve analiz eder.

K6. Bir algoritmayı akış şemasında gösterir.

Materyaller: L1.B6.1. Algoritmaları Eşleştirelim Çalışma Kâğıdı

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

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

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

a) Metin olarak yazım: Bu yöntemde algoritmanın çözüm adımları düz metin olarak, açık ve anlaşılır cümlelerle sıralanır. Her adım numaralandırılır ve genellikle "Başla" ifadesiyle başlar, "Bitir" ifadesiyle sonlanır.

b) Sözde kod (pseudocode) yazım: Sözde kod, programlama diline çok yakın, ancak doğal dille yazılmış bir algoritma şeklidir. Koşullar (eğer, iken), döngüler, matematiksel işlemler ve değişken tanımları içerir. Kod yazımına geçmeden önce taslak niteliğindedir. Sözde kod konuşma dilinde ve programlama mantığı altında, "eğer", "iken" gibi koşul kelimeleri ve '>', '=', '<' gibi ifadeler ile birlikte yazılır. Algoritma yazma ile kod yazma arasında kalan bir kısımdır ama kodlamaya daha yakındır. İyi yazılmış sözde koddan bir programlama diline kolaylıkla geçiş yapabilirsiniz.

c) Akış diyagramlarının çizilmesi: Akış diyagramları, algoritmanın adımlarını semboller ve oklar ile görsel olarak ifade eder. Her işlem bir kutucuk ile temsil edilir ve süreçler oklarla birbirine bağlanır. Akış şemalarının algoritmadan farkı, algoritma adımlarının semboller şeklinde kutulara yazılması ve adımlar arasındaki ilişki ve yönün oklar ile gösterilmesidir.

B7. Çiftleri Toplayalım

Süre: 15 dk.

Kazanımlar: K7. Bir problemin çözümüne uygun farklı algoritmalar tasarlar ve akış şemasını bilgisayar ortamında oluşturur.

Materyaller: L1.B7.1. Çiftleri Toplama Algoritması

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

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

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

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

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

B8. Değişkenlerin Önemi

Süre: 15 dk.

Kazanımlar: K8. Bir problemin çözümünde değişkenleri kullanır.

Materyaller: L1.B8.1. Örnek Kod Çalıştırma Tablosu

Hazırlık: Materyaller ÖYS üzerinden materyal olarak öğrencilere yayınlanır.

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

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

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

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

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

Tablo 10. *Örnek kod çalıştırma tablosu*

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

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

Değişken, zamanla değeri değişebilen bir bilgi saklayıcısıdır. Günlük hayattan örnek verecek olursak, bir kişinin yaşı bir değişkendir. Örneğin, "yas" adlı değişkende şu an 18 değeri olabilir. Zaman geçtikçe bu değer artar ve değişkenin değeri güncellenir. Burada "yas" değişkenin adı, "18" ise o anki değeridir. Bilgisayar programlarında değişkenler, verileri geçici olarak hafızada tutmak için kullanılır. Bir program çalışırken bu değişkenlere değer atanabilir, bu değerler üzerinde işlemler yapılabilir ve gerektiğinde bu değerler değiştirilebilir. Program kapandığında ise bu değerler hafızadan silinir, yani değişkenler program çalıştığı sürece geçerlidir.

Algoritma ya da program yazarken değişken isimleri kısa, açıklayıcı ve anlamlı olmalıdır. Tanımlayıcı kurallarına uygun şekilde isimlendirme yapılmalıdır.. 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: 20 dk.

Kazanımlar: K9. Verilen algoritmanın akış diyagramını sözde koda dönüştürür ve değişkenleri ayırt eder.

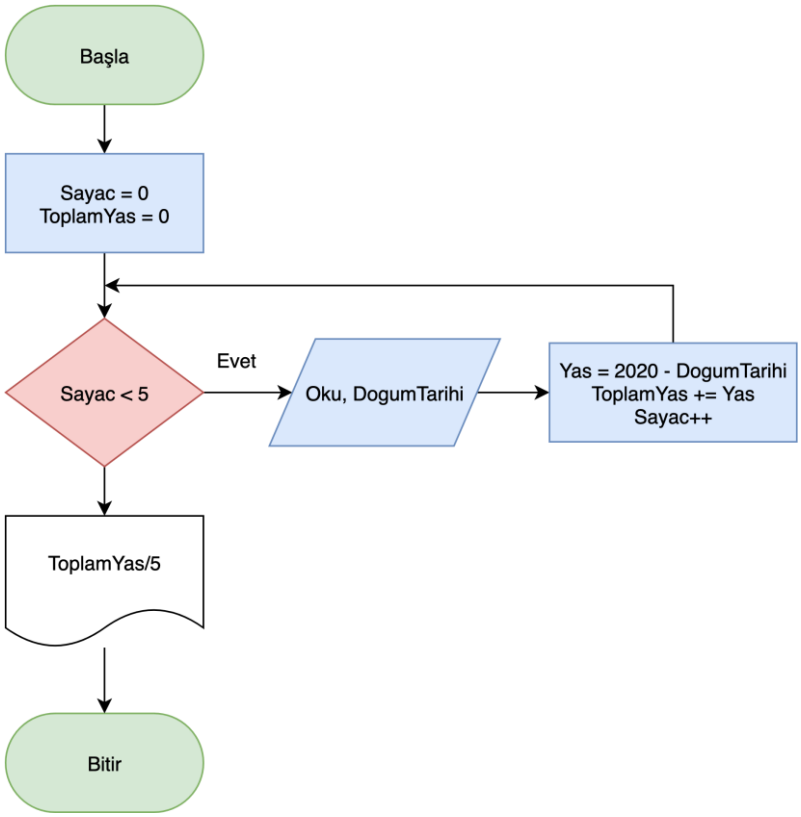
Materyaller: L1.B9.1. Kod Çalıştırma Tablosu

Hazırlık: App.diagrams.net uygulaması öğrenci bilgisayarlarında kurulu olmalıdır. Öğrencilere kod çalıştırma tablosunun ÖYS üzerinden iletilir.

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

> *Görev 1: Sevdiğiniz 5 kişinin doğum tarihlerini girerek yaşlarının ortalamasını bulan bilgisayar programı için akış diyagramı ve sözde kodunu App.diagrams.net uygulamasını kullanarak bilgisayarda çiziniz.*

Öğrencilerin çizmesi gereken akış şeması aşağıdaki gibi olmalıdır:



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

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

Hazırlanacak kod çalıştırma tablosu aşağıdaki gibi olmalıdır.

Tablo 11. *Kod çalıştırma tablosu*

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

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

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

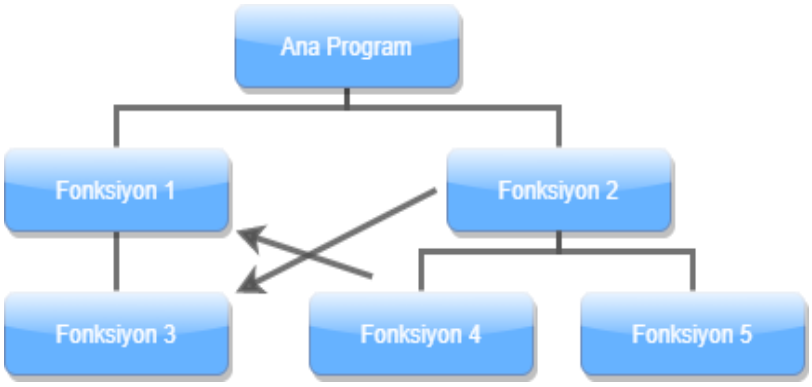
Tablo 12. *Örnek kod çalıştırma tablosu*

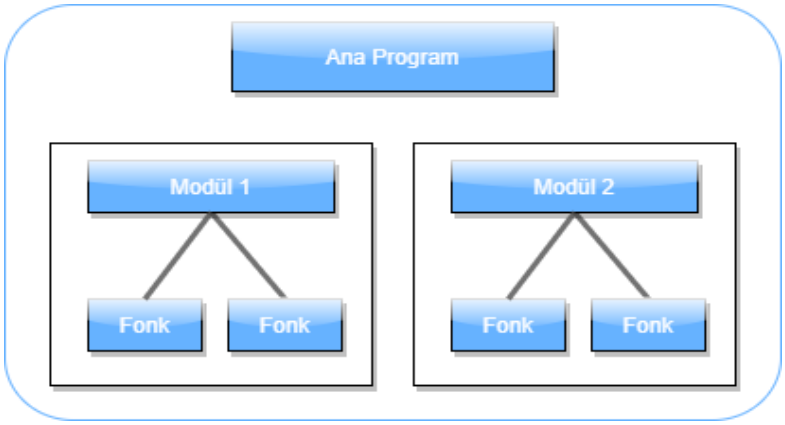
Adım	Açıklama
Başla.	Programın Başlangıcı
Sayac=0, ToplamYas=0	Burada bize iki adet değişken gerekli. Kaç kişinin bilgisi girildiğini Sayac isimli değişkende tutuyoruz. İstediğimiz sayıların toplamını ise ToplamYas isimli değişkende tutuyoruz.
Sayac<5 Olduğu Sürece	Buradaki koşulumuz 5 değerine ulaşıp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

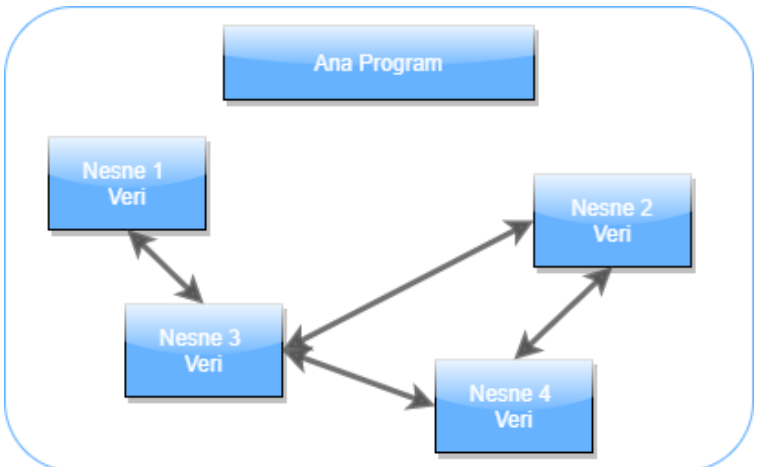
Eğitmene Öneriler: Eğitmen açıklama sırasında değişkenler ve değerlerine vurgu yapmalıdır. Öğrencilerin grup tablolarını word üzerinde oluşturup ödev olarak ÖYS'de paylaşmaları istenebilir.

Hafta 1. Yüz Yüze: Ders Materyalleri

L1. B2.1. Bilgi Kartları







L1. B3.1. 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 iki ifadenin büyüklük küçüklük ya da eşitlik açısından değerlendirilmesi için kullanılan sembollerdir.*	*Programlama dilinde farklı durumların birlikte değerlendirilmesinde kullanılan semboller mantıksal işleçler olarak adlandırılır.*

L1.B3.2. İşleç Bilgi Kartları

Aritmetik İşlemler

Arda'nın yaşı a=15 ve Duru'nun yaşı d=4'tür. Arda ve Duru'nun yaşları arasında tanımlanabilecek aritmetik işlemleri inceleyiniz.

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

Karşılaştırma İşlemleri

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

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

Mantıksal İşlemleri

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

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

L1.B3.3. Örnek Olay Sunum Kartı

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

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

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

1. Koşul 2. Koşul

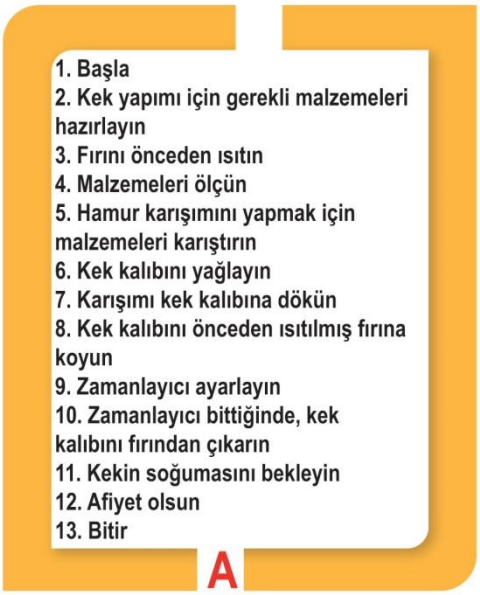
L1.B4.1. Yer Değiştirme Görev Kartları (Ön Yüz)

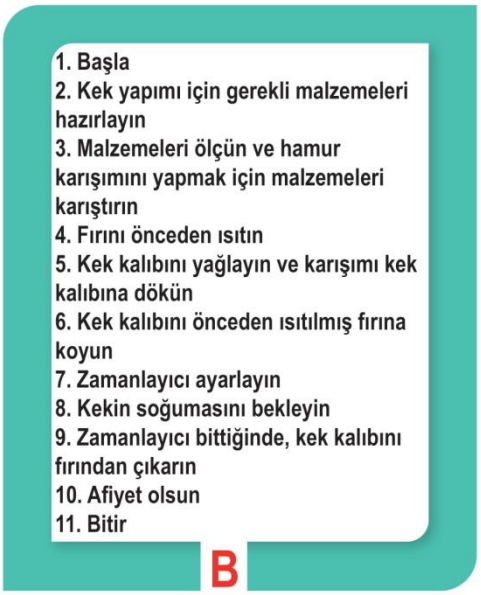
*Görev kartları arkalı önlü olacak şekilde basılmalıdır. Görev kartlarındaki her satır bir görevi oluşturmaktadır.

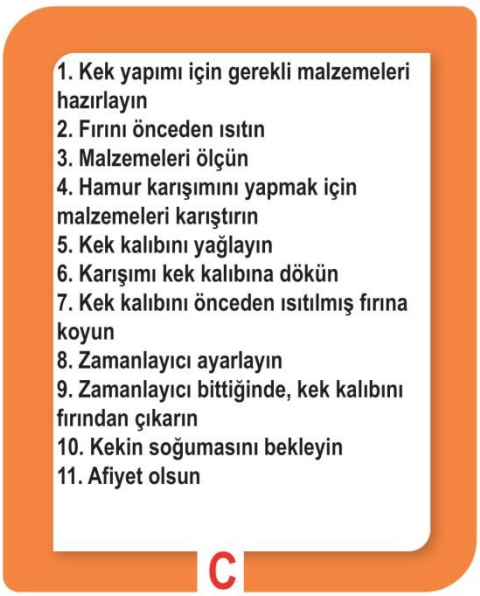
L1.B4.1. Yer Değiştirme Görev Kartları (Arka Yüz)

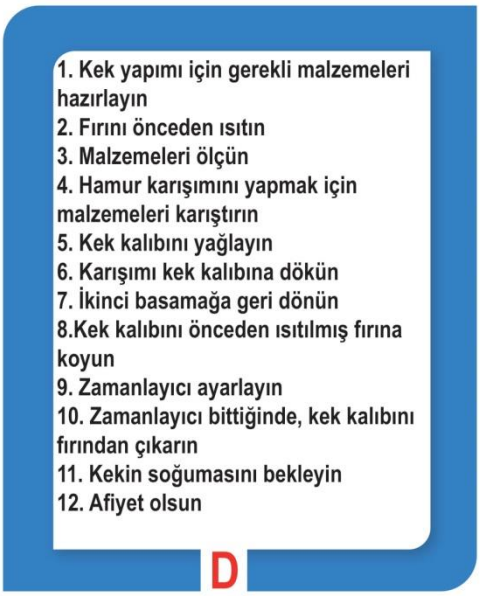
*Görev kartları arkalı önlü olacak şekilde basılmalıdır. Görev kartlarındaki her satır bir görevi oluşturmaktadır.

L1.B5.1. Doğru Algoritma Hangisi?



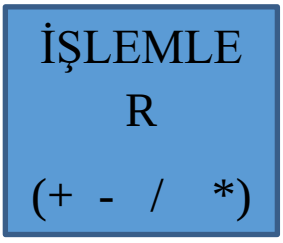






L1.B5.2. Akış Diyagramları Bilgi Afişi

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

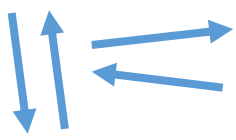


Toplama, çıkarma, aritmetik işlemlerin yapılması gerektiği durumlarda dikdörtgen kullanılır. Örneğin; üçgenin alanını hesaplama

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.

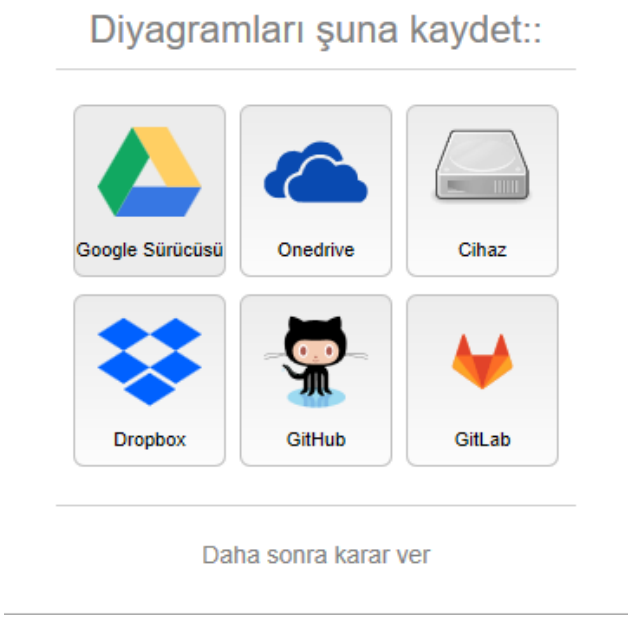
Algoritma akışında işlemler tek yönlü iken karar yapılarında iki farklı ok çıkabilir ve bazı işlemlerin tekrarlanmasını sağlayabilir. Bu şekilde bir "döngü" oluşur.



Yandaki oklar algoritmanın akış yönünü bize belirtir.

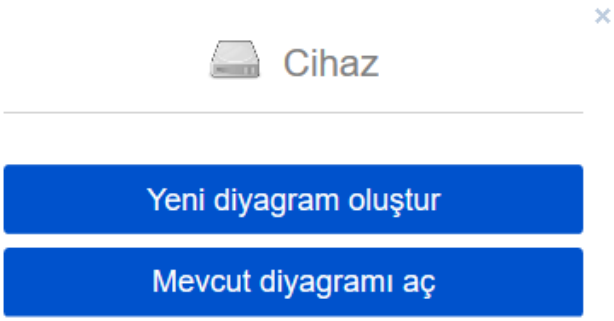
L1.B5.3 App.diagrams.net Kullanım Kılavuzu

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



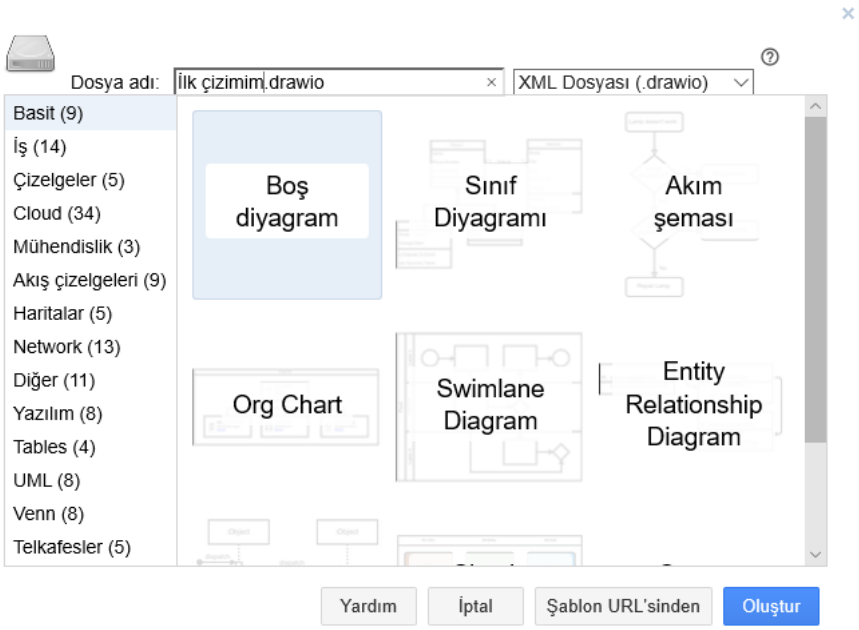
Resim 6. *App.diagrams.net giriş ekranı*

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



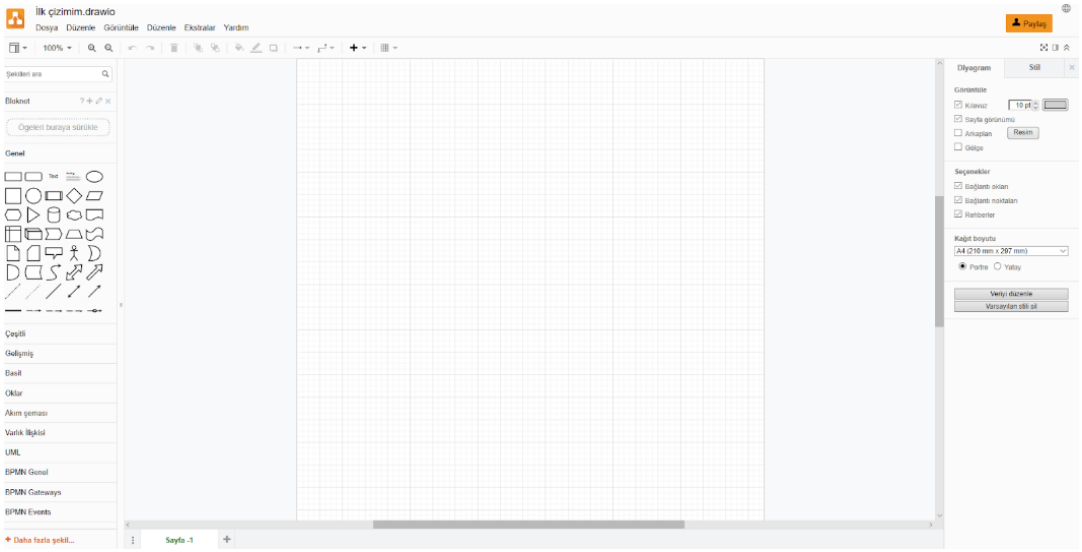
Resim 7. *App.diagrams.net yeni diyagram oluşturma ekranı*

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



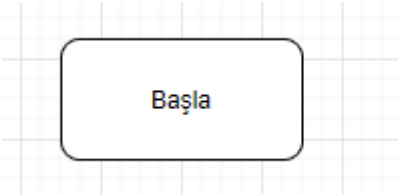
Resim 8. *App.diagrams.net yeni diyagram tipi seçme ekranı*

Karşımıza aşağıdaki çizim ekranı gelecektir.



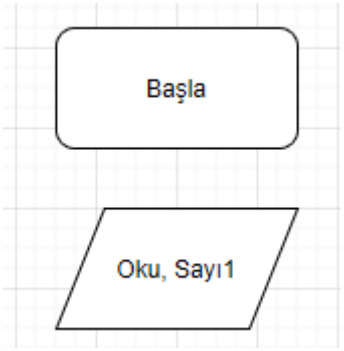
Resim 9. *App.diagrams.net çizim ekranı*

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.



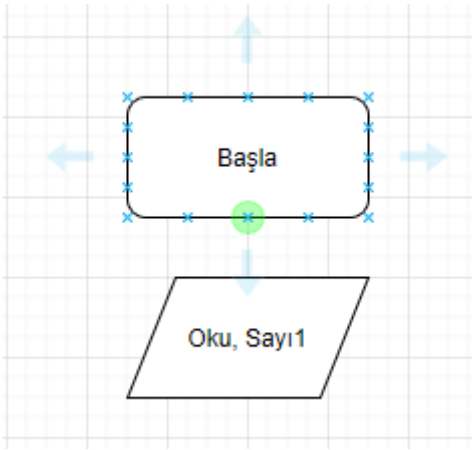
Resim 10. *Genel bloğu*

Ardından bir tane de paralel kenar ekleyelim.



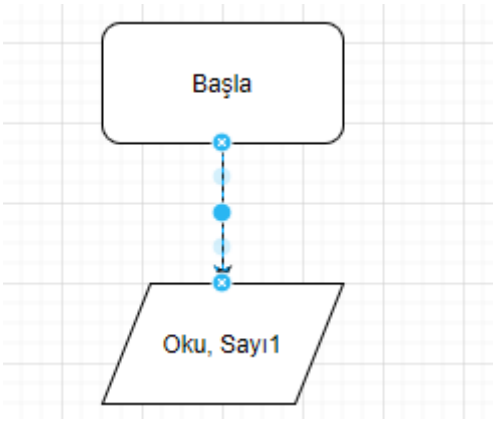
Resim 11. *Paralel kenar bloğu*

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



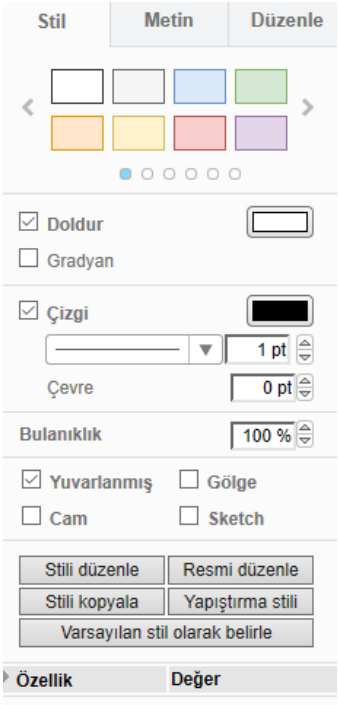
Resim 12. *Blokları bağlama öncesi*

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



Resim 13. *Blokları bağlama sonrası*

Eğer bloğun renklerini değiştirmek istersek sağ paneli kullanabiliriz.

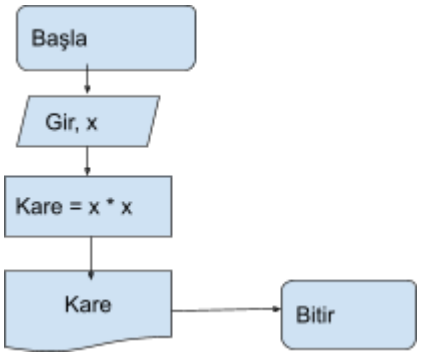
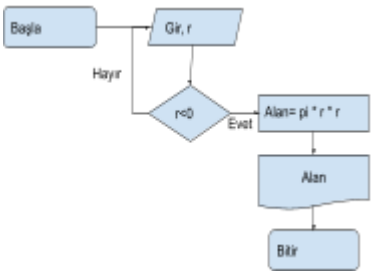


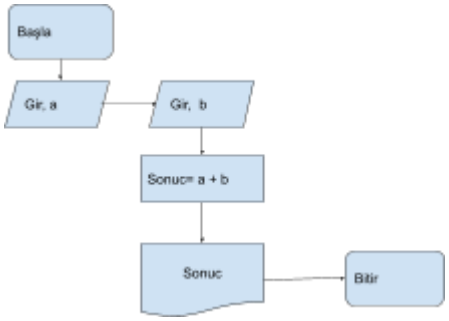
Resim 14. *Blok özellikleri ekranı*

L1.B6.1. Algoritmaları Eşleştirelim Çalışma Kâğıdı

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

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

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

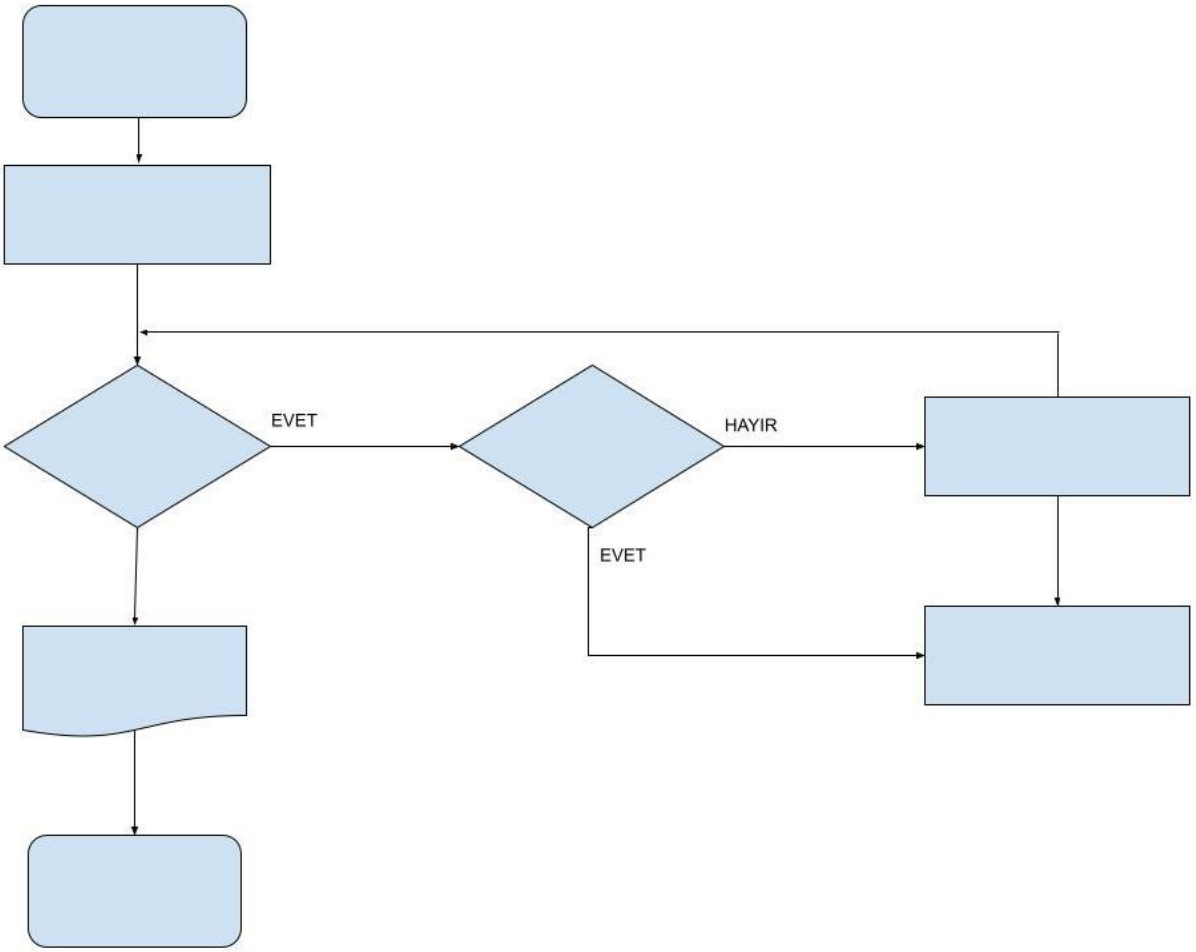
1. Başla. 2. Gir, r 3. Eğer r < 0 ikinci adıma git. 4. alan= π ×r×r 5. Yaz, Sonuc 6. Bitir		1. Başla 2. Gir, x 3. Kare = x*x 4. Yaz, Kare 5. Bitir
Problem: *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*	*Problem:* *Algoritma İfadesi:*

L1.B7.1. Çiftleri Toplama Algoritması

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

Problem: 1 ile 100 arasındaki (100 dahil değil) çift sayıların toplamını yazdırma

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



L1.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 13. *Örnek kod çalıştırma tablosu*

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

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

L1.B9.1. Kod Çalıştırma Tablosu

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

Tablo 14. *Örnek kod çalıştırma 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. Giriş: 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

LÇ1.B1.1. Grup Görevleri

Bknz. L1.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 LÇ1.B1.1. materyalindeki örnek görev ve grup görevleri sütunlar halinde öğrenciye görev sunumu için yüklenecektir.

Uygulama: Eğitmen LÇ1.B1.1 hazırladığı padlet materyalini Eğitmen app.diagrams.net arayüzünü öğrencilere açıklar ve programı gösterir. LÇ1.B1.1. de yer alan padlette ilk sütundaki örnek görevi ve akış diyagramını eğitmen açıklayarak app.diagrams.net de çizerek gösterir. Eğitmen çizim öncesi bu görevi öğrencilerle birlikte tartışır. Probleme göre hangi kod satırının akışı bozduğunu farketmeleri için öğrencilere zaman tanınır. Sohbet aracılığıyla fikir toplanır. Ardından 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 derse katılım durumu ve ödevi zamanında teslim edip etmediği dikkate alınarak puanlama yapılacaktır.

- Öğrenciye başlangıçta bir temel not (0-100 arasında) verilmektedir.
- Eğer öğrenci **derse katıldıysa**, nota **10 puan** eklenir.
- Not **60'tan büyük veya eşitse**, "Başarılı" yazdırılır. Aksi halde "Başarısız" yazdırılır.

```
Başla
    Notu gir
    Derse katıldı mı? (Evet/Hayır)
        Evet → Not = Not – 10 (Akışı bozan kod satırı!)
    Not >= 60
        Evet → Yaz: "Başarılı"
        Hayır → Yaz: "Başarısız"
Bitir
```

Örnek görevin birlikte incelenmesinin ardından, benzer şekilde düzenlenmiş 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

LÇ1.B1.1. Grup Görevleri

Bknz. L1.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: LÇ1.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 LÇ1.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: LÇ1.B4.1. C++ Program Tanıtımı *(Ek Materyaller Dosyasından Erişebilirsiniz)*

LÇ1.B4.2. Derleyici Kurulum Kılavuzu *(Ek Materyaller Dosyasından Erişebilirsiniz)*

Uygulama: Eğitmen C++ programlamanın tarihi, derleyici ihtiyacı ve seçimi ile ilgili bilgi edinebilecekleri LÇ1.B4.1 sunusunu kullanarak Code::Blocks isimli derleyici kurulumu öncesi bilgi verir. Ardından Bu sunumun sonunda yer alan LÇ1.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 << "Hosgeldiniz!" << std::endl;  // Ekrana Hosgeldiniz!
yazdir
return 0;  // Programin basarili sekilde bittigini belirt
}
```

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

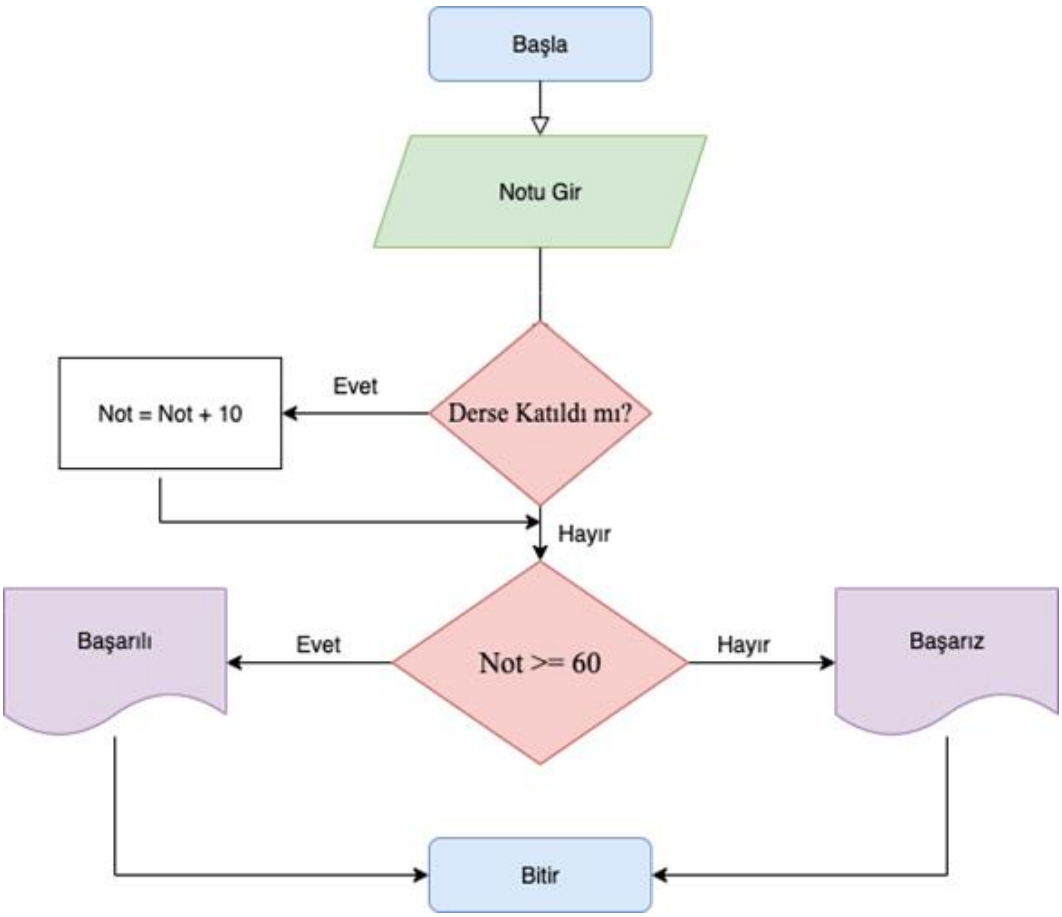
Hafta 1. Çevrim İçi: Ders Materyalleri

LÇ1.B1.1. Grup Görevleri

Örnek görev: Bir öğrencinin derse katılım durumu ve ödevi zamanında teslim edip etmediği dikkate alınarak puanlama yapılacaktır.

- Öğrenciye başlangıçta bir temel not (0-100 arasında) verilmektedir.
- Eğer öğrenci **derse katıldıysa**, nota **10 puan** eklenir.
- Not **60'tan büyük veya eşitse**, "Başarılı" yazdırılır. Aksi halde "Başarısız" yazdırılır.

```
Başla
   Notu gir
   Derse katıldı mı? (Evet/Hayır)
      Evet → Not = Not + 10
   Not >= 60
      Evet → Yaz: "Başarılı"
      Hayır → Yaz: "Başarısız"
Bitir
```



Grup 1: Okulun sosyal sorumluluk kulübü tarafından kitap bağış kampanyası düzenlenmiştir. Öğrenciler sisteme bağışladıkları kitap sayısını girer. Eğer öğrenci aynı zamanda kitap kulübü gönüllüsüyse, fazladan 5 kitap bağışladığı kabul edilir. Toplam bağış sayısı 20 ve üzerindeyse öğrenciye "Teşekkür Belgesi", değilse "Teşvik Ödülü" verili. Buna göre ilgili algoritma:

```
Başla
Kitap sayısını oku → Kitap
Gönüllü mü? (Evet/Hayır) → Gönüllü
Eğer Gönüllü ise → Kitap = Kitap – 5 (Hatalı! Toplama olması gerekir)
Eğer Kitap >= 20 ise
      → Yaz: "Teşekkür Belgesi"
      Aksi halde → Yaz: "Teşvik Ödülü"
Bitir
```

Grup 2: Okul yönetmeliğine göre bir dönemde 90 gün ders yapılır ve öğrenciler en fazla 30 gün devamsızlık yapabilir. Sistem öğrencinin devamsızlık yaptığı gün sayısını alır ve kalan günleri hesaplar. Kalan gün 60'tan azsa öğrenciye "Devamsızlık Sınırına Yaklaştı" uyarısı verilir, aksi halde "Devamsızlık Uygun" yazılır. Buna göre ilgili algoritma:

```
Başla
Devamsızlık gün sayısını oku → Devamsızlık
KalanGün = 90 - Devamsızlık
Eğer KalanGün = 60 ise, (Hatalı! < işaret olmalı)
        Evet→: "Devamsızlık Sınırına Yaklaştı"
        Hayır → Yaz: "Devamsızlık Uygun"
Bitir
```

Grup 3: Bir lise, yazılı sınavlara hazırlık amacıyla öğrencileri çalışma gruplarına ayırmaktadır. Her öğrencinin bir sınav numarası vardır. Bu numaraya göre öğrenciler şu şekilde gruplandırılır:

- Eğer numara 2 ile tam bölünüyorsa, öğrenci Grup A'ya,
- Eğer 3 ile tam bölünüyorsa, öğrenci Grup B'ye,
- Her iki koşul sağlanmıyorsa, öğrenci Grup C'ye yerleştirilir.

Buna göre ilgilli algoritma:

```
Başla
Öğrencinin sınav numarasını gir
Eğer numara % 2 == 0 → "Grup A"
Değilse, numara % 3 == 2 → "Grup B" (Hatalı! 3'bölümde kalan 0 olmalı)
Aksi halde → "Grup C"
Bitir
```

Grup 4: Okulda her öğrencinin okul numarasına göre kulübe yerleştirme yapılır. Eğer numara 3 ile tam bölünüyorsa öğrenci Matematik Kulübü'ne, bölünemiyorsa Sanat Kulübü'ne yerleştirilir.

```
Başla
Öğrenci numarasını oku → Numara
Eğer Numara % 5 == 0 ise, (Hatalı! Bölündüğü sayı değişmeli)
      → Yaz: "Matematik Kulübü"
      Aksi halde → Yaz: "Sanat Kulübü"
Bitir
```

LÇ1.B3.1. Anket Sorusu:

Bir marangoz, elindeki tahta parçalarını kullanarak üçgen şeklinde bir raf yapmaya karar verir. Ancak her üç tahta parçası bir üçgen oluşturmaz. Bu nedenle, tahta parçalarını kesmeden önce, uzunluklarının bir üçgen oluşturup oluşturmadığını kontrol etmek için bir algoritma geliştirmiştir.

Bu marangoz üçgen yerine **dikdörtgen** şeklinde bir raf yapmak isteseydi, yukarıdaki akış diyagramında hangi değişiklikler yapılırdı? Birden fazla seçenek işaretleyebilirsiniz.

Seçenekler:

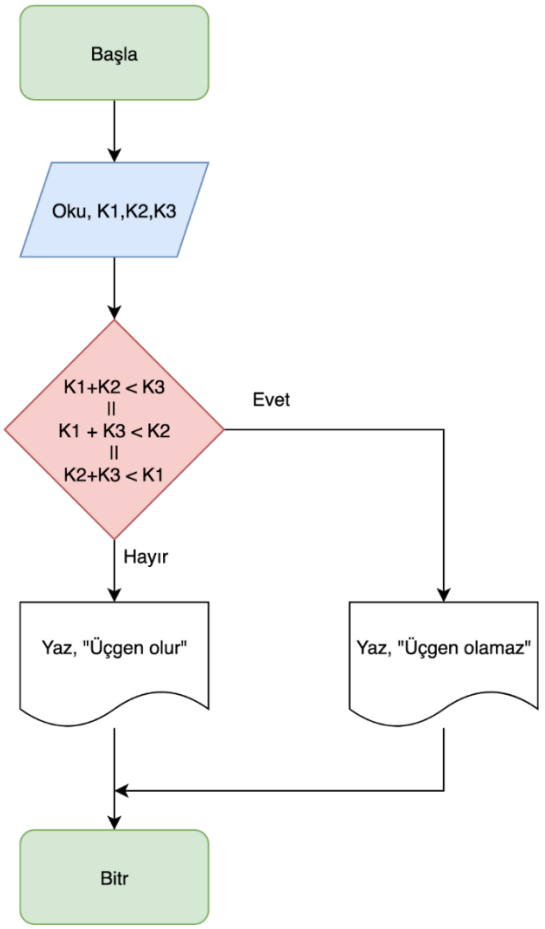
A) "Oku, K1, K2, K3" komutundaki girdi sayısı (3 yerine 2) olarak değişirdi.

B) Karar kutusundaki şart, "K1+K2 < K3 || K1+K3 < K2 || K2+K3 < K1" yerine "K1>0 ve K2>0" gibi bir koşul ile değiştirilirdi.

C) "Yaz" komutlarındaki çıktılar "Üçgen olur" ve "Üçgen olamaz" yerine "Dikdörtgen olur" ve "Dikdörtgen olamaz" olarak değiştirilirdi.

D) Akış diyagramının genel yapısı (Başla, Oku, Karar, Yaz, Bitir) aynı kalırdı, ancak içerisindeki komutlar ve koşullar yeni duruma göre güncellenirdi.

E) Herhangi bir değişiklik yapmaya gerek kalmazdı, mevcut akış diyagramı dikdörtgen için de çalışırdı.



A, B, C ve D seçenekleri doğrudur. İşte her bir seçeneğin neden doğru olduğunun açıklaması:

- **A) "Oku, K1, K2, K3" komutundaki girdi sayısı (3 yerine 2) olarak değişirdi:** Bir üçgenin üç kenarı varken, bir dikdörtgenin sadece iki farklı kenar uzunluğu (uzun kenar ve kısa kenar) vardır. Bu nedenle, dikdörtgen bir raf yapmak için sadece iki girdi (örneğin, K1 ve K2) okunması yeterli olurdu.
- **B) Karar kutusundaki şart, "K1+K2 < K3 || K1+K3 < K2 || K2+K3 < K1" yerine "K1>0 ve K2>0" gibi bir koşul ile değiştirilirdi:** Akış diyagramındaki orijinal koşul, üçgenin var olma koşuludur (üçgen eşitsizliği). Dikdörtgen için böyle bir koşula ihtiyaç yoktur. Sadece girilen kenar uzunluklarının pozitif sayılar olması yeterlidir. Bu nedenle, karar kutusundaki koşul, K1>0 ve K2>0 gibi basit bir kontrolle değiştirilirdi.
- **C) "Yaz" komutlarındaki çıktılar "Üçgen olur" ve "Üçgen olamaz" yerine "Dikdörtgen olur" ve "Dikdörtgen olamaz" olarak değiştirilirdi:** Programın amacı artık bir üçgen değil, bir dikdörtgenin mümkün olup olmadığını belirlemektir. Bu nedenle, çıktılar da amaca uygun olarak güncellenmelidir.
- **D) Akış diyagramının genel yapısı (Başla, Oku, Karar, Yaz, Bitir) aynı kalırdı, ancak içerisindeki komutlar ve koşullar yeni duruma göre güncellenirdi:** Bu seçenek, diğer doğru cevapları özetleyen genel bir ifadedir. Bir programın temel mantığı aynı kalır: girdi al, karar ver, çıktı ver ve bitir. Sadece içerisindeki spesifik detaylar (girdiler, koşullar ve çıktılar) değişir.
- **E)** E seçeneği ise yanlıştır, çünkü üçgen ve dikdörtgenin matematiksel özellikleri birbirinden tamamen farklıdır ve aynı akış diyagramı her ikisi için de kullanılamaz.

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. Veri tipi dönüşüm işlemlerini yapar.
- K5. C++ programlama dilinde sabitleri tanımlamayı bilir.
- K6. Kaçış dizgelerini kullanır.
- K7. C++ temel giriş/çıkış akışlarını bilir.
- K8. C++ işleçlerinin kullanımını bilir.
- K9. Bit işlemleri ile veriler üzerinde hesaplamalar yapar.

Amaç

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

Önerilen Ders Akışı

A. Giriş: Ev Sahibi ve Kiracı (15 dk.)

B. Öğrenme Görevleri

B1. İlk C++ Programım (35 dk.)

Ders Arası (10 dk.)

B2. Değişkenlere İsim Verelim (20 dk.)

B3. Veri Tiplerini Öğrenelim (30 dk.)

Ders Arası (10 dk.)

B4. Veri Tipini Dönüştürelim (20 dk.)

B5. Sabitleri Ayıralım (15 dk.)

B6. Kaçış Dizgelerini Tanıyalım (15 dk.)

Ders Arası (10 dk.)

B7. Çıktıları Karşılaştıralım (10 dk.)

B8. Operatörlerle Yüzleşelim (20 dk.)

B9. Listeyi Dolduralım (20 dk.)

A. Giriş: Ev Sahibi ve Kiracı

Süre: 10 dk.

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

B. Öğrenme Görevleri

B1. İlk C++ Programım

Süre: 40 dk.

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

Materyaller: L2.B1.1. IDE Üzerinde Kod Yazımı

Hazırlık: Eğitmen IDE programını ve L2.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 L2.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: 20 dk.

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

Materyaller: L2.B2.1. Hatalı Değişkenler

L2.B2.2. Değişken İsmi Olamazlar

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

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

"C++ programlarında 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 15. *Saklayıcı yapı kutu gösterimleri*

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;
```

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

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

talimatı verilir. Her grup bir tane değişken belirleme kuralına erişmelidir. Bu kurala erişmek için eğitmen her öğrencinin kartında yazan değişken isimlerini grup üyelerininki ile karşılaştırmalarını ister. Bu şekilde kartlar arası ortak noktaların değişken belirleme kuralına işaret edeceği ipucunu verir. Gruplar beş dakika kartlar üzerinde çalışırlar. Eğitmen sırayla grupların belirledikleri kuralı sesli şekilde okutur.

Eğitmen tüm kurallar tamamlanınca değişken ismi olmayan yasaklı kelimelerin olduğu konusunda bir hatırlatma yapar. Bu kelimeler "Değişken İsmi Olamazlar" materyali ile Öğrenme Yönetim Sistemi üzerinden öğrencilerle paylaşılır. Bununla birlikte eğitmen aşağıdaki açıklamayı kullanarak öğrencilere uyarıda bulunur.

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

Etkinlik sonunda eğitmen değişkenlere değer atama türlerinden kısaca bahseder.

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

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

yazdıktan sonra çıktıları karşılaştırır.

İlk değer atamasız değişken tanımlama

```
#include <iostream>
int main()
{
int sayi;
sayi = 5;
std::cout << sayi;
return 0;
}
```

İlk değer atamalı değişken tanımlama

```
#include <iostream>
int main()
{
    int sayi = 5;
    std::cout << sayi;
    return 0;
}
```

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

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

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

İPUCU: *C++ programlamada, değişkenlerin adları için küçük harfler kullanmak standart bir uygulamadır. Bazı değişkenlerin adları belirli bir kullanımla ilişkilendirilme eğilimindedir. Örnek olarak;*

c, ch: karakterler için kullanılır.

i, j, k, l, m, n: tam sayılar ve indisler için kullanılır.

x, y, z: tam ve virgüllü sayılar için kullanılır.

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

B3. Veri Tiplerini Öğrenelim

Süre: 30 dk.

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

Materyaller: L2.B3.1. Veri Tipleri Çalışma Kâğıtları

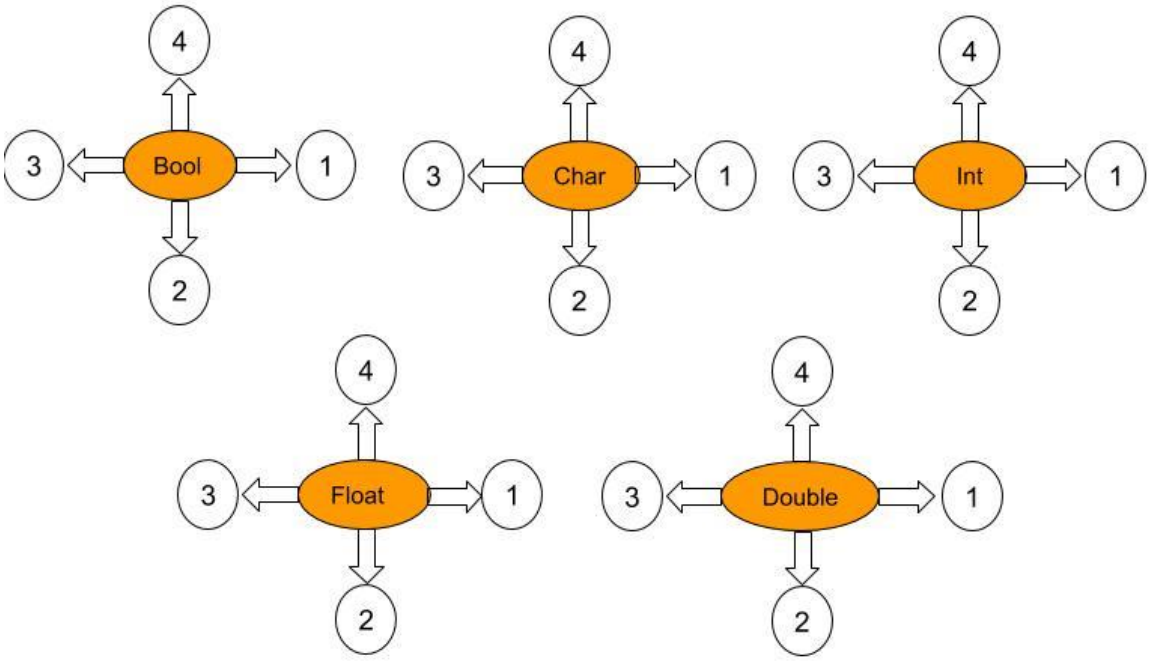
L2.B3.2. Değişken Atama Kodları

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

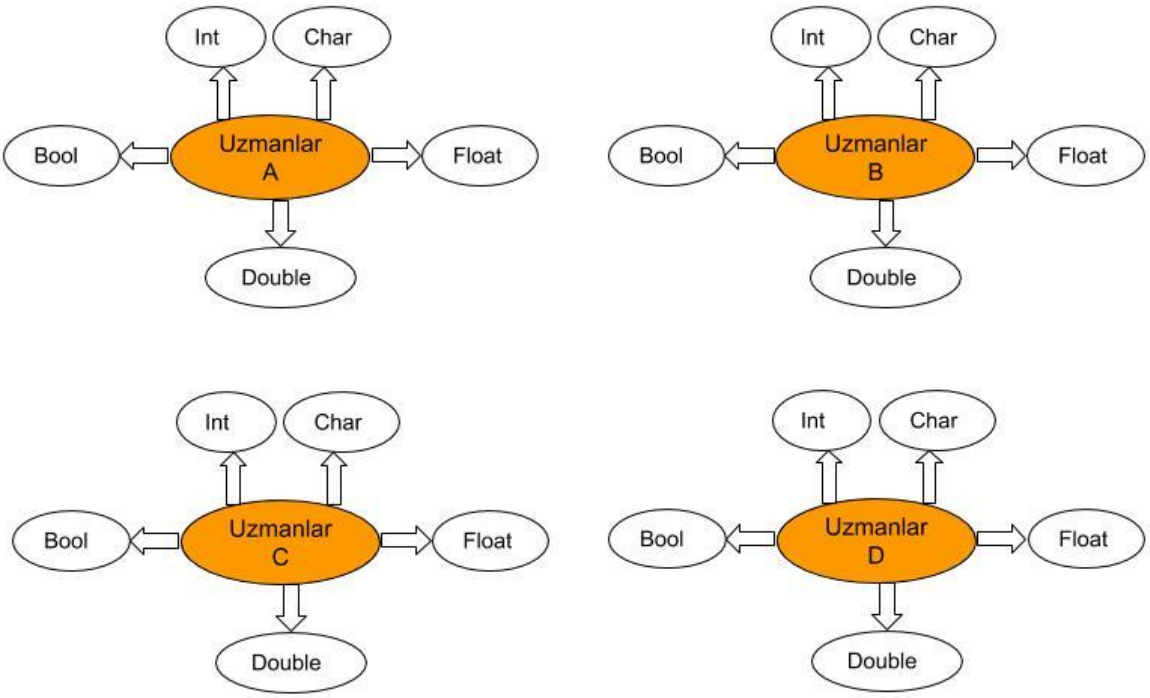
Uygulama: Öğrenciler dörderli beş gruba ayrılır. Eğitmen gruplara beş parçaya ayırdığı materyalleri dağıtır. Bu materyal ile her bir grup bir veri tipini temsil etmektedir. Eğitmen gruplardan grup üyelerinin her birinin veri tipi adi_no şeklinde kodlanmalarını ister. Örneğin Bool ekip üyeleri, Bool_1, Bool_2…. şeklinde dört kişiden oluşmaktadır. Eğitmen öğrencilerden temsil ettikleri veri tipinin özellikleri hakkında çalışma kâğıtlarını incelemeleri ve örnekleri grup içinde beş dakika içinde tartışmalarını ister. Eğitmen materyalin yanı sıra, internet üzerinden de araştırma yapabileceklerini de belirtebilir. Bu tartışmada her bir grup üyesinin temsil ettiği veri tipini başkasına anlatacak seviyede uzman olmaları istenir. Beş dakika ekipler kendi içlerinde çalıştıktan sonra, aynı numara ile kodlanmış öğrenciler bir araya gelir. Örneğin 1 numaralı grup üyeleri birleşerek beş kişiden oluşan uzmanlar grubu elde edilir.

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

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



Oturma Düzeni 1



Oturma Düzeni 2

Resim 16. *Oturma düzenleri*

Öğrenciler uzman gruplarına dağılmadan önce öğrencilerin grup içinde birlikte anlamlandıramadıkları noktaları eğitmenlere danışmaları istenebilir. Ayrıca eğitmen uzman gruplarının her birinin birer öğretmen olarak rol aldıkları konusunda öğrencilere hatırlatıcı ve teşvik edici geri bildirimlerde bulunmalıdır. Örneğin "Bu işi çok iyi başardın; Konuyu diğer

arkadaşlarına senin öğretiyor olman, benim açıklama yapmamdan çok daha etkili olacaktır" vb. gibi.

B4. Veri Tipini Dönüştürelim

Süre: 20 dk.

Kazanımlar: K4. Veri tipi dönüşüm işlemlerini yapar.

Materyaller: L2.B4.1. Veri Tipi Dönüşümü

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

Uygulama: Öğrenciler B2 etkinliğindeki uzman grupları oturma düzeninde dört grup hâlindedir. Eğitmen bu etkinlikte öğrencilere materyali dağıtır. Kod üzerindeki veri tiplerine ilişkin çıktıyı uzman grubu içerisinde beş dk süresince tartışmaları istenir. Bu materyal ile öğrencilerden kod içerisindeki değişkenler ve veri tiplerinin hangileri olduğu konusunda düşünmeleri beklenir. Eğitmen öğrencilere temel problemlerini tasarlamalarını ve kod üzerinde ilgili değişken isimlerini değiştirmelerini söyler. Bu aşamadan sonra kod çıktısını tahmin etmeleri ve kodu çalıştırıp tahminlerini çıktı ile karşılaştırmaları istenir. Kodun çıktısı aşağıdaki gibidir:

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

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

Çıktı:

a = 70

b = A

c = 73

d = 3.4

e = 5

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

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

degisken_adi = *(veri-tipi) degisken_adi*;

degisken_adi = static_dönüsüm <veri-tipi> degisken_adi;

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

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

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

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

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

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

Kodun Çıktısı:

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

B5. Sabitleri Ayıralım

Süre: 15 dk.

Kazanımlar: K5. C++ programlama dilinde sabitleri tanımlamayı bilir.

Materyaller: L2.B5.1. Sabitler İçin Örnek Kod

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

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

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

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

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

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

```
const veriTipi SABITADI = deger;

const double PI = 3.14159265;
```

Programlama sırasında sabitleri (const) kullanmak, hem kodun okunabilirliğini hem de bakımını kolaylaştıran önemli bir uygulamadır. Sabitler, değerleri program boyunca değiştirilmeyen değişken benzeri yapılardır ve bu özellikleri sayesinde çeşitli avantajlar sunar.

Örneğin, matematikte sıkça kullanılan Pi sayısını ele alalım. Eğer bir programda Pi sayısına ihtiyaç duyduğumuz her yerde doğrudan 3.14 değerini yazarsak, bu değer kodun birçok noktasında tekrar edilmiş olur. Diyelim ki daha sonra uygulamanızın daha hassas hesaplamalar yapması gerekti ve Pi'nin daha kesin bir değeri olan 3.14159265 kullanılmak istendi. Bu durumda, kodun her satırında geçen 3.14 ifadesini manuel olarak yeni değerle değiştirmek gerekir. Bu işlem, sadece zaman kaybettirici olmakla kalmaz, aynı zamanda bazı yerleri atlama veya yanlış değiştirme gibi hatalara da yol açabilir.

B6. Kaçış Dizgelerini Tanıyalım

Süre: 15 dk.

Kazanımlar: K6. Kaçış dizgelerini kullanır.

Materyaller: L2.B6.1. Kaçış Dizgeleri Örnek Kod

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

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

Eğitmene Öneriler: Etkinlik sonunda erişilecek liste aşağıdaki gibidir:

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

B7. Çıktıları Karşılaştıralım

Süre: 10 dk.

Kazanımlar: K7. C++ temel giriş/çıkış akışlarını bilir.

Materyaller: L2.B7.1. 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 dili, kullanıcıyla etkileşim kurmak amacıyla giriş (input) ve çıkış (output) işlemlerini gerçekleştirebilmek için çeşitli kütüphaneler ve yapılar sunar. Bu işlemler, verilerin bilgisayar sistemine alınması veya sistemden dış ortamlara (örneğin ekrana ya da dosyaya) aktarılması anlamına gelir. C++ dilinde giriş-çıkış işlemleri genellikle akış (stream) kavramı çerçevesinde ele alınır. Akışlar, verilerin bayt dizileri şeklinde bir kaynaktan bir hedefe doğru hareketini temsil eder.

Giriş akışı (input stream), verilerin bir dış kaynaktan (örneğin klavye, dosya, disk, ağ gibi aygıtlardan) bilgisayarın ana belleğine aktarılmasını ifade eder. Buna karşılık, çıkış akışı (output stream), bellekteki verilerin bir hedefe (ekran, yazıcı, dosya vb.) gönderilmesini sağlar. Giriş ve çıkış işlemlerinin temel amacı, programın veri almasını ve bu verileri dış ortamlara anlamlı bir şekilde sunmasını sağlamaktır.

C++ dilinde bu işlemleri gerçekleştirmek için kullanılan bazı önemli başlık dosyaları (header files) vardır:

***<iostream>:** Standart giriş ve çıkış işlemlerini gerçekleştirmek için kullanılır. Bu başlık dosyası, veri girişinde kullanılan cin, veri çıkışında kullanılan cout, hata mesajlarında kullanılan cerr gibi standart akış nesnelerini içerir. Programda kullanıcıdan veri almak ya da ekrana bilgi yazdırmak istiyorsak bu dosyanın dahil edilmesi zorunludur.*

***<iomanip>:** Giriş ve çıkış işlemleri sırasında biçimlendirme yapmak için kullanılır. Örneğin sayıları belirli bir genişlikte hizalama (setw), ondalık hassasiyet ayarlama (setprecision) gibi işlemler bu başlık dosyasındaki işlevlerle gerçekleştirilir. Daha okunabilir ve profesyonel görünümlü çıktılar üretmek için sıklıkla kullanılır.*

***<fstream>:** Dosya işlemlerinde kullanılır. Bu başlık dosyası sayesinde, dosyalardan veri okuyabilir (ifstream), dosyalara veri yazabilir (ofstream) ya da her iki işlemi birden yapabilirsiniz (fstream). Programda dosya tabanlı veri saklama ve okuma gibi işlemler gerekiyorsa bu başlık dosyasına ihtiyaç duyulur.*

C++ dilinde veri almak ve ekrana yazdırmak için en yaygın kullanılan iki yapı cin ve cout anahtar kelimeleridir. cin, kullanıcıdan veri almak için kullanılırken; cout, ekrana bilgi yazdırmak için kullanılır. Bu ifadeler, iostream başlık dosyasındaki istream ve ostream sınıflarına dayanır. Dolayısıyla, bu işlemleri kullanabilmek için programın başına #include <iostream> ifadesi mutlaka eklenmelidir.

B8. Operatörlerle Yüzleşelim

Süre: 20 dk.

Kazanımlar: K8. C++ işleçlerinin kullanımını bilir.

Materyaller: L2.B8.1. Operatör Çalışma Yaprağı

L2.B8.2. Destekleyici Bilgiler

Hazırlık: Her bilgisayarda Code::Blocks kayıtlı ve aktif çalışır durumda olmalıdır. Çalışma yapraklarından 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.

B9. Listeyi Dolduralım

Süre: 20 dk.

Kazanımlar: K9. Bit işlemleri ile veriler üzerinde hesaplamalar yapar.

Materyaller: L2.B9.1. Keşfetme Kartları

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

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

Tablo 15. *Boş operatör görev tablosu*

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

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

Eğitmene Öneriler: Etkinlik sonunda erişilecek liste aşağıdaki gibidir:

Tablo 16. *Operatör görev tablosu*

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

Hafta 2. Yüz Yüze: Ders Materyalleri

L2.B1.1. IDE Üzerinde Kod Yazımı

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

int main() {
    cout << "Merhaba Dunya!" << 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 Karaca, 14 yasindayim." << endl;
   return 0;
}
```

Öğrenmeyi Derinleştirme Soruları:

- using namespace std; satırını silersek ne olur?
- return 0; ifadesini yazmazsak ne olur?
- cout ve endl hangi namespace'e aittir?

L2.B2.1. Hatalı Değişkenler

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

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

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

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

L2.B2.2. Değişken İsmi Olamazlar

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

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

L2.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. Ondalık kısmında genel olarak 7 basamak hassasiyete sahiptir. Float veri tipi 4 bayt uzunluğundadır. Pozitif değerler için $1,2 \times 10^{-38}$ ile $3,4 \times 10^{38}$ ve negatif değerler için $-1,2 \times 10^{-38}$ ile $-3,4 \times 10^{38}$ aralığında değer alabilir.	4 Bayt	Boy (1,75), Kilo (60,5)
double	Double veri tipi 8 bayt uzunluğundadır. Pozitif değerler için $2,2 \times 10^{-308}$ ile $1,7 \times 10^{308}$ ve negatif değerler için -2.2×10^{-308} ile -1.7×10^{308} aralığında değer alabilir. Görüldüğü üzere son derece yüksek bir hassasiyete sahiptir. Ondalık kısmında genel olarak 16 basamak hassasiyete sahiptir. Virgüllü sayıların kullanımında bilimsel gösterim kullanılabilir. Aşağıda bazı sayıların bilimsel gösterimde nasıl yazıldığını görebilirsiniz: 2.2748e+21 -3.15479e+3 342.234e-14 44.78e-13	8 Bayt	Pi Sayısı (3.1415926535)

L2.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
```

L2.B4.1. Veri Tipi Dönüşümü

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

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

L2.B5.1. 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;
}
```

L2.B6.1. Kaçış Dizgeleri Örnek Kod

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

Kodun Çıktısı:

```
Bu      cümlede

        cok “fazla” kacis dizgesi vardir!
```

L2.B7.1. 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;
}
```

L2.B8.1. Operatör Çalışma Yaprağı

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

return 0;
}
```

Kodun Çıktısı:

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

Kodun Çıktısı:

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

Kodun Çıktısı:

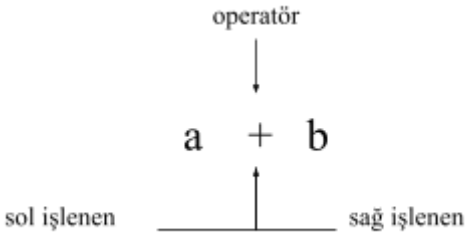
```
#include <iostream>
using namespace std;
int main()
{
    float x, y, z;
    x = 2.4;
    y = x;
    z = x + 1.3 + (y * 2.0);

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

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

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

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

Kodun Çıktısı:

L2.B8.2. 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 1 arttırmak ya da 1 azaltmak için kullanılır. “x++” ifadesinde x değişkeni var olan değeri ile işleme girer ve işlem tamamlandıktan sonra x değişkeninin değeri bir arttırılır. “++x” ifadesinde ise önce değişkenin değeri bir arttırılır ve daha sonra işlem yapılır. Aynı şekilde -- operatörü içinde benzer kullanım geçerlidir. Aşağıdaki örneği inceleyiniz.

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

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


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


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

UYARI:

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

Karşılaştırma Operatörleri

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

Tablo 18. *Önemli aritmetik operatörler*

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

UYARI:

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

Atama Operatörleri

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

Tablo 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 böler ve ardından sonucu soldaki değişkene atar.	x/=y (x=x/y)
%=	Bu işleç "%" ve "=" operatörlerinin birleşimidir. Önce soldaki değişkenin geçerli değerini sağdaki değere göre modunu alır ve ardından sonucu soldaki değişkene atar.	x%=y (x=x%y)

Mantıksal Operatörler

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

Tablo 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 "v1" 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.

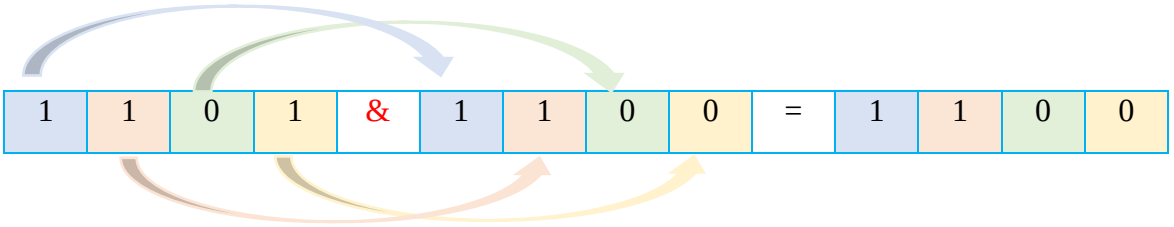
Tablo 22. *Operatör öncelikleri*

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

L2.B9.1. Keşfetme Kartları

Bit Ve (&) Operatörü



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

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

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

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

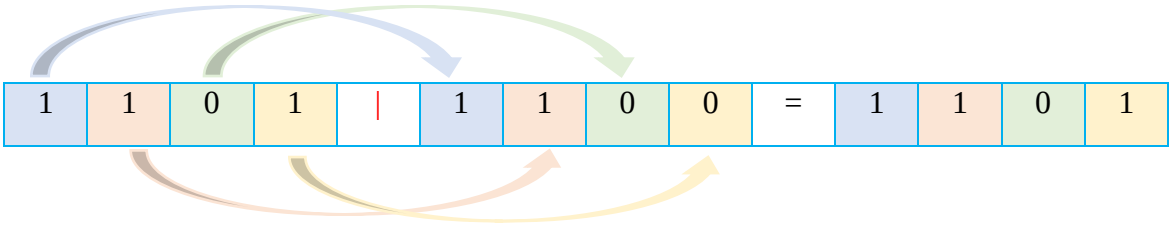
Örnek Kod:

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

Kodun Çıktısı:

```
Sonuc: 26
```

Bit Veya (|) Operatörü



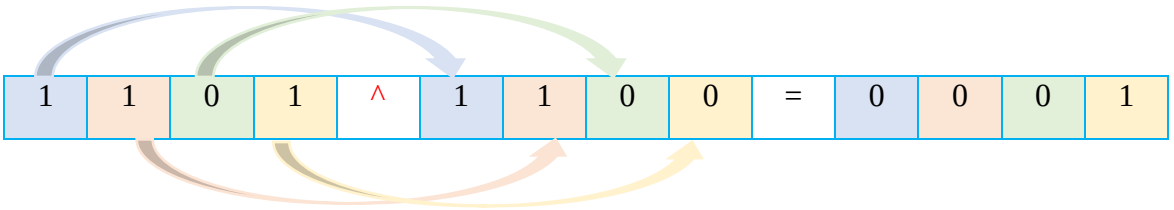
Örnek Kod:

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

Kodun Çıktısı:

```
Sonuc: 95
```

Bit Özel Veya (^) Operatörü



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

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

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

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

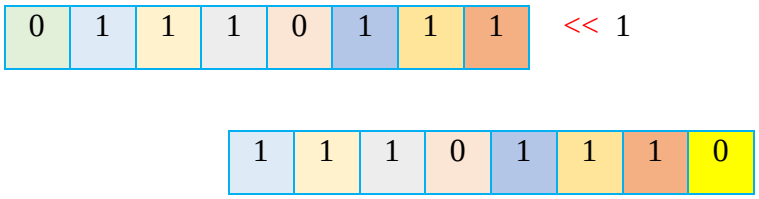
Örnek Kod:

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

Kodun Çıktısı:

```
Sonuc: 89
```

Bit Sola (<<) Öteleme Operatörü



Örnek Kod:

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

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

Kodun Çıktısı:

```
23 << 1: 46

23 << 3: 184
```

Bit Değil (~) Operatörü

~	1	0	0	1	=	0	1	1	0

~	1	1	0	0	=	0	0	1	1

~	1	0	0	0	=	0	1	1	1

~	1	1	1	1	=	0	0	0	0

Örnek Kod:

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

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

Kodun Çıktısı:

```
~23: 232

~78 : 177
```

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: LÇ2.B1.1. Öğrenci Yönerge Metni

LÇ2.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 LÇ2.B1.1 ve LÇ2.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 LÇ2.B1.1'de yer alan öğrenci yönerge metnini sunarak görevi açıklar. Görevin teknik detayı olan kod bloğunun ise (LÇ2.B1.2) öğrencilerin kendi bilgisayarlarındaki Code::Blocks programına aktarmaları gerektiğini belirtir. Etkinliğin devam eden 10 dakikalık bölümü, öğrencilerin bireysel çalışma zamanıdır. Bu sürede öğrenciler, LÇ2.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: LÇ2.B2.1. Anket Soruları

Hazırlık: Eğitmen B1 etkinliğindeki padlet ortamında ilerlemeye devam eder. Bu ortamda LÇ2. 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, LÇ2.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: 40 dk.

Materyaller: LÇ2.B3.1. Operatör Bulmacaları ve Cevapları

Hazırlık: Eğitmen, ders öncesinde bir sunum hazırlar. LÇ2.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

LÇ2.B1.1. Öğrenci Yönerge Metni

Göreviniz: Bu görevde, bir video oyunundaki envanter eşyasını temsil eden değişkenler için C++'ın isimlendirme kurallarına ve camelCase stiline uygun, anlamlı isimler oluşturacaksınız. Unutmayın, iyi yazılmış kod, sadece çalışan değil, aynı zamanda başkaları (ve gelecekteki siz!) tarafından kolayca okunup anlaşılabilen koddur. Bu, özellikle oyun geliştirme gibi büyük ve karmaşık projelerde hayati önem taşır.

Adımlar:

1. Aşağıda verilen LÇ2.B1.2 kod bloğunu kopyalayın ve Code::Blocks'ta oluşturduğunuz yeni .cpp dosyanıza yapıştırın.
2. Kod içerisindeki yorum satırları (/* ... */), sizden ne tür bir bilgi tutmanızın beklendiğini açıklıyor. Her bir değişken tanımı için (... ile belirtilen yerlere) **camelCase** standardına uygun (örneğin: oyuncuSeviyesi, esyaDayanikliligi) bir değişken ismi yazarak kodu tamamlayın.
3. Kodunuzu tamamladıktan sonra, hem metin halini (.cpp dosyası olarak) hem de ekran görüntüsünü, başlık kısmına adınızı ve soyadınızı yazarak öğretmeninizin belirttiği Padlet panosuna yükleyin.

LÇ2.B1.2. Paylaşılacak Kod Bloğu

```
#include <iostream>
#include <string>

int main() {

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

    double ... /* Esyanin altin cinsinden degeri (Orn: 299.99) */ ;
    double ... /* Esyanin kritik vurus orani (Orn: 15.5) */ ;
    int    ... /* Esyanin saldiri gucu */ ;
    int    ... /* Kullanmak icin gereken minimum seviye */ ;
    float  ... /* Esyanin agirligi (kg cinsinden) */ ;
    bool   ... /* Esyanin buyulu olup olmadigi (true/false) */ ;
    char   ... /* Esyanin nadirligi ('S'iradan, 'N'adir, 'E'fsanevi) */ ;

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

    // Bu asamada degiskenlere deger atamaniz gerekmiyor
    // Odaklanmaniz gereken tek sey, dogru ve anlamli isimlendirme yapmak

    return 0;
}
```

LÇ2.B2.1. Anket Soruları

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

Gönderi 1: (int) Tamsayı Bölmesi

İçerik:

```
#include <iostream>
int main() {
    int sonuc = 9 / 4;
    std::cout << sonuc;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) 2.25

B) 2

C) 3

D) Hata verir.

Gönderi 2: (bool) Mantıksal Operatörler

İçerik:

```
#include <iostream>
int main() {
    bool durum = (7 > 5) && (2 == 1);
    std::cout << durum;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) true

B) false

C) 1

D) 0

Gönderi 3: (double) Ondalıklı Sayı Hassasiyeti

İçerik:

```
#include <iostream>
int main() {
    double sonuc = 10.0 / 4;
    std::cout << sonuc;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) 2

B) 3

C) 2.5

D) Hata verir.

Gönderi 4: (char) Karakter Aritmetiği

İçerik:

```
#include <iostream>
int main() {
    char harf = 'B' + 3;
    std::cout << harf;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur? (ASCII tablosu temel alınmıştır)

A) B3

B) Hata verir.

C) E

D) 69

Gönderi 5: (float) Aritmetik İşlem

İçerik:

```
#include <iostream>
int main() {
    float sonuc = 5.0f / 2;
    std::cout << sonuc;
    return 0;
}
```

Soru: Bu kodun ekran çıktısı ne olur?

A) 2

B) 2.0

C) 2.5

D) Hata verir.

Cevap Anahtarı: 1-**B**, 2-**D**, 3-**C**, 4-**C**, 5-**C**

LÇ2.B3.1. Operatör Bulmacaları ve Cevapları

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

Bulmaca 1: Veri Paketi Birleştirme

Bir ağ yönlendiricisi saniyede 10 veri paketi işlerken, yoğunluktan dolayı ek olarak 5 paket daha alırsa, bir saniyede toplam kaç paket işlemiş olur?

Tahmin için Gösterilecek Kod:

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

Açıklama için Gösterilecek Kod:

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

Doğru Çıktı: 15

Bulmaca 2: Bellek Bloğu Paylaşımı

17 MB'lık bir bellek bloğunu, 5 farklı işleme (process) eşit olarak paylaştırmaya çalıştığımızda, arta kalan ve paylaştırılamayan bellek miktarı kaç MB olur?

Tahmin için Gösterilecek Kod:

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

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int toplamBellek = 17;
    int islemSayisi = 5;
    // Modulo (%) operatoru, 17'nin 5'e bolumunden kalani bulur.
    // 17 = (3 * 5) + 2
    int artaKalanBellek = toplamBellek % islemSayisi;
    std::cout << artaKalanBellek; // Cikti: 2
    return 0;
}
```

Doğru Çıktı: 2

Bulmaca 3: Güvenlik Duvarı Kuralı

Bir güvenlik duvarı (firewall), sadece 1000'den daha yüksek port numaralarından gelen isteklere izin veriyor. 8080 numaralı porttan gelen bir istek, bu kuralı geçer mi?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int portNumarasi = 8080;
    bool izinVerildiMi = (portNumarasi > 1000);
    std::cout << izinVerildiMi; // Cikti: ??? (true icin 1, false icin 0)
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int portNumarasi = 8080;
    // 8080, 1000'den buyuk oldugu icin kosul dogrudur (true).
    // C++'ta 'true' degeri ekrana 1 olarak yazdirilir.
    bool izinVerildiMi = (portNumarasi > 1000);
    std::cout << izinVerildiMi; // Cikti: 1
    return 0;
}
```

Doğru Çıktı: 1

Bulmaca 4: Veri Transferi Hızlandırma (Sola Öteleme)

Bir sistemin temel veri aktarım hızı saniyede 6 megabit (Mb) olarak ayarlanmıştır. Özel bir komutla bu hız 2 bit sola ötelenerek (yani 4 katına çıkarılarak) artırılıyor. Yeni hız ne olur?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int hiz = 6; // Ikilik tabanda: 00000110
    int yeniHiz = hiz << 2;
    std::cout << yeniHiz; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int hiz = 6; // Ikilik tabanda: 00000110
    // Sola 2 bit otelemek, sayiyi 2^2=4 ile carpmak gibidir.
    // 00000110 << 2  =>  00011000 (Bu da onluk tabanda 24'tur)
    int yeniHiz = hiz << 2;
    std::cout << yeniHiz; // Cikti: 24
    return 0;
}
```

Doğru Çıktı: 24

Bulmaca 5: Veri Sıkıştırma Simülasyonu (Sağa Öteleme)

Bir sensörden gelen 140 birimlik bir veri, basit bir sıkıştırma algoritmasıyla 1 bit sağa öteleniyor (yani tam sayı olarak 2'ye bölünüyor). Sıkıştırılmış yeni veri değeri ne olur?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int veri = 140; // Ikilik tabanda: 10001100
    int sikistirilmisVeri = veri >> 1;
    std::cout << sikistirilmisVeri; // Cikti: ???
    return 0;
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int veri = 140; // Ikilik tabanda: 10001100
    // Saga 1 bit otelemek, sayiyi 2^1=2 ile tam bolmek gibidir.
```

```
    // 10001100 >> 1  =>  01000110 (Bu da onluk tabanda 70'tir)
    int sikistirilmisVeri = veri >> 1;
    std::cout << sikistirilmisVeri; // Cikti: 70
    return 0;
}
```

Doğru Çıktı: 70

Bulmaca 6: Yetki Kontrolü (Bitsel VE)

Bir sistemde kullanıcı yetkileri bitlerle temsil ediliyor. OKUMA=4 (100), YAZMA=2 (010), ÇALIŞTIRMA=1 (001). Bir kullanıcının yetki kodu 5. Bu kullanıcının OKUMA yetkisi var mıdır?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int kullaniciYetkisi = 5; // Ikilik: 101 (Okuma ve Calistirma)
    int kontrolEdilenYetki = 4; // Ikilik: 100 (Okuma)
    // Eger sonuc kontrol edilen yetkiye esitse, yetki vardir.
    int sonuc = kullaniciYetkisi & kontrolEdilenYetki;
    std::cout << sonuc; // Cikti: ???
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int kullaniciYetkisi = 5; // Ikilik: 101
    int kontrolEdilenYetki = 4; // Ikilik: 100
    // Bitsel VE (&) operatoru, her iki sayida da ayni pozisyondaki
    // bit 1 ise sonucu 1 yapar.
    //   101 (5)
    // & 100 (4)
    // -----
    //   100 (4) -> Sonuc 0'dan buyuk oldugu icin yetki vardir.
    int sonuc = kullaniciYetkisi & kontrolEdilenYetki;
    std::cout << sonuc; // Cikti: 4
}
```

Doğru Çıktı: 4 (Sonucun 0'dan farklı olması yetkinin varlığını gösterir.)

Bulmaca 7: Yeni Yetki Ekleme (Bitsel VEYA)

Bir kullanıcının mevcut yetkisi ÇALIŞTIRMA (değeri 1). Bu kullanıcıya OKUMA (değeri 4) yetkisi de eklenirse, yeni yetki kodu ne olur?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int mevcutYetki = 1; // Ikilik: 001
    int eklenecekYetki = 4; // Ikilik: 100
    int yeniYetki = mevcutYetki | eklenecekYetki;
    std::cout << yeniYetki; // Cikti: ???
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int mevcutYetki = 1; // Ikilik: 001
    int eklenecekYetki = 4; // Ikilik: 100
    // Bitsel VEYA (|) operatoru, herhangi bir sayida ilgili bit 1 ise
    // sonucu 1 yapar. Yetki eklemek icin idealdir.
    //   001 (1)
    // | 100 (4)
    // -----
    //   101 (5)
    int yeniYetki = mevcutYetki | eklenecekYetki;
    std::cout << yeniYetki; // Cikti: 5
}
```

Doğru Çıktı: 5

Bulmaca 8: Ayarları Tersine Çevirme (Bitsel ÖZEL VEYA)

Bir konfigürasyon ayarı 4 bit ile tutuluyor ve mevcut değeri 10 (ikilik: 1010). Bir maske (değeri 12, ikilik: 1100) kullanılarak bu ayarların bazıları tersine çevriliyor (toggle). Yeni ayar değeri ne olur?

Tahmin için Gösterilecek Kod:

```
#include <iostream>
int main() {
    int ayarlar = 10; // Ikilik: 1010
    int maske = 12;   // Ikilik: 1100
    int yeniAyarlar = ayarlar ^ maske;
    std::cout << yeniAyarlar; // Cikti: ???
}
```

Açıklama için Gösterilecek Kod:

```
#include <iostream>
```

```
int main() {
    int ayarlar = 10; // Ikilik: 1010
    int maske = 12;   // Ikilik: 1100
    // Bitsel OZEL VEYA (^), iki bit birbirinden farkliysa sonucu 1
yapar.
    // Bu, maskenin 1 oldugu bitleri tersine cevirir.
    //   1010 (10)
    // ^ 1100 (12)
    // ------
    //   0110 (6)
    int yeniAyarlar = ayarlar ^ maske;
    std::cout << yeniAyarlar; // Cikti: 6
}
```

Doğru Çıktı: 6

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.
- K6. Problemin çözümünde iç içe döngü yapısını uygular.

Amaç

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

Önerilen Ders Akışı

A. Giriş: Taş, Kağıt, Makas (10 dk.)

B. Öğrenme Görevleri

B1. Karar Yapılarını Tanıyalım ve Kodlayalım (40 dk.)

Ders Arası (10 dk.)

B2. İç İçe Koşula Farklı Bir Bakış (50 dk.)

Ders Arası (10 dk.)

B3. Döngüleri Tanıyalım (20 dk.)

B4. Döngülerin Ne İçin Kullanıldığını Keşfedelim (30 dk.)

Ders Arası (10 dk.)

B5. Döngü Görevlerini Kodlayalım (50 dk.)

A. Giriş: Taş, Kağıt, 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.

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

B. Öğrenme Görevleri

B1. Karar Yapılarını Tanıyalım ve Kodlayalım

Süre: 40 dk.

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

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

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

Materyaller: L3.B1.1 Karar Yapılarını Tanıyalım Görev Kâğıtları

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

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

sayılar belirleyerek ve her değişimde bu sayıları değiştirerek grupların o sayıya yönelik programda hangi çıktıyı vereceği öğrenciler tarafından bulunması istenir.

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

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

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

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. Tek satırlık işlemler için ‘{‘ ve ‘}’ karakterlerine gerek yoktur.

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

yerine aşağıdaki şekilde yazılabilir.

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

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

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

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

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

Bu aşamada, öğrencilerin materyalde belirtilen "Karar Yapılarını Tanıyalım Görev Kâğıtları" etkinliği içerisinden seçtiği gerçek yaşamla ilgili iki karmaşık görevi kendi başlarına kodlaması beklenir. Öğrenciler kodlama yaparken eğitmen sınıfta dolaşarak öğrencilerin ihtiyaç duyduğu anda yöntemsel bilgi vermesi beklenir. Ayrıca kodları bilgisayarda yazarak öğrencilerin seçtiği görevlerdeki bazı aşamaları ihtiyaç hâlinde görmesi sağlanır.

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

B2. İç içe Koşula Farklı Bir Bakış

Süre: 50 dk.

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

Materyaller: L3.B2.1. SWITCH Yapısı Kullanımı 1

L3.B2.2. SWITCH Yapısı Kullanımı 2

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

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

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

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

break: Bu komut, koşul bloğundan çıkmamızı sağlar.

default: Verilen koşulların gerçekleşmemesi hâlinde çalışır.

Programlamada hem if-else hem de switch yapıları, belirli koşullara göre program akışını yönlendirmek için kullanılır; ancak kullanım alanları ve yapıları açısından bazı farklılıklar vardır. if-else yapısı, hem sayısal hem de mantıksal ifadelerin kontrolü için uygundur ve koşullar karşılaştırma (==, !=, <, >, vb.) veya mantıksal işlemler (&&, ||) içerebilir. Bu nedenle daha esnek ve çok yönlüdür. Öte yandan switch yapısı yalnızca sabit değerlere (genellikle tamsayılar, karakterler veya bazı dillerde sabit dizeler) göre dallanma yapılacak durumlarda kullanılır. Bu yapı, çok sayıda sabit durumu kontrol etmek gerektiğinde daha okunabilir ve düzenli bir alternatif sunar. Kısacası, if-else genel amaçlı karar yapısıyken, switch daha sınırlı ama belirli durumlarda daha verimli bir seçenektir.

Eğitmen rehberliğinde "L3.B2.2. SWITCH Yapısı Kullanımı 2" adlı materyal kısmında verilen switch kullanımına yönelik iki etkinlikten birinin öğrenciden seçerek kodlama yapması istenir.

B3. Döngüleri Tanıyalım

Süre: 20 dk.

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

Materyaller: L3.B3.1. Döngüleri Tanıyalım 1

L3.B3.2. Döngüleri Tanıyalım 2

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

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

Resim 17. *Çözüm 1*

Resim 18. *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 kullanmamak bir kaba 12 bardak süt dökmek için 12 tane ayrı bardak kullanmaya benzemektedir. Yapılabilir ama mantıklı değildir.

B4. Döngülerin Ne İçin Kullanıldığını Keşfedelim

Süre: 30 dk.

Kazanımlar: K5. Problem çözümünde döngü yapısını kullanır.

Materyaller: L3.B4.1 Döngülerin Ne İçin Yazıldığını Keşfediyorum

L3.B4.2 Döngü Çeşitleri Afişi

Hazırlık: Eğitmen L3.B4.2 kodlu materyali ÖYS üzerinden erişime açar. Diğer L3.B4.1. kodlu materyalden ise 10 adet çıktı alır.

Uygulama: Her iki öğrenciye bir tane verilecek şekilde "L3.B4.1 Döngülerin Ne İçin Yazıldığını Keşfediyorum" adlı materyal öğrencilerle paylaşılır. Öğrencilerden verilen kodun hangi problemi çözmek için veya hangi amaca yönelik olduğunu bulup yazmaları istenir. Eğitmen destekleyici bilgi olarak döngüler için hazırlanan afişi öğrencilerle paylaşarak döngüler hakkında aşağıdaki gibi bir özetleme yapar.

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

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

Bir döngüde olması gerekenler şunlardır;

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

B5. Döngü Görevlerini Kodlayalım

Süre: 50 dk.

Kazanımlar: K6. Problemin çözümünde iç içe döngü yapısını uygular.

Materyal: L3.B5.1 Döngü Görevlerini Kodlayalım

Hazırlık: Eğitmen materyali ÖYS üzerinden erişime açar. Code::Blocks uygulaması

öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Eğitmenler döngü görevi etkinliğindeki uygulamalarından iki tanesini öğrencilerin seçmesini sağlayarak, öğrencilerin bu derste kodlama yapmasını ister. Öğrenciler kodlama esnasında daha önceki derste hazırlanan afişi destekleyici bilgi olarak kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi 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. Bu fonksiyon 0 ila üst sınır (en az 32767 en çok RAND_MAX) arasında rastgele sayı üretir. Üst sınırı sınırlandırmak için mod (%) operatörü kullanılır. 0-100 (100 hariç) arasında rastgele sayı üretmek istersek rand()%100 şeklinde kullanırız. Alt sınırı arttırmak/azaltmak istersek toplama işlemini kullanırız. Örneğin 10 ile 100 arasında rastgele sayı üretmek için 10 + (rand() % 90) şeklinde kullanırız. srand() fonksiyonu rand() fonksiyonu için hazırlayıcı fonksiyondur. Rastgele sayı üreteci için kullanılacak başlangıç değerini ayarlar. Eğer srand() kullanılmaz ise program her çalıştırıldığında aynı rastgele değerler elde edilir. "srand(time(0));" satırı main bloğunun başına eklenerek kullanılabilir.

Döngü görevleri ve cevapları aşağıdaki gibidir:

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

Cevap 1:

```
#include <iostream>
using namespace std;
int main()
{
    int sayi = 1;
    while(sayi<101)
    {
        cout << sayi <<endl;
        sayi ++;
    }
    return 1;
}
```

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

Cevap 2:

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

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

Cevap 3:

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

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

Cevap 4:

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

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

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

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

Cevap 5:

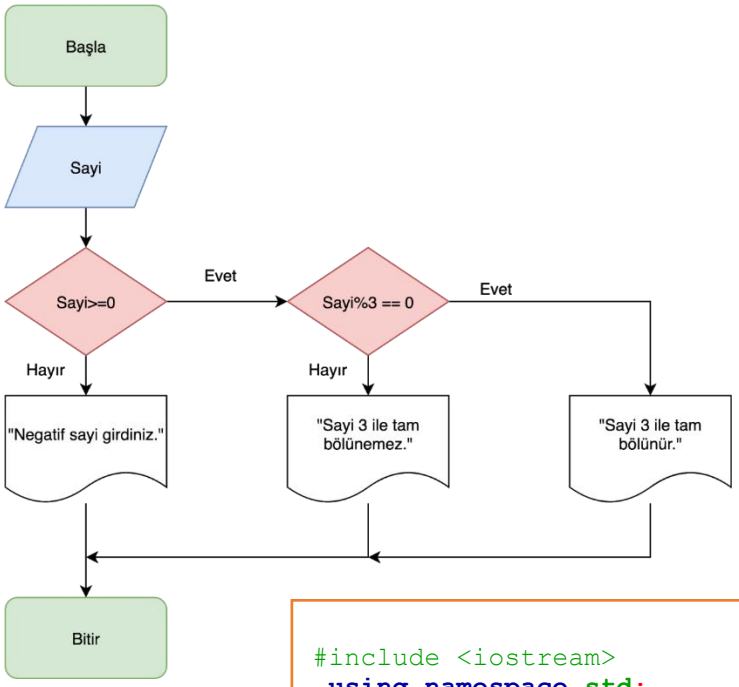
```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<10;i++)
    {
        for(int j=1; j<10;j++)
        {
            cout << i << "*" << j << "=" << i*j <<"\t";
        }
        cout << endl;
    }
}
```

Hafta 3. Yüz Yüze: Ders Materyalleri

L3.B1.1. Karar Yapılarını Tanıyalım Görev Kâğıtları

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

1. Görev




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

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

1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

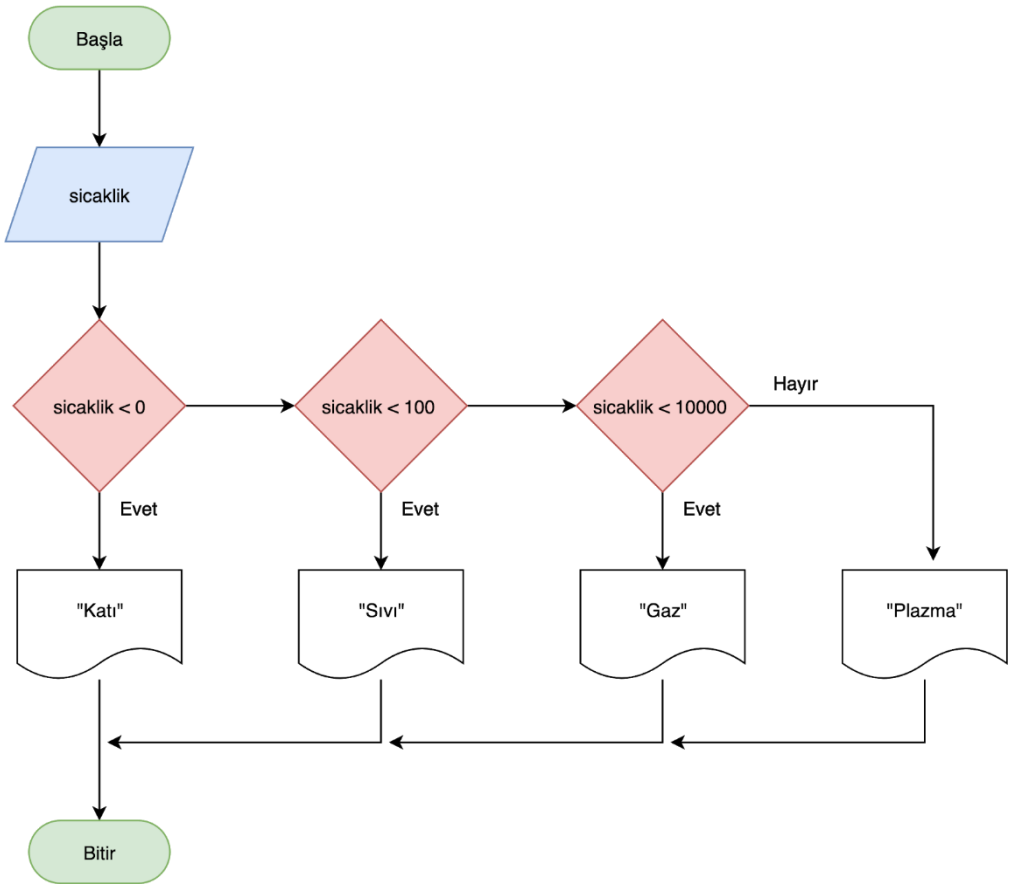
4. **Grup Ekran Çıktısı:**

   ……...

5. **Grup Ekran Çıktısı:**

   ……...

2. Görev



1. **Grup Ekran Çıktısı:**

   ……...

2. **Grup Ekran Çıktısı:**

   ……...

3. **Grup Ekran Çıktısı:**

   ……...

4. **Grup Ekran Çıktısı:**

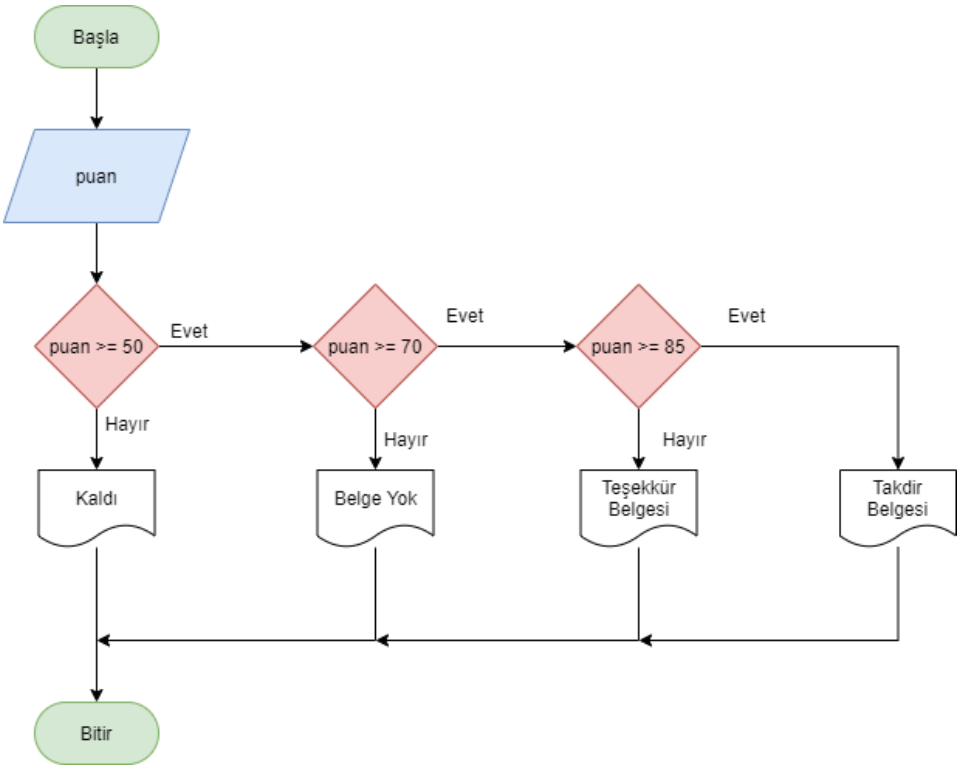
   ……...

5. **Grup Ekran Çıktısı:**

   ……...

```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:**

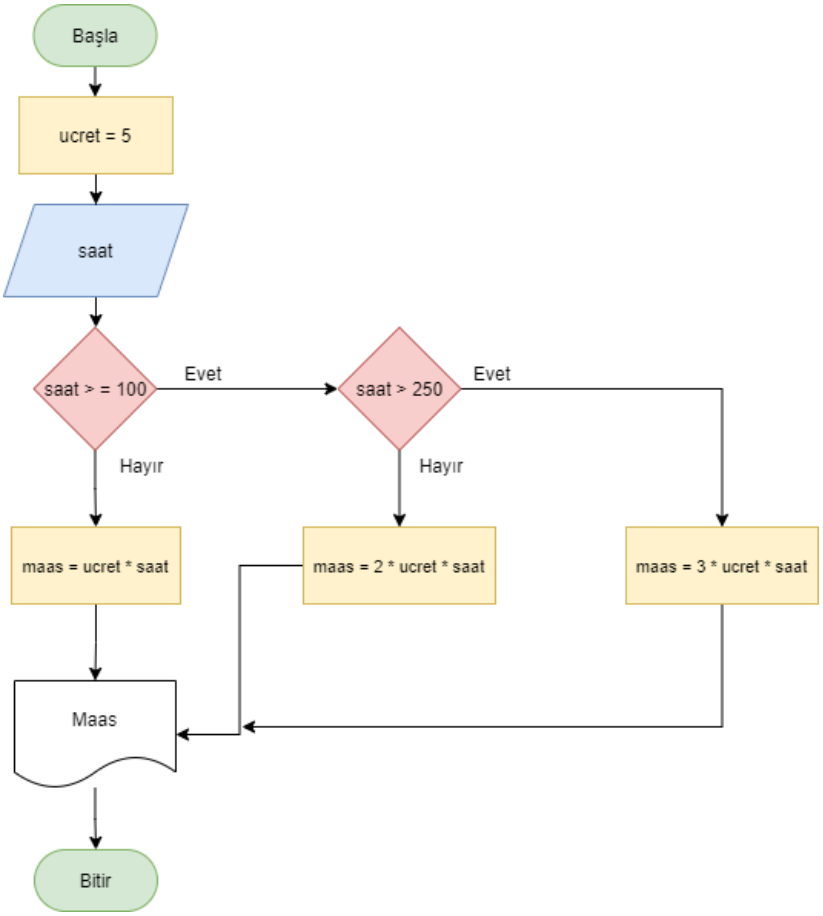
   ……...

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:

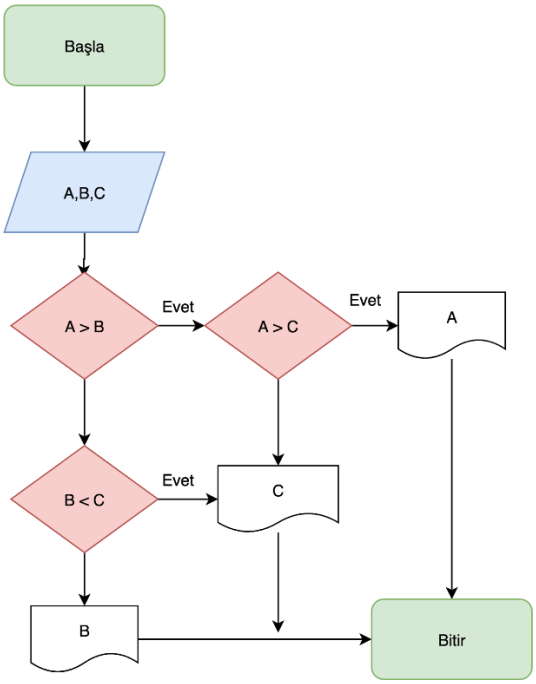
……

5. Grup Ekran Çıktısı:

……

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

5. Görev



1. Grup Ekran Çıktısı:

……

2. Grup Ekran Çıktısı:

……

3. Grup Ekran Çıktısı:

……

4. Grup Ekran Çıktısı:

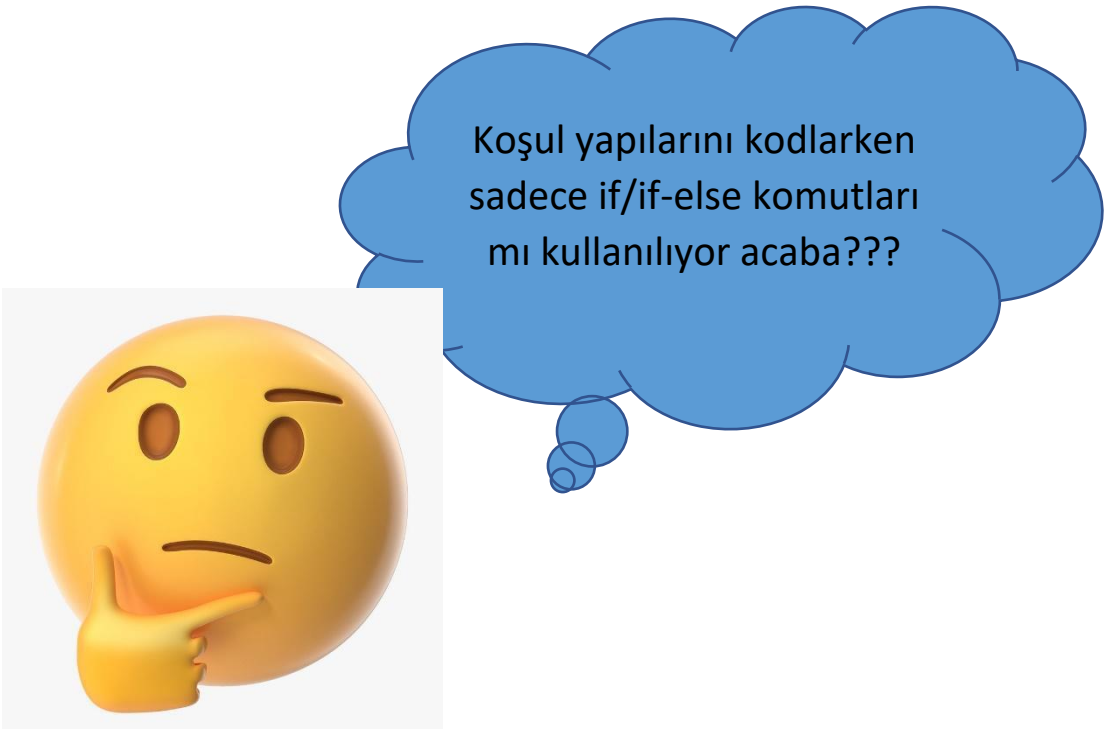
……

5. Grup Ekran Çıktısı:

……

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

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

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

L3.B2.1. SWITCH Yapısı Kullanımı 1



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

L3.B2.2. SWITCH Yapısı Kullanımı 2

If/ If Else Kullanımı

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

Switch/Case Kullanımı

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

If/ If Else Kullanımı

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

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

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

    return 0;
}
```

Switch/Case Kullanımı

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

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

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

L3.B3.1. Döngüleri Tanıyalım 1

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

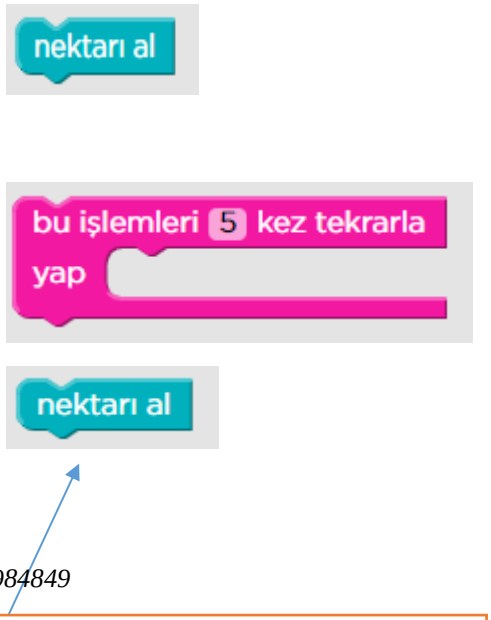
Yukarıdaki arının tüm çiçeklerdeki nektarı almasını isteseydiniz **blok** kodlardan hangisini kaç kere kullanırdınız?

CEVAP:

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

L3.B3.2. Döngüleri Tanıyalım 2

Yazılımcılar programlarında neden döngüleri kullanmayı çok severler?



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

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

CEVAP:

L3.B4.1. Döngülerin Ne İçin Yazıldığını Keşfediyorum

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;`	

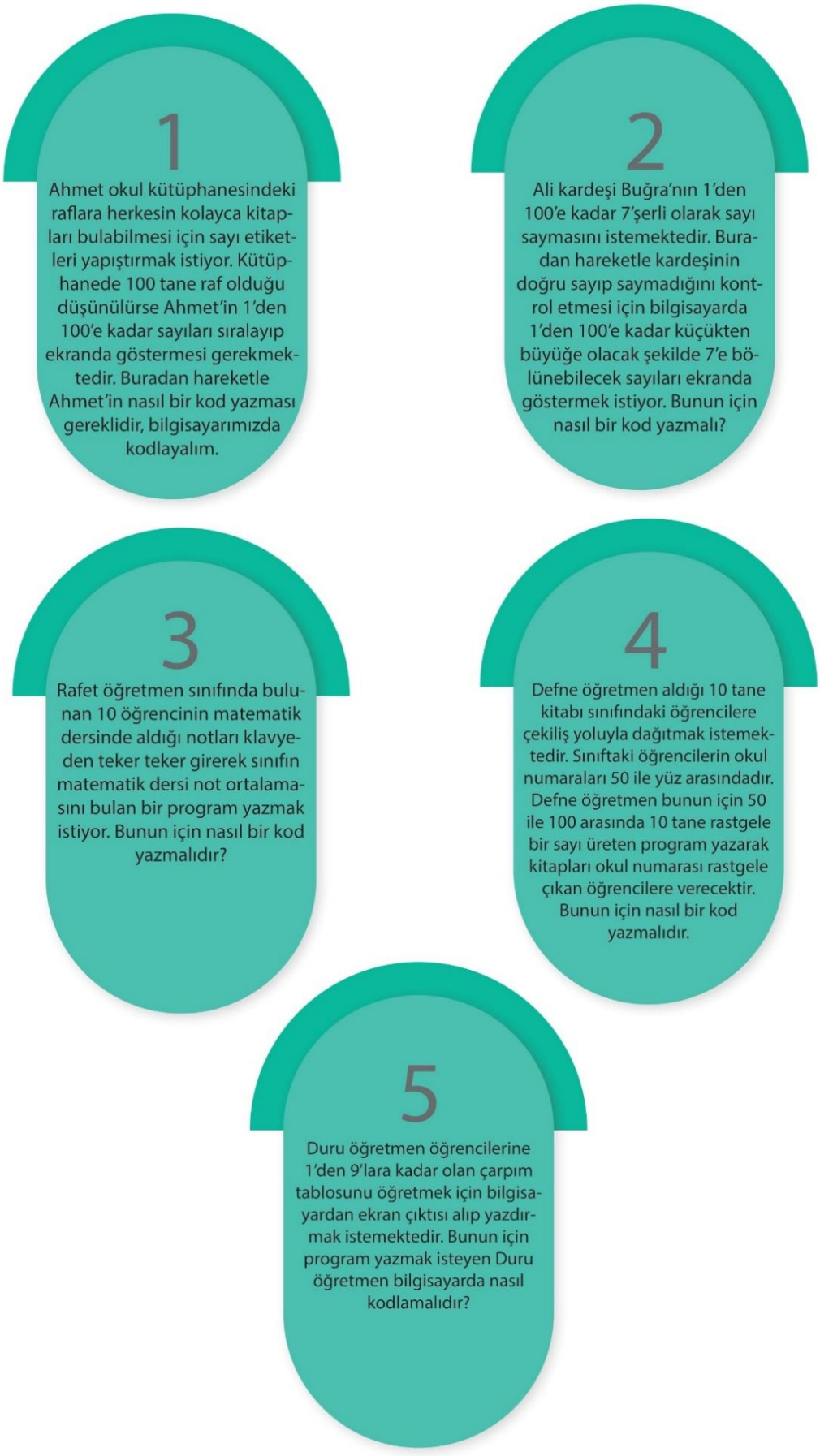
`int i=1;`
`while(i<20)`
`{`
`  if(i%2==1)`
`  cout << i << endl;`
`  i++;`
`}`	

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

L3.B4.2. Döngü Çeşitleri Afişi

Resim 19. *Döngüler*

L5.B5.1. Döngü Görevlerini 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. Spor Salonu Çekiliş Kontrol Programı: Karar Yapıları ile Koşul Kontrolü (15 dk.)

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

B3. Akıştan Koda: Algoritma Dönüştürme Yarışması (15 dk.)

Ders Arası (10 dk.)

B4. Rastgele Skorlar Üzerinde Tek Sayıları Sayma ve Toplama (25 dk.)

B5. Oyun Skor Takip Programı (25 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. Spor Salonu Çekiliş Kontrol Programı: Karar Yapıları ile Koşul Kontrolü

Süre: 15 dk.

Materyaller: LÇ3.B1.1. Öğrenci Yönerge Metni

LÇ3.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 LÇ3.B1.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir spor salonu çekilişinde, üye numarası tek olanların katılabilmesi için numaranın 3 ile, çift olanların katılabilmesi için ise 4 ile tam bölünmesi gerekmektedir. Program, kullanıcıdan üye numarasını isteyecek, geçerli bir sayı olup olmadığını kontrol edecek ve if/else yapısıyla sonucu ekrana yazdıracaktır.

Zaman Planı:

- **0-2 dakika:** Senaryonun okunması, veri tipi seçimi ve koşulların planlanması. Bu kısımda eğitmen düşündükleri veri tipi seçimi ve koşulların planlanması hakkında sorular sorabilir. Bu sorular öğrenciye kod yazımında ip ucu sağlayabilir ancak bu kısımda kod yapısı hakkında detaya girilmemelidir.
- **2-12 dakika:** Kodun öğrenciler tarafından yazılması ve en az iki tek, iki çift sayı ile test edilmesi. Bu kısımda eğitmen öğrenciden gelebilecek sorulara yanıt verecek şekilde hazır olmalı ve öğrencilerin kodu yazması için onları teşvik etmelidir.
- **12-15 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B2. Çalıştırmadan Tahmin Et: Karar Yapılarını Anlama

Süre: 15 dk.

Materyaller: LÇ3.B2.1. Öğrenci Yönerge Metni

LÇ3.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 LÇ3.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.

Materyaller: LÇ3.B3.1. Öğrenci Yönerge Metni

LÇ3.B3.2. 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, LÇ3.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:

- **0-2 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.
- **2-10 dakika:** C++ kodunun Code::Blocks ortamında yazılması. Girdi alma (cin), çıktı verme (cout) ve koşul/döngü yapılarını akış diyagramına uygun şekilde kullanma.
- **10-15 dakika:** Kodun test edilmesi, beklenen çıktılar ile karşılaştırılması ve varsa hataların düzeltilmesi. Eğitmen, doğru çözümleri vurgular ve geliştirme önerileri sunar.

B4. Rastgele Skorlar Üzerinde Tek Sayıları Sayma ve Toplama

Süre: 25 dk.

Materyaller: LÇ3.B4.1. Öğrenci Yönerge Metni

LÇ3.B4.2. 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 LÇ3.B4.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir sınıfta öğretmen mini bir oyun düzenliyor ve 10 farklı skor rastgele belirleniyor (0-100 arası). Program, bu skorlar arasında tek olanların adedini ve toplamını hesaplayıp ekrana yazdıracaktır.

Zaman Planı:

- **0-5 dakika:** Senaryoyun okunması, hangi değişkenlerin kullanılacağını, tek sayı kontrolünü ve toplam hesabın planlanması.
- **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.

B5. Oyun Skor Takip Programı

Süre: 25 dk.

Materyaller: LÇ3.B5.1. Öğrenci Yönerge Metni

LÇ3.B5.2. 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 LÇ3.B5.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir grup arkadaş basketbol oynuyor ve her atış sonrası puanlarını sisteme giriyor. Girilen puan çift sayı ise, toplam puana eklenir. Girilen puan tek sayı ise, o ana kadar girilmiş çift puanların ortalaması ekrana yazdırılır. Bu senaryoya uygun olarak programı bilgisayar ortamında yazınız ve farklı puan girişleri ile çıktıları test ediniz.

Zaman Planı:

- **0-5 dakika:** Senaryonun okunması, hangi değişkenlerin, karar yapılarının ve döngülerin kullanılacağının planlanması.
- **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 3. Çevrim İçi: Ders Materyalleri

LÇ3.B1.1. Öğrenci Yönerge Metni

Göreviniz: Bir spor salonu, üyelerine özel olarak düzenlenen çekilişe katılım şartlarını kontrol eden küçük bir program geliştirmek istiyor. Kurallar şöyle:

Eğer üye numarası tek ise, çekilişe katılabilmesi için numarasının 3 ile tam bölünmesi gerekiyor.

Eğer üye numarası çift ise, çekilişe katılabilmesi için numarasının 4 ile tam bölünmesi gerekiyor.

Klavyeden girilen üye numarasına göre, kişinin çekiliş şartlarını sağladığını veya sağlamadığını ekrana yazdıran programı yazınız.

LÇ3.B1.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int main() {
    int uyeNo;
    cout << "Uye numarasini giriniz: ";
    cin >> uyeNo;
    if (!cin) {
        cout << "Gecersiz giris!" << endl;
        return 1;
    }
    if (uyeNo % 2 != 0) { // tek
        if (uyeNo % 3 == 0)
            cout << "Tek ve 3 ile tam bolunuyor: Cekilise katilabilir." <<
endl;
        else
            cout << "Tek ama 3 ile tam bolunmuyor: Cekilise katilamaz." <<
endl;
    } else { // cift
        if (uyeNo % 4 == 0)
            cout << "Cift ve 4 ile tam bolunuyor: Cekilise katilabilir." <<
endl;
        else
            cout << "Cift ama 4 ile tam bolunmuyor: Cekilise katilamaz." <<
endl;
    }
    return 0;
}
```

LÇ3.B2.1. Öğrenci Yönerge Metni

Arda, Duru'ya kısa bir C++ kodu yazmış ve bu kodu çalıştırmadan önce ekranda nasıl bir sonuç oluşacağını tahmin etmesini istemiştir. Arda'nın 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;
}
```

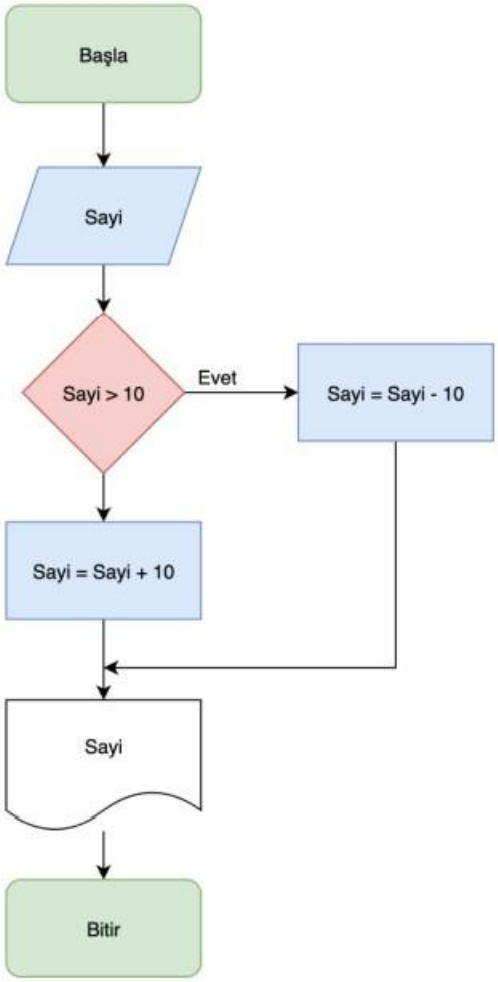
Duru'nun tahmini doğru olduğuna göre sizce nasıl bir tahminde bulunmuştur?

LÇ3.B2.2. Yanıt

Cevap: 6

LÇ3.B3.1. Öğrenci Yönerge Metni

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



LÇ3.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;
}
```

LÇ3.B4.1. Öğrenci Yönerge Metni

Göreviniz: Öğrenciler, aşağıdaki senaryoya uygun bir program yazacaklardır:

Bir sınıfta öğretmen mini bir oyun düzenliyor ve 10 farklı skor rastgele belirleniyor (0-100 arası). Program, bu skorlar arasında tek olanların adedini ve toplamını hesaplayıp ekrana yazdıracaktır.

LÇ3.B4.2. Kod Bloğu

```
#include <iostream>
#include <cstdlib>  // rand, srand
#include <ctime>    // time
using namespace std;
int main() {
    srand(time(0)); // Rastgele sayi uretimi icin baslangic degeri
    int tekAdet = 0;
    int tekToplam = 0;
    cout << "Rastgele 10 skor uretildi:\n";
    for (int i = 0; i < 10; i++) {
        int skor = rand() % 101; // 0-100 arasi rastgele sayi
        cout << skor << " ";
        if (skor % 2 != 0) { // tek sayi kontrolu
            tekAdet++;
            tekToplam += skor;
        }
    }
    cout << "\nTek sayilarin adedi: " << tekAdet << endl;
    cout << "Tek sayilarin toplami: " << tekToplam << endl;
    return 0;
}
```

LÇ3.B5.1. Öğrenci Yönerge Metni

Göreviniz: Bir grup arkadaş basketbol oynuyor ve her atış sonrası **puanlarını sisteme giriyor**.

- Girilen puan **çift sayı** ise, toplam puana eklenir.
- Girilen puan **tek sayı** ise, o ana kadar girilmiş **çift puanların ortalaması** ekrana yazdırılır ve program durdurulur.

Bu senaryoya uygun olarak programı bilgisayar ortamında yazınız ve farklı puan girişleri ile çıktıları test ediniz.

LÇ3.B5.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int main() {
    int sayi, toplam = 0, sayac = 0;
    float ortalama;
    cout << "Cift sayilar giriniz (tek sayi girildiginde program
duracak):\n";
    do {
        cin >> sayi;
        if (sayi % 2 == 0) {
            toplam += sayi;
            sayac++;
        }
    } while (sayi % 2 == 0);
    if (sayac > 0) {
        ortalama = (float)toplam / sayac;
        cout << sayac << " adet sayinin ortalamasi: " << ortalama <<
endl;
    } else {
        cout << "Henuz cift sayi girilmedi." << 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 ve katarlar 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: L4.B1.1. Destekleyici Bilgi: Dizileri Tanıyalım

Çeşitli madenî paralar

Hazırlık: Materyal ÖYS üzerinden erişime açılır.

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

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

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

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

B2. Dizilere Değer Verelim

Süre: 20 dk.

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

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

Materyaller: L4.B2.1. Destekleyici Bilgiler: Dizilere Değer Verelim

L4.B2.2. Görev Kartları: Dizilere Değer Verelim

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

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

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

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

B3. Döngülerle Diziler

Süre: 25 dk.

Kazanımlar: K4. Dizileri döngü içinde kullanır.

Materyaller: L4.B3.1. Destekleyici Bilgi: Döngülerle Diziler

L4.B3.2. Görev Kartı: Döngülerle Diziler

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

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

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

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

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

Görev İç Çember Yanıtı	Görev Dış Çember Yanıtı

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

B4. Dizilerle Kodlayalım

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: L4.B4.1. Görev Kodları

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

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

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

Öğrenciler ikili gruplar hâlinde çıkan görev üzerinde 5 dk. kadar çalışır. Torbada toplam 10 görev kodu olmasına rağmen aynı görev kodundan iki tane bulunmaktadır. Bu nedenle beş dk. sonra aynı kod üzerinde çalışan çiftlerin bir araya gelmeleri istenir. Bu şekilde dörtlü grup olan öğrenciler çalıştıkları görev üzerinde yaptıklarını birbirlerine aktarır. Eğitmen dörtlü grupların 3-5 dk. kadar kendi aralarında tartışmalarına izin vermelidir. Süre sonunda grupların yaptıklarını incelemek için onları dinler, sorularını cevaplar ve yanlış öğrenmelerin önüne geçer. Eğitmen öğrencilerden tüm grupların üzerinde çalıştıkları görevleri ve çözümlerini Öğrenme Yönetim Sisteminde paylaşmalarını ister. Öğrencilerden üzerinde çalıştıkları görev

dışında kalan diğer dört görevi evde tamamlamaları istenebilir.

Eğitmene Öneriler: Verilen görevlerin doğru yanıtları aşağıda yer almaktadır.

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

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

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

Görev Kodu 4: Bu görevde program girilen dizide 3'e bölünebilen ile 5'e bölünebilen sayıların adedini ayrı ayrı ekrana yazdırmayı amaçlamaktadır.

Görev Kodu 5: Bu görevde program girilen dizide tek ve çift sayıların adedini ekrana yazdırmayı amaçlamaktadır.

B5. Kodlama Ekibi

Süre: 25 dk.

Kazanımlar: K5. Karşılaşılan problemlere dizileri kullanarak çözüm üretir.

Materyaller: L4.B5.1. Problem

L4.B5.2. İç Çember-Dış Çember Görev Kartı

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

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

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

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

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

Eğitmene Öneriler: İç çember ve dış çember grupları, sınıfın düzenine göre öğrenci mevcudunun ikiye bölünmesi ile grup 1 ve grup 2 şeklinde de oluşturulabilir. Problemin kodlarını aşağıda bulabilirsiniz. Dört kişilik grup toplandığında, 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, sat, sut, matris1[5][5], matris2[5][5];
    sat = 5;
    sut = 5;

    cout << "Matrisin elemanlarini sirayla giriniz: " << endl;
    for(i =0; i < sat; i++) {
        for(j=0; j < sut; j++) {
            cin >> matris1[i][j];
        }
    }
    for(i=0; i < sat; i++) {
        for(j=0; j < sut; j++) {
            matris2[j][i] = matris1[i][j];
        }
    }
    cout<<"Matrisin Yer Degistirilmis Hali: " << endl;
    for(i=0; i < sut; i++) {
        for(int j=0; j < sat; j++) {
            cout << matris2[i][j] << " ";
        }
        cout << endl;
    }
}
```

B6. Farkı Bul

Süre: 25 dk.

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

Materyaller: L4.B6.1. Farkı Bul

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

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

> *Koda bakacak olursanız klavyeden girilen karakterlerden sırasıyla "Arda" ismini karakter dizisi kullanarak, "Duru" ismini ise "katar" kullanarak ekrana yazdırıyoruz.*

Buradaki dizi ve katar arasındaki farkı kodun basamaklarını tek tek inceleyerek tahmin etmeye çalışın.

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

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

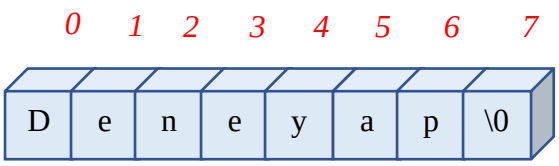
Programlamada metin türünde verilerimizi saklamak için kullanılan özel karakter dizileridir. Katarlar, null ('\0') karakter ile sonlandırılmış tek boyutlu karakter dizileri olarak tanımlanabilir. Aşağıda verilen örnekte, "Deneyap" sözcüğünden oluşan bir katar oluşturulmaktadır. Normalde verilen kelime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

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

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

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

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



B7. Arama Yapalım

Süre: 25 dk.

Kazanımlar: K7. Diziler ve katarlar üzerinde arama yapabilir.

Materyaller: L4.B7.1. Dizide Arayalım

L4.B7.2. Katarda Arayalım

Hazırlık: Eğitmen, ders öncesinde öğrencilerin ikili gruplar halinde çalışabilmesi için laboratuvardaki bilgisayarları hazırlar. Her bilgisayarda Code::Blocks programını açar ve L4.B7.1. Dizide Arayalım ile L4.B7.2. Katarda Arayalım materyallerindeki C++ kodlarını programa ayrı ayrı sekmelerde yapıştırarak öğrencilerin kullanımına hazır hale getirir.

Uygulama: Eğitmen, etkinliği bir oyunla başlatır: "Şimdi sizinle basit bir sayı tahmin oyunu oynayacağız. Aklımda bir sayı listesi var. Sizden biri rastgele bir sayı söylediğinde, ben o sayının listede olup olmadığını nasıl kontrol ederim?" Eğitmen, öğrencilerden gelen cevaplar üzerine, "Tıpkı bir listede bir ismi arar gibi, listedeki her sayıya en baştan başlayarak sırayla bakarım. 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 **L4.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 sayılar girerek ne yaptığını anlamaya çalışın. Anladığınız her satırın yanına bir yorum satırı ekleyerek kodun ne işe yaradığını açıklayın."

Sürenin sonunda eğitmen, kodu çalıştırarak tartışmayı başlatır: "Evet arkadaşlar, keşifleriniz neler? 'break;' komutunun ne işe yaradığını fark eden bir grup var mı?" veya "'bool bulundu' satırı olmasaydı programımız nasıl bir sorun yaşardı?" gibi sorularla öğrencilerin kendi bulgularını paylaşmalarını sağlar. "Program, aranan sayıyı bulduktan sonra listeyi gezmeye devam etmeli mi?" sorusuyla break; komutunun verimliliğini tartışmaya açar. Ardından, "bool bulundu değişkenini ve ilgili satırları silersek ne olur?" sorusunu sorar, kodu düzenleyip tekrar çalıştırarak bu "bayrak" değişkeninin programın hafızası olarak nasıl çalıştığını uygulamalı olarak gösterir. Böylece tartışmayı yöneterek ana fikirleri pekiştirir.

Bu bölümü tamamladıktan sonra eğitmen, "Peki bu arama mantığını sadece sayılar için mi kullanabiliriz? Elbette hayır. Aynı mantıkla bir metnin, yani bir kataın içinde harf de arayabiliriz." diyerek L4.B7.2 koduna geçiş yapar. Eğitmen, bu kodun da aynı temel mantığa sahip olduğunu vurgular: yine bir for döngüsü, bir if koşulu ve bir bulundu bayrağı vardır. Aradaki farkların <string> kütüphanesi, metin almak için getline kullanımı ve dizinin uzunluğunu metin.length() ile dinamik olarak öğrenmek olduğunu belirtir. Eğitmen bu kodu da çalıştırır, öğrencilerden gelebilecek soruları yanıtlayarak bu bölümü tamamlar.

B8. Sıralayalım

Süre: 25 dk.

Kazanımlar: K8. Diziler üzerinde basit sıralama yapabilir.

Materyaller: L4.B8.1. Dizi Sıralama

Hazırlık: Eğitmen, sınıfın ortasında öğrencilerin rahatça hareket edebileceği bir alan oluşturur. Projeksiyon cihazına bağlı bilgisayarında Code::Blocks programını açar ve L4.B8.1. Dizi Sıralama materyalindeki C++ kodunu programa yapıştırarak hazır hale getirir.

Uygulama: Eğitmen, etkinliği lise öğrencilerinin enerjisine uygun bir meydan okumayla başlatır: "Haydi bir oyun oynayalım! Bana gönüllü 5 arkadaş lazım." Eğitmen, gönüllü öğrencileri sınıfın önüne rastgele bir sırada dizer ve diğer öğrencilere döner: "Arkadaşlar, önünüzde karışık bir 'dizi' duruyor. Amacımız, bu diziyi boy sırasına göre kısadan uzuna doğru sıralamak. Ama bir kuralımız var: Her adımda sadece iki kişinin yerini değiştirebilirsiniz. Sınıf olarak, bu sıralamayı nasıl yapardınız? İlk sıraya en kısa kişiyi nasıl getiririz, komutlar sizde!" Eğitmen, sınıfın yönlendirmesiyle (örn: "Ali'yi Ayşe ile karşılaştır," "Mehmet en kısa, onu en baştaki Veli ile yer değiştir") ilk elemanı bulma ve yer değiştirme işlemini canlandırır. Bu işlemi ikinci ve üçüncü en kısa öğrenciler için de tekrarlatır. Bu fiziksel canlandırma tamamlandıktan sonra eğitmen, "Harika! İşte az önce yaptığınız bu işleme 'Seçmeli Sıralama' deniyor. Şimdi gelin, sınıfça yaptığımız bu oyunun C++ dilindeki karşılığına bakalım." diyerek projeksiyonda hazır olan **L4.B8.1** kodunu gösterir.

Eğitmen, tartışmayı şu sorularla yönetir: "Bu kodda iki tane iç içe döngü var. Dış döngü, sıralama işlemini kaç kez yapacağımızı kontrol ederken, iç döngü tam olarak ne iş yapıyor olabilir?" Öğrencilerden gelen cevaplar üzerine iç döngünün en küçük elemanı aradığını pekiştirir. Ardından, "Peki, en küçük elemanı bulduktan sonra oyunda yaptığımız 'yer değiştirme' işlemini kodun hangi bölümü gerçekleştiriyor?" diye sorarak takas (swap) bölümünü bulmalarını sağlar. Eğitmen, öğrencilerden farklı dizi örnekleri vermelerini ister ve bu yeni dizilerle kodun adımlarını beyaz tahtada izleyerek hangi elemanın seçildiğini ve nasıl yer değiştirildiğini birlikte inceler. Son olarak eğitmen, "Peki, kodun içindeki o yer değiştirme (swap) bölümü hiç olmasaydı ne olurdu? Program yine de en küçük elemanı bulur muydu? Ama sıralama gerçekleşir miydi?" sorusuyla tartışmayı derinleştirir ve takas işleminin önemini vurgular.

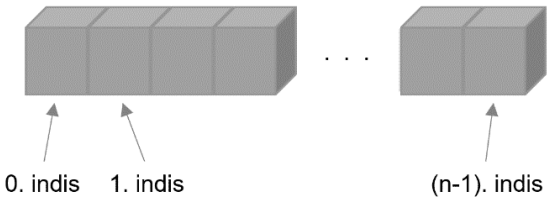
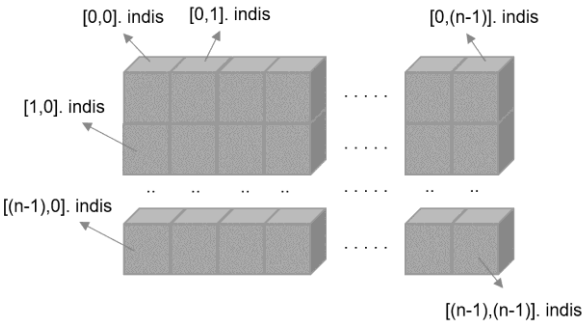
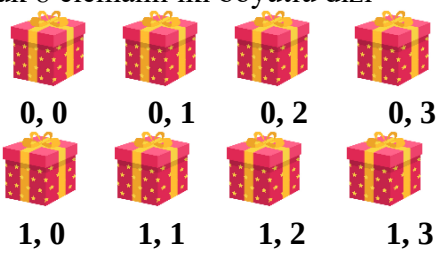
Eğitmene Öneriler: Bu fiziksel canlandırma, lise öğrencilerinin soyut bir sıralama algoritmasını somut bir deneyimle ilişkilendirmesi için oldukça etkilidir. Kod incelemesi sırasında, öğrencilerden gelen farklı dizi örneklerini beyaz tahtaya yazarak dış döngünün her adımında dizinin nasıl değiştiğini adım adım göstermek, algoritmanın anlaşılmasını kolaylaştıracaktır.

Hafta 4. Yüz Yüze: Ders Materyalleri

L4.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 indis numaralandırması sıfırdan başlar. Bu nedenle n elemandan oluşan tek boyutlu bir dizideki son elemanın indisi n değil (n-1) olur.	 ***UYARI***: Ayrıca çok boyutlu bir dizide saklanabilecek toplam eleman sayısı, tüm boyutların çarpımı ile hesaplanabilir. 3*2, 6 elemanlı dizi anlamına gelir.
5 elemanlı tek boyutlu dizi  **0 1 2 3 4** İndisler	2*4'lük 8 elemanlı iki boyutlu dizi 
3 elemanlı tek boyutlu hatalı dizi: Dizi aynı veri türünde olmalıdır. 	2*3'lik iki boyutlu ancak hatalı dizi: Dizi aynı veri türünde olmalıdır. 

L4.B2.1. Destekleyici Bilgi: Dizilere Değer Verelim

Tek Boyutlu Diziler: Tek bir veri türü içeren ve birden fazla değişkeni bir arada tutmaya yarayan veri yapısıdır.	***Çok Boyutlu Diziler:*** Dizilerin elemanları da bir dizi tutabilir. Bu dizileri ifade etmek için dizilerin dizisi ya da dizilerden oluşan diziler ifadesi kullanılır. Aşağıda iki boyutlu dizi örneği verilmektedir.
veri_tipi dizi_adi[eleman_sayisi]={değer1, değer2,...}; Beş adet öğrenciye ait öğrenci notunu saklamak için oluşturulan "notlar" dizisine değer atamak için: **int** notlar[5] = {90, 70, 50, 80, 85}; **Uyarı!** Notlar dizisi 5 elemana sahiptir.	veri_tipi dizi_adi[boyut1][boyut2]…[boyutN]= {değer1, değer2,..değerN}; Dört adet öğrencinin bir dersten aldığı iki yazılı notunu saklamak için oluşturulan "notlar" dizisine değer atamak için: **int** notlar[4][2] = {{90, 70}, {50, 80}, {85, 86}, {50, 70}}; **Uyarı!** Notlar dizisi 4*2 = 8 elemana sahiptir.
veri_tipi dizi_adi[] = {değer1, değer2, değer3, …}; Beş adet öğrenci adlarının ilk harflerini saklamak için oluşturulan "harfler" dizisine değer atamak için: **char** harfler[] = {'H', 'K', 'A', 'R', 'S'}; **Uyarı!** Derleyici, bu şekilde dizinin elemanları verildiği zaman dizi boyutunun ne olacağına kendisi karar verecektir.	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 notlar[5] = {90, 70, 50}; **int** notlar[5] = {90, 70, 50, 0, 0}; **Uyarı!** Dizi ilk değer atamasında eleman sayısı 5'tir. Ancak diziye sadece üç değer girilmiş, yani eleman sayısı kadar değer girilmemiştir. Bu durumda 5 elemanlı dizinin 4. ve 5. elemanlarının değeri 0 olacaktır. Yukarıdaki iki tanımlamada aynıdır.	**int** sayilar [4][3][5]; Yukarıda tanımlanan *sayilar* dizisi 4 * 3 * 5 = 60 elemana sahiptir. **Uyarı!** Çok boyutlu bir dizide istediğiniz sayıda boyuta sahip olabilirsiniz. Ancak, çok fazla boyut tanımlamanız bilgisayarın belleğini hızlı bir şekilde doldurabilir. Çok boyutlu dizilerin en basit formu iki boyutlu dizilerdir. İki boyutlu bir dizi dizi elemanlarının da bir dizi olması hâlidir.

L4.B2.2. Görev Kartları: Dizilere Değer Verelim

Tek Boyutlu Diziler	*Çok Boyutlu Diziler*

Sıra Sizde!
 	
`int notlar[5] = {90, 70, 50, 80, 85};`
 	
Örnekteki notlar dizisine benzer bir başka dizide siz oluşturun ve diziye değer atayın. Yazdığınız dizinin eleman sayısını birlikte tartışın.
	Sıra Sizde!
	
	`int notlar[3][5]={{90, 70, 50, 80, 85},`
	`{86, 50, 70, 90, 95} };`
	
	Örnekteki notlar dizisine benzer bir başka dizide siz oluşturun ve diziye değer atayın. Yazdığınız dizinin eleman sayısını birlikte tartışın.

Sıra Sizde!
 	
`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?

Sıra Sizde! `int notlar[6] = {90, 70, 50};` Yukarıdaki dizi başka hangi şekilde yazılabilirdi?	**Sıra Sizde!** `int sayilar [2][5][4];` Yukarıdaki dizi kaç elemanlıdır?

L4.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*

L4.B3.2. Görev Kartı: Döngülerle Diziler

Problem: 6 arkadaşının doğduğu yılı ekrana yazdıran program.

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

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

    return 0;
}
```

Kodun Çıktısı:

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

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

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

L4.B4.1. Görev Kodları

Görev Kodu 1

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

Görev Kodu 2

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

Görev Kodu 3

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

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

Görev Kodu 4

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

Görev Kodu 5

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

L4.B5.1. Problem

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

Hafta 1

Skor Tablosu:

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

Hafta 2

Skor Tablosu:

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

L4.B5.2. İç Çember-Dış Çember Görev Kartı

İç Çember Görev Kartı

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

Dış Çember Görev Kartı

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

Kodlama Ekibi Görev Kartı

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

L4.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;
}
```

L4.B7.1. Dizide Arayalım

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

int main() {

    int sayilar[] = {17, 29, 44, 65, 78, 40, 53, 12};
    int diziBoyutu = 8;

    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;
}
```

L4.B7.2. Katarda Arayalım

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

int main() {
    string metin;
    char harf;
    bool bulundu = false;

    cout << "Bir metin girin: ";
    getline(cin, metin);

    cout << "Aramak istediginiz harfi girin: ";
    cin >> harf;

    for (int i = 0; i < metin.length(); ++i) {
        if (metin[i] == harf) {
            cout << harf << " harfi bulundu! Konumu: " << i << endl;
            bulundu = true;
        }
    }

    if (!bulundu) {
        cout << " harfi metin icinde bulunamadi." << endl;
    }

    return 0;
}
```

L4.B8.1. Dizi Sıralama

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

int main() {
    int sayilar[] = {65, 42, 33, 78, 10, 80, 25, 46};
    int n = 8;

    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 amacı, öğrencilerin dizi ve katar yapılarını kullanarak problemleri çözmelerini sağlamaktır. 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 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.
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: Derse Isınma (5 dk.)

B. Öğrenme Görevleri

B1. Kod Çıktısını Bulalım (10 dk.)

B2. Kodu Tamamlayalım (20 dk.)

B3. İkili Çalışma: Kullanıcı Adı Oluşturma (15 dk.)

Ders Arası (10 dk.)

B4. Final Projesi ve Değerlendirme (50 dk.)

A. Giriş: Derse Isınma

Süre: 5 dk.

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

Uygulama: Etkinlik akışı, dersin ilk 1 dakikasında öğrenciler çevrimiçi sınıfa katılırken müziğin çalınmasıyla başlar. Bu sürede öğrencilerin derse girişi beklenir. Müzik bittikten sonra eğitmen, bu dersin amacının yüz yüze eğitimde öğrenilen dizi ve katar konularını uygulamalı olarak pekiştirmek olduğunu kısaca açıklar. Ardından, öğrencilere sohbet ekranı üzerinden şu soruyu yöneltir: "Geçen haftaki yüz yüze dersimizden diziler ve katarlar konusuyla ilgili aklınızda en çok ne kaldı? Tek bir kelime veya kısa bir cümle ile yazar mısınız?" Gelen cevapları sesli bir şekilde okuyarak ve üzerine kısa yorumlar yaparak derse başlangıç yapılır ve bir sonraki etkinliğe geçilir.

B. Öğrenme Görevleri

B1. Kod Çıktısını Bulalım

Süre: 10 dk.

Materyaller: LÇ4.B1.1. Görev Kodu: Kod Ne Çıktı Verir?

Hazırlık: Eğitmen LÇ4.B1.1 materyalindeki kod bloğu, IDE'ye aktarır.

Uygulama: Etkinlik akışı, eğitmen ilk 3 dakikayı ekranını paylaşarak Code::Blocks derleyici üzerinde hazırlanan LÇ4.B1.1 Görev Kodu'nu öğrencilere göstermesiyle başlar. Eğitmen, kodun ne iş yaptığını sormadan sanal sınıf etkileşim araçlarından anlık çevrimiçi bir anket başlatır. Ankette öğrencilere şu soru yöneltilir:

Anket Sorusu: Gördüğünüz C++ kod bloğu çalıştırıldığında ekrana hangi çıktı yazdırılır?

A) 78 bulundu! Konumu (indeksi): 4

B) 78 bulundu! Konumu (indeksi): 5

C) bulundu! Konumu (indeksi): 4

D) Kod hata verir, çalışmaz.

Öğrencilerin cevaplaması için 2 dakikalık bir süre tanınır ve bu süre ekranda bir sayaç ile gösterilebilir. Anket süresi tamamlandıktan sonra, eğitmen gelen cevapları kısaca değerlendirir ve ardından derleyici üzerinde kodu çalıştırarak doğru çıktıyı gösterir. Eğitmen etkinliğin kalan 5 dakikalık bölümü, doğru sonuç üzerinden dizilerin temel mantığını pekiştirmek için kullanılır.

Eğitmene Öneriler: Etkinliği yönetirken aşağıdaki noktalara dikkat edebilirsiniz.

- **Kodun Doğru Çıktısı:** Öğrencilerle birlikte kodu inceledikten ve tahminlerini aldıktan sonra, doğru çıktının ne olduğunu teyit edin. Görseldeki kod çalıştırıldığında ekrana yazılacak olan çıktı şudur:

78 bulundu! Konumu (indeksi): 4

- **"Neden 5 Değil de 4?" Sorusunu Vurgulayın**: Öğrencilerin en sık karıştırdığı nokta, dizilerde saymanın **sıfırdan** başlamasıdır. "78" sayısı listedeki 5. eleman olmasına rağmen, indis numarasının "4" olmasının sebebini (0. indis, 1. indis, 2. indis, 3. indis, 4. indis şeklinde) bir kez daha vurgulamak, bu temel kuralın pekişmesini sağlayacaktır.

B2. Kodu Tamamlayalım

Süre: 20 dk.

Materyaller: LÇ4.B2.1. Görev Tanımı

Hazırlık: Eğitmen, LÇ4.B2.1 materyalindeki eksik kod bloğunu paylaşımlı bir çevrimiçi IDE'ye önceden hazırlar. Eğitmen, öğrencilerin tamamladıkları çözümleri gönderecekleri bir Padlet panosu oluşturur ve paylaşım linkini hazır bulundurur.

Uygulama: Eğitmen, ekranını paylaşarak derleyici üzerindeki eksik kod bloğunu (LÇ4.B2.1) öğrencilere gösterir. Bu kod bloğunun çözdüğü problem:

Bir grup arkadaş birlikte bir bilgisayar oyunu oynamıştır ve her birinin aldığı puanlar bir listede (puanlar dizisi) tutulmaktadır. Görevimiz, bu listedeki tüm puanları kontrol ederek o oyun turundaki en yüksek puanı bulmak ve ekrana yazdırmaktır. Kod, bu problemi çözmek için önce ilk oyuncunun puanını geçici olarak "en yüksek" kabul eder, ardından diğer tüm puanları tek tek bu "en yüksek puan" ile karşılaştırır. Eğer daha yüksek bir puana denk gelirse, "en yüksek puan" değişkenini günceller. Döngü bittiğinde ise eldeki son değer, grubun aldığı en yüksek puandır.

Eğitmen görevi açıkladıktan sonra öğrencilerin, kod içindeki yorum satırlarında belirtilen eksik if koşulunu tamamlamak için bireysel çalışacaklarını söyler. Eğitmen, bu görev için öğrencilere bireysel olarak 5-7 dakika süre tanır. Süre sonunda eğitmen, öğrencilerden tamamladıkları kodun ekran görüntüsünü veya kodun kendisini Padlet panosuna yüklemelerini ister. Dersin kalan süresinde eğitmen, Padlet'e yüklenen birkaç farklı öğrenci çözümünü ekrana yansıtır, doğru çözümü gösterir ve kodun çalışma mantığını açıklar.

Eğitmene Öneriler: Süreci yönetirken aşağıdaki adımları izleyebilirsiniz. Öğrencilerin tamamlaması gereken eksik kod parçası şudur:

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

Öğrenciler zorlanırsa, onlara "Şampiyonu Bulma Oyunu" analojisini hatırlatın.

- İlk olarak, enYuksekPuan değişkeninin "şimdilik şampiyon" olduğunu belirtin.
- Döngünün, sıradaki yarışmacıların şampiyona meydan okuması olduğunu anlatın.

- "Sıradaki oyuncunun puanı, şampiyonun puanından daha yüksekse ne yapmalıyız?" sorusuyla if koşulunun mantığını, "Peki, yeni şampiyonu nasıl ilan ederiz?" sorusuyla da atama (=) işlemini bulmalarına yardımcı olun.

B3. İkili Çalışma: Kullanıcı Adı Oluşturma

Süre: 15 dk.

Materyaller: LÇ4.B3.1. Görev Kartı: Kullanıcı Adı Oluşturma

Hazırlık: Eğitmen, çevrimiçi platformunda 2 kişilik çalışma odaları hazırlar. LÇ4.B3.1 materyalindeki görev tanımını ve problem durumunu bir metin belgesine kopyalar. Eğitmen, öğrencilerin tamamladıkları kod parçalarını ve yorumlarını toplayacağı bir Padlet panosu grup çalışması olacak şekilde raf temelli bir düzende oluşturur ve paylaşım linkini hazırlar.

Uygulama: Eğitmen, "Şimdiye kadar dizileri ve katarları ayrı ayrı kullandık. Peki bu ikisi bir metni birleştirmek gibi basit bir işlemde nasıl davranır? Bunu bir kullanıcı adı oluşturma problemi üzerinden göreceğiz." diyerek etkinliği başlatır. Eğitmen, problemi ve görevi öğrencilere açıklar: Problemimiz şu:

> *Yeni bir oyun platformuna kaydoluyorsunuz ve sistem, kullanıcı adınızın isminiz ve doğum yılınızın birleşiminden oluşmasını istiyor. Göreviniz ise, kullanıcıdan ismini ve doğum yılını string olarak alıp birleştirerek 'Ahmet2008' gibi bir kullanıcı adı oluşturmak ve ekrana yazdırmak.*

Eğitmen, bu görevi içeren LÇ4.B3.1 materyalini ve Padlet linkini sohbetten paylaşır. Ardından eğitmen, öğrencileri 2 kişilik odalara ayırır ve kendi IDE ortamlarında görevi tamamlamaları için 10 dakika süre verir; bu sırada grup üyelerinin ekran paylaşımı yaparak birlikte çalışmasını teşvik eder. Süre sonunda eğitmen, öğrencilerden çözümlerini ve görev kartındaki tartışma sorusuna verdikleri cevabı Padlet panosuna yüklemelerini ister. Eğitmen, Padlet üzerinden rastgele seçtiği 1-2 grup çözümünü ekranına yansıtır ve inceler. Son olarak eğitmen, string ile birleştirmenin ne kadar kolay olduğunu, char dizi ile ise bunun neden doğrudan mümkün olmadığını beyaz tahta üzerinde görselleştirerek açıklar.

Eğitmene Öneriler: Odaları gezerken eğitmen, gruplara "İki string'i '+' operatörü ile birleştirebildiniz mi?", "Aynı şeyi char dizileri için denediğinizde derleyici ne hata verdi?" gibi yönlendirici sorular sorar. Bu, öğrencilerin aradaki farkı kendilerinin keşfetmesini sağlar.
Aşağıda, görevin tüm adımlarını yerine getiren, açıklama satırları eklenmiş C++ kodu bulunmaktadır.

```
#include <iostream>
#include <string> // string veri tipini kullanabilmek icin bu kutuphane
gereklidir.

using namespace std;

int main() {
    // --- 1. Adim: Degiskenleri Olusturma ---
    // Kullanicinin ismini ve dogum yilini saklamak icin iki string
degiskeni tanimliyoruz.
    string isim;
```

```
    string dogumYili;

    // --- 2. Adim: Kullanicidan Veri Alma ---
    // Kullanicidan ismini istiyoruz ve 'cin' ile 'isim' degiskenine
kaydediyoruz.
    cout << "Lutfen isminizi giriniz: ";
    cin >> isim;

    // Kullanicidan dogum yilini istiyoruz ve 'dogumYili' degiskenine
kaydediyoruz.
    cout << "Lutfen dogum yilinizi giriniz: ";
    cin >> dogumYili;

    // --- 3. Adim: String'leri Birlestirme ---
    // Tipki sayilari toplar gibi, '+' operatorunu kullanarak iki metni
birlestiriyoruz.
    string kullaniciAdi = isim + dogumYili;

    // --- 4. Adim: Sonucu Ekrana Yazdirma ---
    // Olusturdugumuz yeni kullanici adini ekrana yazdiriyoruz.
    cout << "Harika! Yeni kullanici adiniz: " << kullaniciAdi << endl;

    return 0;
}
```

B4. Final Projesi ve Değerlendirme

Süre: 50 dk.

Materyaller: LÇ4.B4.1. Problem Tanımı: Not Ortalaması Hesaplayıcı

Hazırlık: Eğitmen, video konferans platformunda 4 kişilik odalar hazırlar. " Not Ortalaması Hesaplayıcı " probleminin tanımını içeren LÇ4.B4.1 materyali ve beklenen çıktı, odalara kolayca gönderilebilecek şekilde padlette paylaşılır. Eğitmen, grupların nihai çözümlerini toplayacağı bir Padlet panosu grup çalışması olacak şekilde raf temelli bir düzende oluşturur ve paylaşım linkini hazırlar.

Uygulama: Eğitmen, "Dersimizin son projesinde, öğrendiğimiz her şeyi bir araya getirerek gerçek hayata benzer bir problemi çözeceğiz." diyerek etkinliği başlatır. Eğitmen, LÇ4.B4.1 materyalindeki "Not Ortalaması Hesaplayıcı" problemini ve beklenen çıktıyı ana ekranda tanıtır.

Eğitmen, öğrencileri 4 kişilik odalara ayırır; problem tanımını, Padlet linkini sohbetten gönderir (5 dk). Eğitmen, gruplara problemi çözmek için algoritma tasarlamaları ve kodu yazmaları için 20 dakika süre verir ve bu sürede odaları gezerek süreci takip eder. Süre sonunda eğitmen, gruplardan çözümlerini ekran görüntüsü alarak Padlet panosuna yüklemelerini ister. 1-2 gönüllü grup, kendi çözümünü Padlet üzerinden açarak tüm sınıfa sunar (10 dk).

Eğitmene Öneriler: Bu görevde iç içe döngü mantığı kritiktir. Eğitmen, gruplara şu şekilde rehberlik edebilir: "Hangi döngü öğrencileri gezecek, hangi döngü o öğrencinin notlarını gezecek?", "Her öğrencinin not toplamını nerede sıfırlamalısınız?". Süreci yönetmek için:

- **Kritik Noktayı Vurgulayın:** Öğrencilere toplam değişkenini her ders için döngü içinde sıfırlamanın önemini, "Bir dersin toplamı bitince diğerine sıfırdan başlamaz mıyız?" gibi sorularla hatırlatın.
- **Basitleştirin:** Zorlanan gruplara problemi daha küçük parçalara ayırmaları için yol gösterin (örneğin, önce sadece tek bir dersin ortalamasını bulmalarını isteyin).

Aşağıda, görevin adımlarını eksiksiz olarak yerine getiren, açıklamalı C++ kodu bulunmaktadır. Bu kod, grupların ulaşması beklenen nihai hedeftir.

```
#include <iostream>
#include <string> // Ders isimlerini saklamak icin

using namespace std;

int main() {
    // Ders isimlerini ayri bir dizide tutmak kodu daha esnek hale
getirir.
    string dersler[] = {"Matematik", "Fizik", "Edebiyat"};

    int notlar[3][4] = {
        {85, 90, 78, 92},  // Matematik Dersinin notlari
        {76, 88, 80, 85},  // Fizik Dersinin notlari
        {95, 91, 93, 89}   // Edebiyat Dersinin notlari
    };

    // --- DIS DONGU: Dersler (Satirlar) arasinda gezer ---
    // i = 0 iken Matematik, i = 1 iken Fizik, i = 2 iken Edebiyat
notlarini isler.
    for (int i = 0; i < 3; i++) {

        float toplam = 0; // KRITIK ADIM: Her yeni derse baslarken
toplami sifirliyoruz.

        // --- IC DONGU: O derse ait notlar (Sutunlar) arasinda gezer ---
        for (int j = 0; j < 4; j++) {
            toplam += notlar[i][j]; // O dersin notlarini toplama ekle.
        }

        // Ic dongu bittiginde, o dersin toplami hazir olur. Simdi
ortalamayi hesapla.
        float ortalama = toplam / 4.0; // Ondalikli sonuc icin 4.0'a
boluyoruz.

        // Sonucu ekrana yazdir.
        cout << dersler[i] << " Dersinin not ortalamasi: " << ortalama <<
endl;
    }

    return 0;
}
```

Hafta 4. Çevrim İçi: Ders Materyalleri

LÇ4.B1.1. Kod Çıktısı Ne Olur?

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

int main() {
    // Onceden tanimlanmis bir sayi dizisi
    int sayilar[] = {17, 29, 44, 65, 78, 40, 53, 12};

    // Aranacak sayiyi sabit olarak belirliyoruz
    int aranan = 78;

    // Bu basit etkinlikte, ogrencilerin zihinsel olarak takip etmesini
kolaylastirmak icin
    // dizinin boyutu ve bulunamama durumu gibi ekstra kontroller
cikarilmistir.
    // Amac, dongu ve if yapisinin dizi uzerindeki etkisini net bir
sekilde gormektir.

    // Dongu ile dizideki elemanlari bastan sona kontrol et
    for (int i = 0; i < 8; ++i) {
        // Eger aranan sayi, dizinin o anki elemanina esitse
        if (sayilar[i] == aranan) {
            // Ekrana sayinin bulundugunu ve indisini yazdir
            cout << aranan << " bulundu! Konumu (indeksi): " << i <<
endl;
        }
    }
    return 0;
}
```

LÇ4.B2.1. Görev Tanımı

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

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

    // En yuksek puani saklamak icin bir degisken olusturuyoruz.
    // Baslangic degeri olarak ilk oyuncunun puanini en yuksek kabul
edelim.
    int enYuksekPuan = puanlar[0];

    // Diger oyuncularin puanlarini kontrol etmek icin bir dongu kuralim.
    // Ilk oyuncuyu zaten en yuksek olarak atadigimiz icin donguye 1.
indisten baslayabiliriz.
    for (int i = 1; i < oyuncuSayisi; ++i) {

        // GOREV: Bu kisma o anki oyuncunun puani (puanlar[i]) ile
        // enYuksekPuan degiskenini karsilastirip, gerekirse enYuksekPuan
        // degiskenini guncelleyecek 'if' kosulunu yaziniz.
        // --- KODUNUZU BURAYA YAZIN ---

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

    }

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

    return 0;
}
```

LÇ4.B3.1. Görev Kartı: Kullanıcı Adı Oluşturma

Senaryo: Yeni bir oyun platformuna kaydoluyorsunuz ve kendinize havalı bir kullanıcı adı oluşturmanız gerekiyor. Sistem, kullanıcı adınızın isminiz ve doğum yılınızın birleşiminden oluşmasını istiyor.

Görev Adımları:

1. Kullanıcıdan ismini (string olarak) ve doğum yılını (string olarak) alan bir C++ programı yazın.
2. Bu iki bilgiyi birleştirerek kullaniciAdi adında yeni bir string değişkeninde saklayın. (Örnek: İsim "Zeynep", yıl "2007" ise kullaniciAdi "Zeynep2007" olmalıdır.)
3. Oluşturduğunuz kullanıcı adını ekrana yazdırın.

Teslimat: Tamamladığınız C++ kodunu ve tartışma sorusuna verdiğiniz cevabı Padlet panosunda oda numaranızla aynı olan sütun altına iletiniz.

LÇ4.B4.1. Final Projesi: Not Ortalaması Hesaplayıcı

Senaryo:
Üç farklı dersin 4 farklı zamanda yapılan sınavlarından aldığınız notlar aşağıdaki gibi bir tabloda tutuluyor. Her bir dersin sınavlarındaki not ortalamasını hesaplayıp ekrana yazdırmak istiyorsunuz.

Verilen Notlar (2 Boyutlu Dizi):

```
int notlar[3][4] = {
   {85, 90, 78, 92},  // Matematik Dersinin notlari
   {76, 88, 80, 85},  // Fizik Dersinin notlari
   {95, 91, 93, 89}   // Edebiyat Dersinin notlari
};
```

Göreviniz:
Bu notlar dizisini kullanarak, her dersin not ortalamasını hesaplayan ve aşağıdaki gibi ekrana yazdıran bir C++ programı yazınız.

Beklenen Çıktı:
Matematik Dersinin not ortalamasi: 86.25
Fizik Dersinin not ortalamasi: 82.25
Edebiyat Dersinin not ortalamasi: 92

İpuçları:

- Dersler arasında gezinmek için bir dış döngüye ihtiyacınız olacak.
- Her dersin notları arasında gezinmek için bir iç döngüye ihtiyacınız olacak.
- Her dersin not toplamını tutmak için bir değişkene ihtiyacınız olacak. Bu değişkeni nerede sıfırlamanız gerektiğini iyi düşünün!
- Ortalama hesaplarken ondalıklı sonuç alabilmek için float veya double tür dönüşümü yapmanız gerekebilir.

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: L5.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: L5.B2.1. C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri

L5.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 "L5.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 (L5.B2.2) öğrencilerle ÖYS üzerinden paylaşılır. Öğrencilerin ihtiyaç duyduğu aşamada eğitmenler konu ile ilgili yöntemsel bilgiyi öğrenciye verir.

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

C++ programlama dilinde 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: L5.B3.1. Fonksiyonları Kullanarak Kodlama Yapma

L5.B3.2. Fonksiyon Kodlama ile İlgili Destekleyici Bilgi Sunusu

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

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

Eğitmene Öneriler:

Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

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

Cevap 1:

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

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

Şimdi bunu sadece fonksiyonun ismini yazarak ana programdan çağıralım:

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

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

Cevap 2:

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

```
}
```

Fonksiyonumuz hazır hâle geldi. Artık kullanıma hazır, üçgen alanı hesaplayıp ekrana yazdıran bir fonksiyona sahibiz. Programın istediğimiz yerinde çağırıp kullanabiliriz. Tabanı 2 ve yüksekliği 4 olan bir üçgen için alan aşağıdaki gibi hesaplanır. parametreleri doğrudan sayı olarak girebildiğimiz gibi değişken ismi de girebiliriz.

```
int main()
{
    ucgen_alan_hesapla(2,4);
}
```

Görev 3: Fonksiyona gönderilen tam sayı tipindeki dizinin en büyük sayısını ekrana yazan fonksiyonu yazalım.

Cevap 3:

```
void en_buyuk(int dizi[5])
{
    int enbuyuk = dizi[0];
    for(int i=1;i<5;i++)
    {
        if(enbuyuk<dizi[i])
            enbuyuk=dizi[i];
    }
    cout << enbuyuk;
}
```

Ana programımız ise aşağıdaki şekilde olacaktır.

```
int main()
{
    int sayilar[] = {5,3,4,5,8};
    en_buyuk(sayilar);
}
```

Görev 4: İki dizi içerisindeki en büyük iki sayının toplamını bulan fonksiyonu yazalım.

Cevap 4:

Önceki örneği sadece 1 düzenleme ile kullanabiliriz. Ekrana yazdırmak yerine en büyük sayıyı ana programa göndermemiz gerekiyor. Böylece sonraki işlemlerde kullanabiliriz.

```
int en_buyuk(int dizi[5])
{
    int enbuyuk = dizi[0];
    for(int i=1;i<5;i++)
    {
        if(enbuyuk<dizi[i])
            enbuyuk=dizi[i];
    }
    return enbuyuk;
}
```

Ana programı da aşağıdaki gibi yazabiliriz. İlk olarak dizilerimizi tanımlıyoruz. Ardından dizi1'i fonksiyona gönderip en büyüğünü buluyoruz. Sonra dizi2'nin en büyük elemanını bulup ekrana yazdırıyoruz. Son olarak yapmak istediğimiz toplama işlemini gerçekleştiriyoruz.

```
int main()
{
    int dizi1[] = {5,3,4,5,8};
    int dizi2[] = {9,3,4,5,8};

    int en_buyuk_1 = en_buyuk(dizi1);
    cout << "1. dizinin en buyugu:" << en_buyuk_1 << endl;
    int en_buyuk_2 = en_buyuk(dizi2);
    cout << "2. dizinin en buyugu:" << en_buyuk_2 << endl;
    cout << en_buyuk_1 << "+" << en_buyuk_2 << "=" << en_buyuk_1 + en_buyuk_2;
}
```

Görev 5: Bir bilgisayar programında, iki adet fonksiyon bulunmaktadır. İlk fonksiyonda, yaş bilgisi alınmakta ve fonksiyon içerisinde güncellenmektedir. Bu güncellenen değer ana program bloğunu etkilememektedir. İkinci fonksiyonda alınan yaş bilgisi fonksiyon içerisinde

güncellenmekte ve değişiklik ana programda etkili olmaktadır. Bunun için nasıl bir program yazmalıyız?

Cevap 5:

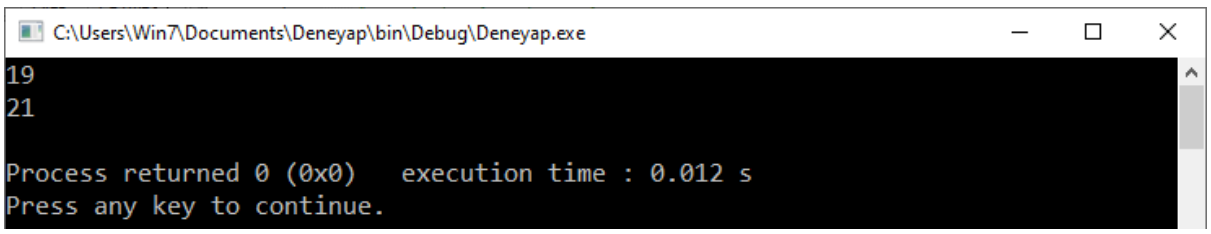
Referans gönderme:

Eğer fonksiyon içerisinde parametre olarak gönderilen değişkenin değeri değişecek ise, değişkenin değeri yerine adresini göndeririz. Böylece fonksiyon içerisindeki değişiklikler değişken üzerine yansıyacaktır. Aşağıdaki örnek bu konuyu açıklayacaktır.

Örnek: Değişken değeri değiştirme referanslı ve referanssız.

```
void fonksiyon1(int sayi)
{
    sayi = 20;
}
void fonksiyon2(int& sayi)
{
    sayi = 21;
}
int main()
{
   int yas = 19;
   fonksiyon1(yas);
   cout<<yas <<endl;

   fonksiyon2(yas);
   cout<<yas <<endl;
}
```

fonksiyon1'e yas isimli değişkenin değerini yani 19 değerini gönderiyoruz. fonksiyon içerisinde sayi isimli değişken üzerinde yapılan değişiklik ana programdaki "yas" isimli değişkeni etkilemeyecektir. Fakat fonksiyon2'de fonksiyona "&" işareti kullanarak değişkenin adresini gönderiyoruz. Dolayısıyla fonksiyon içerisindeki tüm değişiklikler ana programa yansıyacaktır. Sonuç olarak program çıktısı aşağıdaki gibi olacaktır. Bu sayede birden fazla değişkenin değerini fonksiyon içerisinde değiştirebiliriz.



Resim 20. *Ekran çıktısı*

B4. Gelişmiş Fonksiyon Örnekleri Kodluyorum

Süre: 50 dk.

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

K3. C++ programlama dilinde fonksiyon çağırmayı bilir.

Materyaller: L5.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 dört 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: Harita mühendisi olan Ali kendisine gönderilen arazinin kenarları bir tane gönderildi ise karenin, iki tane gönderildi ise dikdörtgenin çevresini bulduracak bir fonksiyon tanımlamak istemektedir. Sizce Ali nasıl bir kod yazmalıdır?

Aşırı Yükleme:

Bir fonksiyonu birden farklı şekilde kullanabilmek için fonksiyonları farklı şekilde tanımlayabiliriz. Örneğin: bir fonksiyon ile dikdörtgen/kare çevresini hesaplayalım. Eğer kullanıcı fonksiyona iki değişken gönderir ise bu bir dikdörtgen, tek değişken gönderirse kare olarak ele alır ve işlemleri buna göre yaparız.

Cevap 1:

```
#include <iostream>
using namespace std;
int cevre_hesapla(int a)
{
    int cevre = a*4;
    return cevre;
}
int cevre_hesapla(int a, int b)
{
    int cevre = 2*a + 2*b;
    return cevre;
}
int main()
{
   int cevre1 = cevre_hesapla(5);
   int cevre2 = cevre_hesapla(5,4);

   cout << cevre1 <<endl;
   cout << cevre2 <<endl;
}
```

Görev 2: Matematik Öğretmeni Hasan, öğrencilerine gösterebilmek adına ekrana girilen sayının faktöriyelini bulduran bir program oluşturmak istemektedir. Bunun için nasıl bir kod yazmalıdır?

Cevap 2:

```
#include <iostream>
using namespace std;
int faktoriyel(int sayi)
{
    if(sayi == 1)
        return 1;
    else
        return sayi*faktoriyel(sayi-1);
}
int main()
{
   int sonuc = faktoriyel(5);
   cout << sonuc <<endl;
}
```

Görev 3: Hüma, kendisine verilen 3x3 boyutundaki bir matrisin (iki boyutlu dizi) 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_toplam(int matris[3][3])
{
    int toplam = 0;

    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
        {
            toplam += matris[i][j];
        }
    }

    return toplam;
}

int main()
{
    int matris[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int sonuc = matris_toplam(matris);

    cout << "Matrisin elemanlari toplami: " << sonuc << endl;
```

```
    return 0;
}
```

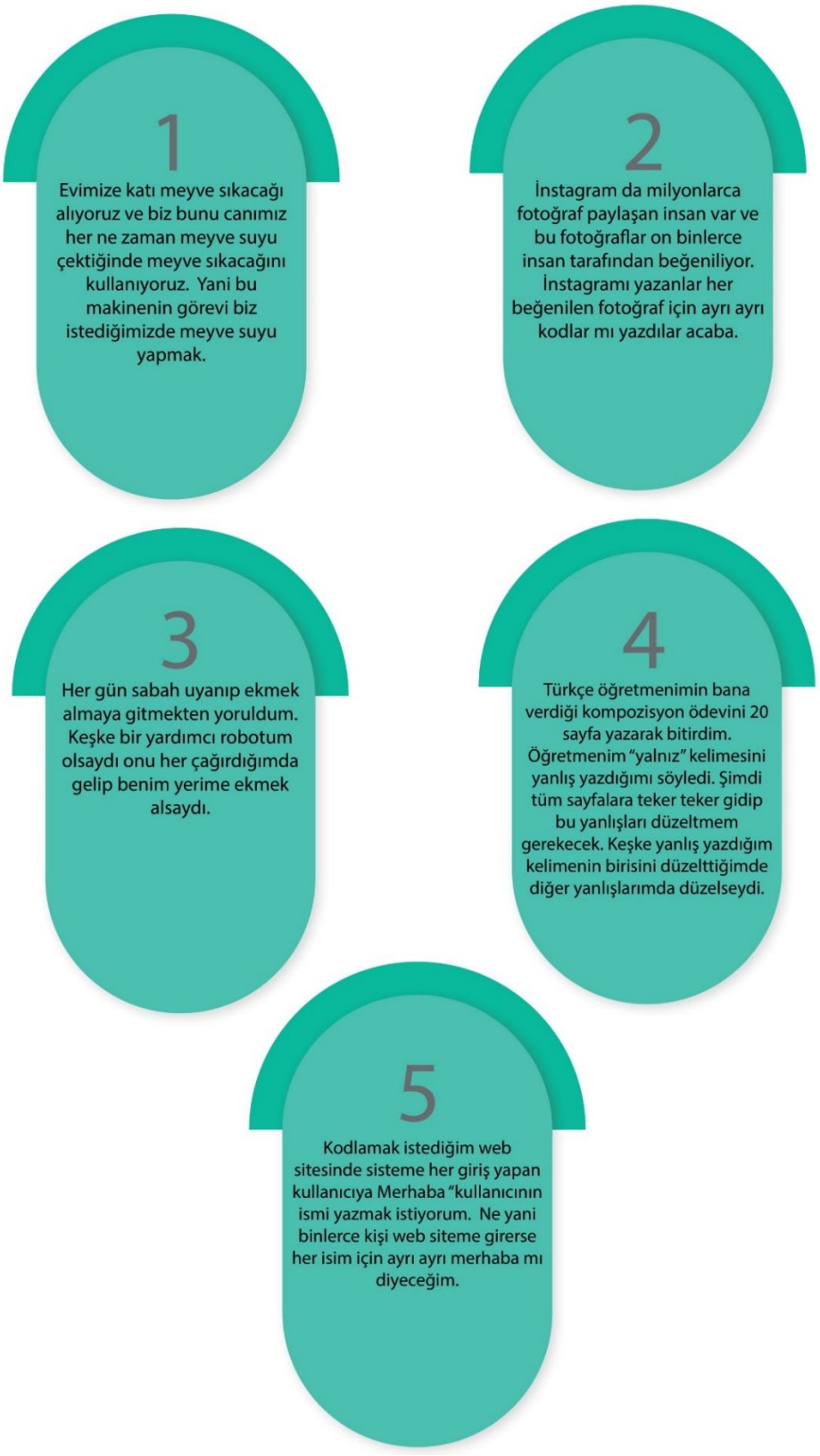
Görev 4: Makine mühendisliği öğrencisi Ahmet, belirli bir ürün için üretim maliyetlerini hesaplayan bir program yazmak istemektedir. Toplam maliyet, üretim adedi ile birim maliyetin çarpımıyla bulunacaktır. Ancak, birim maliyete vergi de eklenmelidir. Bu nedenle Ahmet, önce vergi dahil birim maliyeti hesaplayan bir fonksiyon, ardından bu sonucu kullanarak toplam maliyeti hesaplayan başka bir fonksiyon yazmalıdır. Ahmet'in bu işlemi gerçekleştirmesi için C++ programı nasıl olmalıdır?

Cevap 4:

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

float birim_maliyet_hesapla(float birim_fiyat, float vergi_orani) {
    return birim_fiyat + (birim_fiyat * vergi_orani / 100);
}

float toplam_maliyet_hesapla(int adet, float birim_fiyat, float
vergi_orani) {
    float vergili_fiyat = birim_maliyet_hesapla(birim_fiyat, vergi_orani);
    return adet * vergili_fiyat;
}

int main() {
    int adet = 100;
    float birim_fiyat = 50.0;
    float vergi_orani = 18.0;

    float toplam = toplam_maliyet_hesapla(adet, birim_fiyat, vergi_orani);

    cout << "Vergi dahil birim fiyat: "
         << birim_maliyet_hesapla(birim_fiyat, vergi_orani) << " TL" <<
endl;
    cout << "Toplam maliyet: " << toplam << " TL" << endl;

    return 0;
}
```

Hafta 5. Yüz Yüze: Ders Materyalleri

L5.B1.1. Fonksiyonların Özelliklerini Keşfediyorum



L5.B2.1. C++ Programlama Dilinde Fonksiyon Tanımlama Görevleri

1) Instagram da işe başlayan bir yazılımcı, kullanıcıların fotoğraflara gönderdikleri yorumlara yönelik bir fonksiyon tanımlamak istiyor. Bu fonksiyonu siz olsaydınız nasıl tanımlardınız?

2) Dikdörtgen şeklinde olan büyük bir arazi üçgensel bölgelere ayrılmak istenmektedir. Bunun içinde araziye ne kadar üçgen sığabileceğini bulmak isteyen bir yazılımcı ihtiyaç duyduğu anda çağırabileceği üçgen alanının hesaplanmasına yönelik bir fonksiyon tanımlamak istemektedir. Yazılımcı bu alan fonksiyonunu nasıl tanımlamalıdır?

L5.B2.2. Fonksiyonların Kullanım Afişi

Resim 21. *Fonksiyon tanımlama afişi*

L5.B3.1. Fonksiyonları Kullanarak Kodlama Yapma

L5.B3.2. Fonksiyon Kodlama ile İlgili Destekleyici Bilgi Sunusu

Yukarıda verilen görevlerin her biri için destekleyici bilgi hazırlanmıştır. Bu destekleyici bilgilere ulaşmak için L5.B3.2 adlı sunumu açınız. Sunumun çıktısı alınarak her öğrenciye bu çıktılar dağıtılabilir veya öğrencinin kendi bilgisayarına yüklenmesi sağlanabilir.

L5.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 Programın Çıktısını Tahmin Etme (20 dk.)

B2. İki Sayı Arasındaki İlişkiyi Kalan Hesabı ile İnceleyelim (25 dk.)

Ders Arası (10 dk.)

B3. Dizi İçindeki Matematik Notlarının Toplamını Bulma (25 dk.)

B4. Girilen Sayının Asal Olup Olmadığını Fonksiyon ile Kontrol Etme (25 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 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: LÇ5.B1.1. Öğrenci Yönerge Metni

LÇ5.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 LÇ5.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. İki Sayı Arasındaki İlişkiyi Kalan Hesabı ile İnceleyelim

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: LÇ5.B2.1. Öğrenci Yönerge Metni

LÇ5.B2.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen LÇ5.B2.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Parametre olarak gönderilen iki sayının büyüğünün küçüğüne bölümünden kalanı geriye döndüren bir fonksiyon yazınız.

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. Dizi İçindeki Matematik Notlarının Toplamını Bulma

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: LÇ5.B3.1. Öğrenci Yönerge Metni

LÇ5.B3.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen LÇ5.B3.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Caner parametre olarak gönderilen dizi içerisindeki sıfırdan büyük öğrencilerin aldığı matematik notlarını toplamını bulup geriye döndüren bir fonksiyon yazmak istemektedir. Bunun için nasıl bir fonksiyon yazmalıdır?

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. Girilen Sayının Asal Olup Olmadığını Fonksiyon ile Kontrol Etme

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: LÇ5.B4.1. Öğrenci Yönerge Metni

LÇ5.B4.2. Kod Bloğu

Uygulama: Öğrenciler, belirtilen LÇ5.B4.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Ahmet yazacağı kodda, kullanıcı tarafından girilecek sayının kendisinden başka bir sayıya bölünüp bölünmediğini bulmak istemektedir. Diğer bir ifadeyle sayının asal sayı olup olmadığını bulmak istemektedir. Bunun için nasıl bir fonksiyon yazmalıdır? (Asal ise 1 değilse 0 döndürsün.)

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

LÇ5.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?

LÇ5.B1.2. Yanıt

Cevap: 5

LÇ5.B2.1. Öğrenci Yönerge Metni

Göreviniz: Parametre olarak gönderilen iki sayının büyüğünün küçüğüne bölümünden kalanı geriye döndüren bir fonksiyon yazınız.

LÇ5.B2.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int kalan_bul(int sayi1, int sayi2)
{
if(sayi1 > sayi2)
        return sayi1 % sayi2;
    else
        return sayi2 % sayi1;
}
int main()
{
    int sonuc = kalan_bul(9,12);
    cout << sonuc;
}
```

LÇ5.B3.1. Öğrenci Yönerge Metni

Göreviniz: Duru parametre olarak gönderilen dizi içerisindeki öğrencilerin aldığı sıfırdan büyük matematik notlarını toplamını bulup geriye döndüren bir fonksiyon yazmak istemektedir. Bunun için nasıl bir fonksiyon yazmalıdır?

LÇ5.B3.2. Kod Bloğu

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

LÇ5.B4.1. Öğrenci Yönerge Metni

Göreviniz: Ahmet yazacağı kodda, kullanıcı tarafından girilecek sayının kendisinden başka bir sayıya bölünüp bölünmediğini bulmak istemektedir. Diğer bir ifadeyle sayının asal sayı olup olmadığını bulmak istemektedir. Bunun için nasıl bir fonksiyon yazmalıdır? (asal ise 1 değilse 0 döndürsün.)

LÇ5.B4.2. Kod Bloğu

```
#include <iostream>
using namespace std;
int asal_sayi_mi(int sayi)
{
    for (int i=2;i<sayi;i++)
        if (sayi% i == 0)
            return 0;
    return 1;
}
int main()
{
    int sonuc = asal_sayi_mi(23);
    cout << sonuc;
}
```

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, sınıf ve erişim belirteci kavramlarını ayırt eder.
- K4. Günlük hayatta karşılaştığı problemlerle ilgili fonksiyon oluşturma işlemini gerçekleştirir.
- K5. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.
- K6. Yapıcı ve yakıcı fonksiyonları kullanarak program geliştirir.
- K7. Veri soyutlama, kapsülleme, kalıtım ve polimorfizm kavramlarını açıklar.

Amaç

Bu haftanın amacı, nesne yönelimli programlamanın temel felsefesini, prosedürel programlamadan farkını, avantaj ve dezavantajları ile nesne yönelimli programlamada sınıf, nesne ve fonksiyon tanımlama işlemlerini uygulamaktır. Yapıcı ve yıkıcı ve fonksiyonları kullanmak, sınıf dosyaları ile büyük boyutlu projeler hazırlamayı anlamak hedeflenir. Ayrıca, veri soyutlama, veri kapsülleme, kalıtım, polimorfizm, aşırı yükleme ve geçersiz kılma kavramlarını öğrenmek amaçlanır.

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. Yarış Benimle (25 dk.)

Ders Arası (10 dk.)

A4. Kod Satırlarını Tamamla (50 dk.)

Ders Arası (10 dk.)

A5. Adam Asmaca (50 dk.)

A. Öğrenme Görevleri

A1. Nesneni Çiz

Süre: 25 dk.

Kazanımlar: K1. C++ programlama dilinde nesne yönelimli programlama mantığını açıklar.

K2. Nesne yönelimli programlamayı prosedürel programlamadan 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.

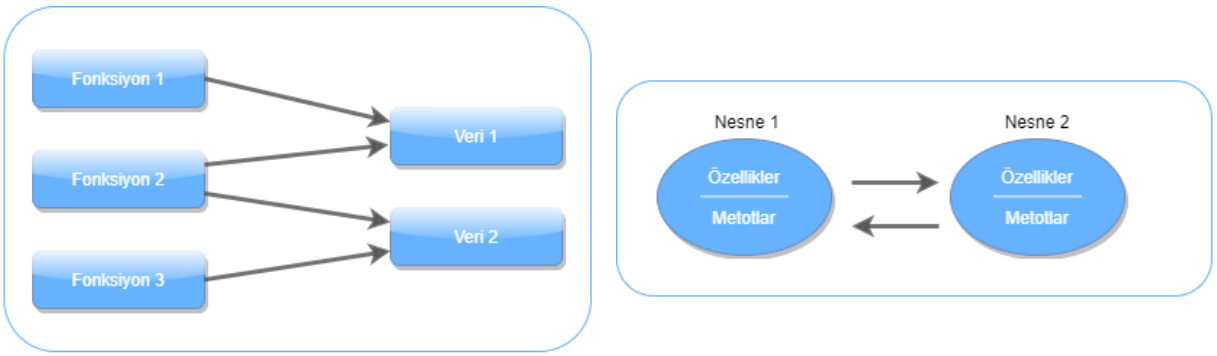
Konu İçeriği: C++ programlama dilinin amacı, kendi başına en güçlü programlama dillerinden biri olan C programlama diline nesne yönelimi eklemekti. Nesne yönelimli programlama, programları nesnelerin sınıfları açısından tanımlayan bir yazılım tasarım ve programlama yöntemidir. Aşağıdaki şekilde görüldüğü üzere yordamsal (prosedürel) programlama dillerinde veri ve fonksiyonlar birbirinden ayrı iken, nesne yönelimli programlama, veri ve fonksiyonları birleştirerek aralarındaki görevleri gerçekleştirmek için düzenli iletişim sağlayan nesneler kümesi şeklindedir.

Prosedürel Programlama	Nesne Yönelimli Programlama
Program, fonksiyon adı verilen küçük parçalara bölünmüştür.	Program, nesneler adı verilen küçük parçalara ayrılır.
Yeni veri ve fonksiyon eklemek kolay değildir.	Yeni veri ve fonksiyon eklemek koladır.
Verileri gizlemek için uygun bir yol yoktur, bu nedenle daha az güvenlidir.	Daha güvenli olmak için veri gizleme sağlar.
Fonksiyon veriden daha önemlidir.	Veri, fonksiyondan daha önemlidir.

Gerçek olmayan dünyaya dayanır.	Gerçek dünyaya dayanır.
Örnekler: C, Fortran, Pascal, Basic vb.	Örnekler: C++, Java, Python, C# vb.

Prosedürel programlama, bir programı, adım adım izlenen işlemler dizisi olarak yapılandıran geleneksel bir programlama yaklaşımıdır. Bu yöntem, genellikle veriler ile bu veriler üzerinde işlem yapan fonksiyonları (veya prosedürleri) ayrı ayrı ele alır. Yani, veri yapıları ve bu verilere uygulanan işlemler birbirinden bağımsızdır. Bir problem çözülürken, öncelikle veriler tanımlanır, ardından bu veriler üzerinde işlem yapacak fonksiyonlar sıralı şekilde yazılır.

Örneğin, bir müşteri kaydı tutan bir programda, müşteri bilgilerini saklayan bir veri yapısı oluşturulur, ardından bu bilgilere erişim sağlayan, güncelleyen veya silen ayrı fonksiyonlar tanımlanır. Bu yapı, küçük ölçekli uygulamalarda yeterli olabilir, ancak sistem karmaşıklaştıkça, veriler ile fonksiyonlar arasındaki ilişkiyi yönetmek zorlaşabilir.



Resim 22. *Yordamsal ve nesne yönelimli programlama*

Nesne yönelimli programlama ise, verileri ve bu verilerle ilgili işlemleri bir araya getirerek daha organize ve modüler bir yapı sunar. Bu yaklaşımda hem veriler (özellikler) hem de bu veriler üzerinde işlem yapan fonksiyonlar (yöntemler), nesne adı verilen yapılar içinde toplanır. Bu nesneler, sınıflar aracılığıyla tanımlanır ve birbirleriyle mesajlaşarak (metot çağrılarıyla) iletişim kurar. Yani bir program, bağımsız işlevlerden oluşmak yerine, birbiriyle etkileşim içinde olan nesnelerin oluşturduğu bir sistem olarak tasarlanır. Her nesne, kendi durumunu (veri) korur ve sadece kendi metodları aracılığıyla bu verileri işler. Bu yapı sayesinde program daha okunabilir, sürdürülebilir ve genişletilebilir hale gelir.

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: bisiklet, sandalye, kitap, top ve cep telefonu gibi.

Resim 23. *Nesne örnekleri*

Nesne bir sınıf 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. Telefon bir sınıf ise bundan üretilen farklı telefon çeşitleri de nesneleri ifade eder.

Iphone Huawei Samsung

Resim 24. *Sınıf ve nesne kavramı*

Nesne Yönelimli Programlamanın Avantaj ve Dezavantajları

Nesne yönelimli programlama, modern yazılım geliştirme sürecinde en çok tercih edilen yaklaşımlardan biridir. Bu programlama yöntemi, yalnızca kod yazmayı kolaylaştırmakla kalmaz, aynı zamanda yazılan kodun bakımını, yeniden kullanılmasını ve genişletilmesini de büyük ölçüde kolaylaştırır. Nesne yönelimli programlamanın kullanılmasının birçok güçlü nedeni vardır:

1. Kodun Yeniden Kullanımı: Nesne yönelimli programlamanın en güçlü özelliklerinden biri olan kalıtım, bir sınıfın başka bir sınıftan özellikleri ve davranışları miras almasını sağlar. Bu sayede:

- Yeni sınıflar tanımlarken sıfırdan başlamak gerekmez.
- Mevcut bir sınıfın kodunu kullanarak yeni sınıflar türetilebilir.
- Yazılımda tekrarı azaltır, tutarlılığı artırır.

2. Kod Bakımı ve Geliştirme Kolaylığı: Nesne yönelimli programlama sayesinde kod daha düzenli, modüler ve anlaşılır bir yapıya kavuşur. Özellikle büyük projelerde, iyi yapılandırılmış sınıflar ve nesneler sayesinde:

- Mevcut kodu değiştirmek daha güvenli ve kolay hale gelir.
- Kodun belirli bölümleri izole edildiği için bir bölümde yapılan değişiklik, diğer bölümleri etkilemez.
- Hataların bulunması ve düzeltilmesi daha kolay olur.

Bu yapı, projeyi devralan başka bir geliştiricinin kodu hızlıca anlamasına da olanak tanır. İyi tasarlanmış ve yorumlanmış sınıflar, yazılımcıya sistemin nasıl çalıştığını ve hangi amaca hizmet ettiğini açıkça gösterir.

3. Takım Çalışmasında Verimlilik ve Anlaşılabilirlik: Nesne yönelimli programlama, büyük yazılım ekiplerinde çalışma kolaylığı sağlar:

- Her geliştirici, sistemin farklı bir parçası üzerinde bağımsız olarak çalışabilir.
- Nesneler arasında tanımlı iletişim sayesinde, her nesne kendi sorumluluğunda olan işlevleri yerine getirir.
- Bu da ekip içinde daha iyi iş bölümü, daha az çatışma ve daha hızlı entegrasyon anlamına gelir.

Yeni geliştiriciler, projenin yapısını sınıflar üzerinden analiz ederek kodun akışını ve sistemin mantığını kısa sürede kavrayabilir.

4. Modülerlik ve Genişletilebilirlik: Nesne yönelimli programlama ile geliştirilen sistemler modülerdir. Yani her sınıf ya da nesne belirli bir görevi yerine getirir. Bu da:

- Gerektiğinde yeni özelliklerin kolayca eklenmesini sağlar.
- Mevcut modüllerin diğerlerini bozmadan güncellenmesine olanak tanır.
- Projenin uzun vadede sürdürülebilir olmasına katkıda bulunur.

5. Prosedürel Programlamaya Göre Avantajları: Nesne yönelimli programlamanın, geleneksel prosedürel programlamaya göre birçok avantajı vardır:

- Daha hızlı geliştirme ve daha kolay uygulanabilirlik ile kodlar daha organize ve modüler olduğu için geliştirme süreci hızlanır.
- Programlarda net bir yapı sağlayarak sınıf ve nesne tabanlı yaklaşım, kodun düzenli ve okunabilir olmasına yardımcı olur.
- Kalıtım ve nesne paylaşımı sayesinde aynı kodun tekrar tekrar yazılması gerekmez.

- Modüler yapı sayesinde, sadece ilgili nesnede değişiklik yaparak sorunlar çözülebilir.
- Bir kez tanımlanan sınıf ya da nesne, farklı projelerde ya da modüllerde tekrar kullanılabilirlik sağlanır.
- Daha az kod ve daha kısa geliştirme süresi sunarak verimli sınıf yapıları sayesinde karmaşık işler bile daha az kodla çözülebilir.

Nesne Yönelimli Programlamanın Temelleri

C++ programlamanın temel amacı, benzerleri arasında en güçlü ve en yaygın kullanılan programlama dillerinden biri olan C programlama diline nesne yönelimi eklemektir. Nesne yönelimli programlamanın özü, kod içerisinde belirli özelliklere ve yöntemlere sahip bir nesne oluşturmaktır. C++ modüllerini tasarlarken tüm dünyayı nesneler şeklinde tasarlamaya çalışıyoruz. Örneğin bir araba, renk, kapı sayısı ve benzeri gibi belirli özelliklere sahip bir nesnedir. Ayrıca hızlanma, frenleme gibi belirli yöntemlere de sahiptir. Nesne yönelimli programlamanın temelini oluşturan altı ana kavram vardır. Bunlardan ilk ikisi yukarıda detaylı olarak açıklanmıştı. Ama tekrar olması açısından hepsini listelersek:

1) Nesne (Object): Nesne yönelimli programlamanın temel birimidir. Veri ve veriler üzerinde çalışan fonksiyonlar, nesne olarak adlandırılan bir birim olarak paketlenmiştir.

2) Sınıf (Class): Nesne yönelimli programlamaya yapı taşıdır. Sınıfın bir örneğini oluşturarak erişilebilen ve kullanılabilen kendi veri üyelerini ve fonksiyonlarını tutan kullanıcı tanımlı bir veri türüdür.

3) Soyutlama (Abstraction): Dış dünyaya yalnızca gerekli bilgilerin sağlanması ve arka plan ayrıntılarının gizlenmesi, yani gerekli bilgilerin programda ayrıntıları verilmeden temsil edilmesidir.

4) Kapsülleme (Encapsulation): Veri ve bilgilerin tek bir birim altında toplanması olarak tanımlanır. Verileri ve onları işleyen fonksiyonları birbirine bağlamak olarak tanımlanır.

5) Miras (Inheritance): Bir sınıf oluşturulurken başka bir sınıftın özellikleri ve karakteristiklerinin türetilerek kullanılmasıdır. Türetilmiş sınıf olarak adlandırılan yeni bir sınıf oluşturulur.

6) Polimorfizm (Polymorphism): Birden fazla form anlamına gelir. Bir üye fonksiyonun onu çağıran nesneye göre farklı davranabilmesidir.

A2. Sınıfının Özelliklerini Tanı

Süre: 25 dk. + 25 dk.

Kazanımlar: K3. Nesne, sınıf ve erişim belirteci kavramlarını ayırt eder.

K4. Günlük hayatta karşılaştığı problemlerle ilgili fonksiyon oluşturma işlemini gerçekleştirir.

Materyaller: L6.A2.1. Grup Afişleri

Hazırlık: Materyallerin 2 adet çıktısı alınır ve her gruba ikişer adet aynı afişten dağıtılır.

Uygulama: Öğrenciler dörderli gruplar hâlinde çalışmaktadır. Eğitmen her gruba grup afişlerinden birini verir. Öğrencilerin gruplar hâlinde bu afişleri incelemeleri ve sınıf, üye listesi, erişim belirteci, fonksiyon ve nesne oluşturma konularını aralarında tartışmaları istenir. Gruplara 5 dk. inceleme yaptıktan sonra, grupların afişlerini ve bu terimleri sırayla diğer arkadaşlarına açıklamaları istenir. Bunun için birinci gruptan birer üye diğer gruplara dağılır. Birinci grubun üyeleri kendi afişlerini dahil olduğu yeni gruptakilere 5 dk. süre ile açıklar ve kendi grubuna geri döner. Benzer şekilde ikinci 5 dk. başlatılır. İkinci grubun her bir üyesi diğer gruplara dağılır ve afişlerini açıklar. İkinci grubun üyeleri geri döndükten sonra, üçüncü grup üyeleri dağılır. Bu şekilde dört grubun da sırası tamamlanır. Eğitmen zaman planlaması için bir çan kullanabilir ya da "Grup 1 Dağılın!", "Grup 1 Geri Dönün" gibi emir ifadeleri ile süreci yönetebilir.

Eğitmene Öneriler: Eğitmen grup etkinliklerinin tamamlanmasını ardından kalan dakikalarda öğrencilere sınıf tanımlama, erişim belirteçleri, üye listesi, fonksiyonlar ve nesne tanımlama ile ilgili sorular yöneltir. Bu şekilde eksik ya da hatalı öğrenmeleri düzenlemeye çalışır.

Konu İçeriği: Sınıf Tanımlama

Sınıflar, nesne yönelimli programlamanın en önemli yapı elemanıdır. Bir sınıf, bir nesnenin özelliklerini ve fonksiyonlarını tanımlar. Cep telefonuna ait bir sınıf oluşturmak istenildiğinde bu sınıfa ait özellikleri ve fonksiyonları belirlememiz gerekmektedir. Özellikler sınıfın ayırt edici bilgilerinden oluşan marka, model ve imei no gibi verileri içermektedir. Fonksiyonlar ise sınıfın arama, mesaj gönderme ve internete bağlanma gibi fonksiyonlarını içermektedir. Sınıfın tasarımı tamamen bizim elimizde olup istediğimiz olası tüm yetenekleri sınıfa ekleyebiliriz. Bir sınıfı tanımlamak için aşağıdaki gibi genel bir sözdizimi kullanılmaktadır.

```
class SınıfAdı
{
        üyeListesi
};
```

Burada **class**, sınıf tanımına ait anahtar kelime, **üyeListesi**, sınıf üyelerinin listesi ve **SınıfAdı**, sınıfın adıdır. Sınıf adının büyük harfle başladığına dikkat edelim. Bu en yaygın kullanımdır ve kodunuzun okunabilir olması için önem arz eder. Sınıf bildiriminin fonksiyonlardaki gibi parantezle başlayıp bittiğine dikkat ediniz. Ayrıca kapanış parantezinden sonra noktalı virgül kullanıldığını unutmayınız. Noktalı virgülün unutulması derleyici tarafından yakalanacak bir hataya sebep olacaktır.

UYARI:

Her sınıf ismini büyük harfle başlatın. Bu kural sadece C++ 'da değil, aynı zamanda diğer tüm nesne yönelimli programlama dillerinde de kullanılır.

üyeListesi, üye bildirimlerini içeren listeden oluşur. Bunlar veri üyesi veya fonksiyon bildirimleri olabilir. Veri üyesi bildirimleri normal değişken bildirimleri ile aynıdır. Örneğin, sırasıyla karakter, tam sayı ve kesirli sayı türlerinde veri üyeleri oluşturmak istersek aşağıdaki tanımlamaları yaparız.

```
int x;
float y;
char z;
```

Ancak, veri üyelerini bildirdiğiniz yerde ilk değer ataması gerçekleştiremezsiniz. Değişkenlere ilk değer atamasının hazırlanan bir fonksiyonda ya da sınıfın dışında gerçekleştirilmesi gerekir. Tanımlanan bu veri üyelerinin kapsamı, sınıftan oluşturulan nesnenin kapsamı ile aynıdır. Bununla birlikte, veri üyelerine sınıfın dışından her zaman erişilebilmesi mümkün değildir. Yukarıda verdiğimiz cep telefonu sınıfında markanın, modelin ve imei numarasının ne olduğu bu veri üyeleri tarafından saklanır. Bir sınıfın veri üyeleri, sınıf özelliklerini tanımlar ve açıklar. Bu nedenle, her bir cep telefonu nesnesinin markası, o nesnenin bir özelliğidir. Aşağıda cep telefonu sınıfına ait marka, model, imei numarası, renk ve fiyat veri üyelerinin sınıf içerisinde nasıl tanımlandığını görebilirsiniz.

```
class CepTelefonu
{
   char marka[30];
   char model[30];
   int imei_no;
   char renk[30];
   float fiyat;
};
```

Erişim Belirteçleri

Erişim belirteçleri, C++ sınıfınızın içerisinde yer alan veri üyelerine nereden erişilebileceğini kontrol etmenizi sağlar. Bu kontrol sayesinde veri üyeleri üzerinde yetkisiz olarak değişiklik yapılmasının önüne geçilmesi sağlanır. Erişim belirteci, sınıftaki veri üyelerine erişimi kontrol eden bir kelimedir. Bir erişim tanımlayıcısının sözdizimi aşağıdaki gibidir:

```
class SınıfAdı
{
        üyeListesi
erişimBelirteci:
        üyeListesi
```

erişimBelirteci:

üyeListesi

};

Bir erişim belirticisi tanımlandıktan sonraki üye listesindeki tüm veriler için geçerlidir. Sınıf içerisinde başka bir erişim belirticisi tanımlanırsa bu tanımlamadan sonraki veri üyeleri bu erişim türüne ait olur. Sınıfın sonuna ulaşıncaya kadar sınıfın tüm üyelerini en son tanımlanan erişim belirteci etkiler. Bir sınıfın üç farklı erişim belirleyicisi bulunmaktadır:

1) public: Bu anahtar kelime ile tanımlanan tüm üyelere, sınıfın ulaşılabilir olduğu her yerden erişilebilir.

2) private: Bu anahtar kelime ile tanımlanan üyelere yalnızca aynı sınıfın diğer üye fonksiyonları tarafından erişilebilir.

3) protected: Bu anahtar kelime ile tanımlanan üyelere yalnızca aynı sınıfın diğer üye fonksiyonları ve bu sınıftan türetilen sınıfların üye fonksiyonları tarafından erişilebilir.

UYARI:

Varsayılan olarak, tüm sınıf üyelerinin erişimi "private" olarak tanımlıdır. Bu nedenle, sınıf bildiriminde erişim belirticisi olmadan görünen tüm üyelerin erişimi "private" olur.

```
class Ceptelefonu
{
    char marka[30];
    int yil;
public:
    char model[30];
    char renk[30];
private:
    float fiyat;
protected:
    int imei_no;
};
```

Yukarıda verilen sınıf tanımlamasında fiyat veri üyesi ile birlikte marka ve yıl veri üyelerinin de "private" olduğunu unutmayınız. Diğer taraftan model ve renk "public" olarak tanımlı iken imei numarası ise "protected" olarak tanımlanmıştır. Sınıf tanımlamalarınızda istediğiniz kadar erişim belirteciniz olabilir, ancak belirteçleri tek bir grup altında toplamak sınıfın anlaşılırlığını artıracaktır.

UYARI:

Sınıflarınızı projeye eklemeden önce çalıştığından emin olmak için test edin. Bir sınıfın çalıştığını gözlemledikten sonra, sınıflar arasındaki etkileşimin doğru şekilde çalıştığından emin olmanız yeterlidir.

Fonksiyon Oluşturma

Bir sınıfın üye fonksiyonu, diğer herhangi bir değişken gibi sınıf içinde tanıma sahip olan bir fonksiyondur. Üyesi olduğu sınıfın herhangi bir nesnesi üzerinde çalışır ve bu nesne için bir sınıfın tüm üyelerine erişim sağlayabilir. Fonksiyon tanımları olmadan bir sınıfın tanımı tam olarak gerçekleştirilmiş sayılmaz. Fonksiyon tanımı yapıldıktan sonra ancak sınıfın nesneleri kullanılabilir. Tanımlanan bu fonksiyonlara yalnızca sınıfın bir nesnesi aracılığıyla erişilebilir. Bir fonksiyon tanımladığınızda, kapsam çözümleme operatörü "::" aracılığıyla fonksiyon adından ayıracak şekilde sınıf adını da eklemeliyiz. Sınıf adını eklemezsek global fonksiyon tanımı çağrılmaya çalışılır.

UYARI:

Tanımladığınız her bir fonksiyonun yalnızca bir iş yaptığından emin olun. Fonksiyon içerisinde çok fazla kod satırınız var ise, çok fazla şey yapmaya çalışıyorsunuzdur.

Bir fonksiyonu iki şekilde tanımlayabilirsiniz. Birinci yol, sınıf bildirimi içinde bir fonksiyon bildirmek ve sonra onu dışarıda uygulamaktır. İkinci yol, fonksiyonu aynı zamanda sınıf bildirimi içinde bildirmek ve uygulamaktır. Genelde uygulamalarda birinci yol tercih edilir. Bir sınıf dışında gerçekleşen bir fonksiyon tanımının genel sözdizimi şu şekildedir:

```
dönüş_tipi SınıfAdı::fonksiyon_adi(parametre_listesi)
{
        fonksiyon gövdesi
}
```

Gördüğünüz gibi, bu sözdizimi fonksiyon tanımı sözdizimine çok benzemektedir. Tanımlanan fonksiyon içinde, sınıfa ait tüm üyelere adları kullanılarak erişilebilir. Sınıf üyeliği otomatik olarak sağlanır ve aynı sınıfa ait fonksiyonlar birbirini doğrudan çağırabilir. "private" üyelere erişim yalnızca aynı sınıfa ait fonksiyonlar ile mümkündür. Böylece "private" üyeler tamamen sınıf tarafından kontrol edilir. Bir sınıf tanımladığınızda, sınıfa ait veri üyeleri için bellekten otomatik olarak yer ayrılmaz. Bellek ayırmak için bir nesne tanımlanması gerekir. Belirli bir nesne için bir fonksiyon çağrıldığında, ilgili fonksiyon bu nesnenin verilerini işleyebilir. Artık Ceptelefonu sınıfının arama ve mesaj gönderme fonksiyonlarını uygulayabiliriz.

Nesne Oluşturma

Nesne oluşturmak için öncelikle nesne için şablon olacak sınıf tasarımınızı gerçekleştirin. Sınıf tanımlamanızı gerçekleştirdikten sonra nesne değişkeni için bir tanımlayıcı oluşturun ve hangi sınıftan oluşacağını belirtin. Nesne oluşturmak için aşağıda verilen sözdizimlerinden birini kullanabilirsiniz:

```
SınıfAdı nesneDeğişkeni;
```

A3. Yarış Benimle

Süre: 25 dk.

Kazanımlar: K5. Yapıcı ve yıkıcı fonksiyon arasındaki farkı ayırt eder.

Materyaller: L6.A3.1. Yarışma Kartları

L6.A3.2. Kod Örneği

Hazırlık: Bir kura torbasında yarışma kartları bir adet çıktı alınır ve kesikli çizgilerden kesilerek kuraya hazırlanır. Kod örneği ÖYS üzerinden erişimine açılır ve tahtaya yansıtılır.

Uygulama: Eğitmen öğrencilerine bu öğrenme görevinde bir oyun oynayacaklarını söyler. Sınıfı iki gruba ayırır. Her gruba kod örnekleri verilir. Bu örneklerde hem yapıcı fonksiyon hem de yıkıcı fonksiyon kodlarının nasıl kullanıldığı bulunmaktadır. Oyunda gruplara sırayla bir kart çektirilir ve yarışma kartı gruba okunur. Bu kartlarda ise, yapıcı ya da yıkıcı fonksiyon arasındaki farkı anlatan bir özellik yazmaktadır. Grup çektikleri yarışma kartındaki özelliğin, ellerindeki kod örneğine bakarak doğru ya da yanlış bir ifade olup olmadığına karar verecektir. Doğru bilinen her bir karta 10 puan eklenir. Grup puanları sınıfta tahtaya yazılır. Bu şekilde en çok puan toplayan grup, diğer gruptan bir ödül alır. Eğitmen oyun sırasında yapıcı ve yıkıcı fonksiyonları detaylandırır. Oyun sonunda ise kısaca sabit fonksiyonlar hakkında da özetleyici bilgi geçer.

Eğitmene Öneriler: Eğitmen bu oyunu Web 2.0 teknolojilerinden yararlanarak hazırlayabilir. Oyun sonunda grup ödülüne öğrencilerle karar verilebilir ya da basit bir şekilde kazanan grubu alkışlamak gibi takdir hareketleri uygulanabilir. Her bir kartta eğitmen konu içeriğine bağlı olarak açıklamalarda bulunabilir.

Konu İçeriği: Yapıcı ve Yıkıcı Fonksiyonlar

Tanımladığınız sınıf içerisinde yapıcı (constructor) ve yıkıcı (deconstructor) olmak üzere iki özel fonksiyon türü olabilir. Her ikisi de isteğe bağlıdır ve diğer fonksiyonları sağlayamayacağı özel fonksiyonlar sağlarlar. Yapıcı fonksiyonlar sınıftan bir nesne oluşturulduğu zaman otomatik olarak çağrılır ve genellikle veri üyeleri için başlangıç değerlerini ayarlamak için kullanılır. Örneğin cep telefonu sınıfı için düşünecek olursak, yapıcı fonksiyon yıl veri üyesinin değerini sıfır yapabilir. Her zaman sınıfla aynı isme sahiptir ve int, void vb. gibi bir dönüş değerine sahip olamaz.

Yıkıcı fonksiyonlar ise, bir nesne yok edildiğinde otomatik olarak çağrılır ve gerekli tüm temizleme görevlerini gerçekleştirir. Yıkıcı fonksiyonlar, her zaman sınıfla aynı adla adlandırılır, ancak başında yaklaşık işareti (~) bulunur. Yıkıcı fonksiyonlar da bir dönüş değerine sahip olamaz. Hem yapıcı hem de yıkıcı fonksiyonlar diğer fonksiyonlar gibi tanımlanır ve uygulanır. Sınıf içinde eş zamanlı olarak tanımlamaları yapılacaksa aşağıdaki sözdizimi kullanılır.

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

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

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

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

Yapıcı metodun parametre alabileceğine dikkat ediniz. Parametre alabilen bir yapıcı fonksiyon oluştururseniz, 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.

A4. Kod Satırlarını Tamamla

Süre: 50 dk.

Kazanımlar: K6. Yapıcı ve yakıcı fonksiyonları kullanarak program geliştirir.

Materyaller: L6.A4.1. Grup Görev Kartları

Hazırlık: Materyalin bir adet çıktısı alınır.

Uygulama: Öğrenciler beşerli gruplar hâlinde çalışmaktadır. Bu etkinlikte eğitmen istasyon tekniği kullanmaktadır. Bunun için sınıfta da dört farklı grup masası oluşturulur. Her masada bir görev kartı bulunmaktadır. Gruplar 5 dk. içinde grup görev kartını bilgisayarda kodlayarak tamamlamaya çalışır. Süre bitiminde grupların saat yönünde bir sonraki grup masasına geçmesi istenir. Diğer grubun kaldığı noktadan grup masasındaki görev kartı tamamlanmaya çalışılır. Bu şekilde tüm gruplar her bir grup masasına geçişini tamamladıktan sonra dönme işlemi bitirilir. Dönme işleminin sonunda eğitmen sırayla ilk grubun görevini ve masada oluşan çözümü arkadaşlarına açıklamalarını ister. Varsa hatalı noktalar giderilir. Bu şekilde tüm grup masalarına hak verilir ve tüm kodlardaki görevin amacı eğitmen tarafından açıklanır.

Eğitmene Öneriler: Eğitmen grup yanıtlarını sadece bir arkadaşın aktarması için gruplara bir moderatör belirlemelerini isteyebilir. Bu şekilde bir ya da iki arkadaşın bilgisayar başında diğer arkadaşların da katkısıyla kodu yazmaları hızlandırılmış olur. Grup oluşturma aşamasında bu moderatörlerin seçilmesi sağlanmalıdır. Diğer bir nokta ise, tüm grupların son kod satırlarını padlet.com gibi bir dijital tartışma panosuna göndermeleri istenebilir. Bu şekilde grupların yaptıklarının tüm öğrenciler tarafından erişilebilir olması sağlanır.

Konu İçeriği: Grup görevlerinin amaçlarını ve yanıtlarını aşağıda bulabilirsiniz.

Grup 1: Basit bir Araba sınıfı oluşturarak, birden fazla nesne tanımlaması yapmak.

```
#include <iostream>
#include <cstring>
using namespace std;
// Bazi niteliklere sahip bir Araba sinifi olusturun
class Araba {
public:
    char marka[30];
    char model[30];
```

```
    int yil;
    int fiyat;
};
int main() {
    // Ilk Araba nesnesini olusturun
    Araba araba1;
    strcpy(araba1.marka, "Suzuki");
    strcpy(araba1.model, "Vitara");
    araba1.yil = 2016;
    araba1.fiyat = 225000;
    // Ikinci Araba nesnesini olusturun
    Araba araba2;
    strcpy(araba2.marka, "Volkswagen");
    strcpy(araba2.model, "T-Roc");
    araba2.yil = 2020;
    araba2.fiyat = 360000;
    // Nesnelerin ozelliklerini yazdirin
    cout << "Araba 1: " << araba1.marka << ", " << araba1.model << ", " << araba1.yil
<< ", " << araba1.fiyat << " \n";
    cout << "Araba 2: " << araba2.marka << ", " << araba2.model << ", " << araba2.yil
<< ", " << araba2.fiyat << " \n";
    return 0;
}
```

Kodun Çıktısı:

```
Araba 1: Suzuki, Vitara, 2016, 225000
Araba 2: Volkswagen, T-Roc, 2020, 360000
```

Grup 2: Araba sınıfına bir fonksiyon ekleme ve nesne tanımlama.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    char marka[30];
    char model[30];
```

```
    int hiz(int max_hiz);
};

int Araba::hiz(int max_hiz) {
  return max_hiz;
}

int main() {
    Araba araba1; // Araba nesnesini olusturun
    strcpy(araba1.marka, "Opel");
    strcpy(araba1.model, "Astra");
    cout << "Araba: " << araba1.marka << " " << araba1.model << " \n";
    cout << "Hiz: " << araba1.hiz(190); // Metodu parametre ile cagirin
    return 0;
}
```

Kodun Çıktısı:

```
Araba: Opel Astra
Hiz: 190
```

Grup 3: Araba sınıfına sınıf içi bir yapıcı fonksiyon ekleme ve nesne tanımlama.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    //Parametreli sinif ici yapıcı fonksiyon
    Araba(string x_marka, string x_model, int x_yil, int x_fiyat) {
        marka = x_marka;
        model = x_model;
        yil = x_yil;
        fiyat = x_fiyat;
    }
};
```

```
int main() {
    // Yapıcı metodu farkli degerlerle cagirarak Araba nesneleri olusturma
    Araba araba1("Suzuki", "Vitara", 2016, 225000);
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);

    // Degerleri yazdirma
    cout << araba1.marka << " " << araba1.model << " " << araba1.yil << " " << araba1.fiyat << "\n";
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil << " " << araba2.fiyat << "\n";
    return 0;
}
```

Kodun Çıktısı:

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

A5. Adam Asmaca

Süre: 50 dk.

Kazanımlar: K7. Veri soyutlama, kapsülleme, kalıtım ve polimorfizm kavramlarını açıklar.

Materyaller: L6.A5.1. Adam Asmaca Oyunu

Hazırlık: Materyal öğrencilerin tümünün görebileceği şekilde ekrana yansıtılır ve ÖYS üzerinden erişime açılır.

Uygulama: Eğitmen bu öğrenme görevine başlamadan önce nesne yönelimli programlamanın temelini oluşturan kavramları bugün inceleyeceklerini belirtir. Bu kelimeleri direkt vermeden önce tahminde bulunabilecekleri bir oyun oynayacaklarını açıklar. Sınıf iki gruba ayrılır. Daha sonra öğrencilere adam asmaca oyunu oynatılır. Eğitmen tahtaya kelimenin tanımı, kelime ile ilgili bir örnek ve görselinden oluşan ipuçlarını herkesin görebileceği şekilde tahtaya yansıtarak gruplara sırayla sorar. İpuçlarına bağlı olarak öğrencilerden kelimeyi tahmin etmelerini ister. Gerektiğinde bir harf alabilecekleri belirtilir. Öğrencilerin beş farklı harf alma hakkı vardır. İlk grup kelimeyi bilemezse, diğer gruba da harf almaksızın bir kere tahmin hakkı verilir. Kelime ortaya çıktıkça eğitmen ilgili kelime hakkındaki konuyu aşağıdaki gibi özetlemeye çalışır.

Eğitmene Öneriler: Eğitmen adam asmaca oyununu dijital ortamda tasarlayıp ve tüm öğrencilerin görebileceği şekilde oyunu tahtada başlatabilir. Bunun için ücretsiz ve basit şekilde oyunun geliştirilebileceği örnek uygulamalardan birisi https://learningapps.org/' dir.

Eğitmen materyalde yer alan ipuçları ve bilinmesi gereken kelimeleri learningapps sitesinden ilgili template seçerek adam asmaca oyununu tasarlayabilir. Oyunun tam ekran paylaşım linki ise öğrencilerle Moodle üzerinden paylaşılabilir.

Konu İçeriği:

Nesne yönelimli programlamanın özü, kodda belirli özelliklere ve yöntemlere sahip bir nesne oluşturmaktır. Nesneleri kullanarak, kodun diğer bölümlerinin bu fonksiyon dışında bu verilere erişememesi için verileri ve üzerinde çalışan fonksiyonları birleştirmektir. C++ modülleri tasarlarken, tüm dünyayı nesne şeklinde görmeye çalışıyoruz. Örneğin araba; renk, kapı sayısı, hız limiti gibi belirli özelliklere sahip bir nesnedir. Farklı isimler ve markalara sahip birçok araba olabilir, ancak hepsi bazı ortak özellikleri paylaşmaktadır. Ayrıca, hızlanma ve yavaşlama gibi belirli yöntemlere de sahiptir. Nesne yönelimli programlamanın temelini oluşturan kavramlar şunlardır:

Resim 25. *Nesne yönelimli programlamanın temelini oluşturan kavramlar*

1. Veri Soyutlama

Veri soyutlama, dış dünyaya sadece gerekli bilgileri sağlama ve arka plan ayrıntılarını gizleme, yani ayrıntıları sunmadan programdaki gerekli bilgileri temsil etme anlamına gelir. Bir arabanın nasıl sürüleceğini ve nasıl yakıt ekleneceğini biliyor olabiliriz, fakat motorun nasıl çalıştığı hakkında bir fikrimiz var mı? Hayır ve ayrıca bilmemize gerek de yok çünkü bu detay soyutlanmış durumda.

Araba örneğinden devam edersek, araba kullanan adam sadece gaza basmanın arabanın hızını artıracağını veya fren yapmanın arabayı durduracağını bilir, ancak gaza bastığında hızın gerçekte nasıl arttığını ya da arabanın iç mekanizması (gaz, fren vb. çalışma prensibini) bilmez. Başka bir örnek vermek gerekirse; mahalledeki pastaneden en sevdiğiniz kurabiyelerden satın alırken kurabiyenin yapım aşamasında kullanılan unun veya şekerin markasını ya da ne kadar kabartma tozu kullanıldığı gibi bilgileri düşünmüyorsunuz. Önemli olan kurabiyenin lezzetidir ki işin asıl amacı da budur yani beğenilmesidir.

Araba (veri soyutlamasız)

Araba (veri soyutlamalı)

Resim 26. *Veri soyutlama araba örneği*

Veri soyutlama, veri üyelerini gizleme ve sınıfın kullanıcısının verileri bozmaması için bir arayüzün arkasında bir sınıfın uygulanması sürecidir. Buradaki fikir, verilerin bir sınıf için uygulamanın içinde gizli olmasıdır. Veriler doğrudan değil, "public" fonksiyonlar aracılığıyla manipüle edilir. Genel kural, tüm değişkenler ve üyeler için mümkün olan en küçük kapsamı kullanmaktır. Ayrıca, olabildiğince az global değişken kullanın. Bunu yaparken, yalnızca verilere erişimi olması gereken kod bu verileri değiştirebilir. Başka bir kodun özel bir nesne içindeki verileri kullanması veya değiştirmesi gerekiyorsa, o nesnenin üye fonksiyonları bu tür erişime izin veren yordamları sağlayabilir.

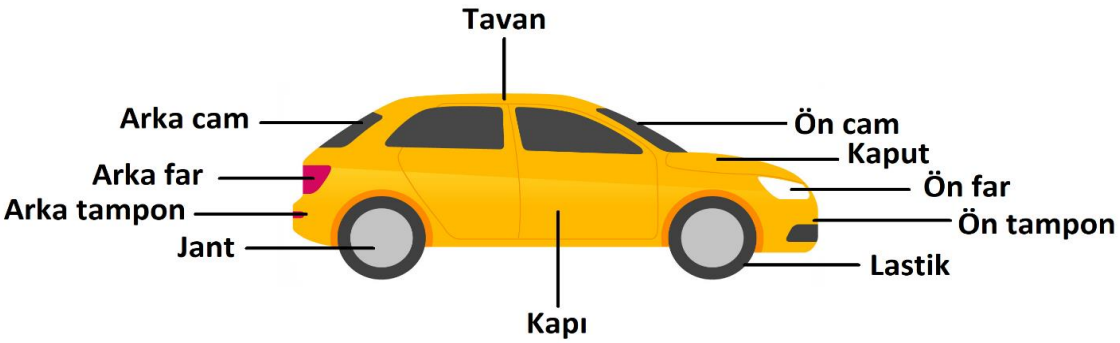
Örneğin, bir veritabanı sistemi, verilerin nasıl saklandığına, oluşturulduğuna ve muhafaza edildiğine ilişkin belirli ayrıntıları gizler. Benzer şekilde, C ++ sınıfları, bu yöntemler ve veriler hakkında dahili ayrıntılar vermeden dış dünyaya farklı yöntemler sağlar.

2. Kapsülleme

Verileri ve bu veriler üzerinde çalışan fonksiyonları aynı yere yerleştirmektir. Nesne yönelimli programlama, size verileri ve ilgili fonksiyonları aynı nesneye yerleştirmek için çerçeve sağlar. Bir sınıfın değişkenleri veya verileri başka herhangi bir sınıftan gizlenir ve yalnızca bildirildikleri kendi sınıfının herhangi bir üye fonksiyonu aracılığıyla erişilebilir. Kapsülleme fikri sınıfları birbirinden ayrı tutmak ve birbirleriyle sıkı sıkıya bağlı olmalarını önlemek için kullanılır. Oluşturduğunuz her sınıf, belirli bir şeyi veya kavramı temsil etmelidir. Birden çok sınıf, nesnelerin veya kavramların kombinasyonlarını temsil etmek için daha sonra bir araya gelir.

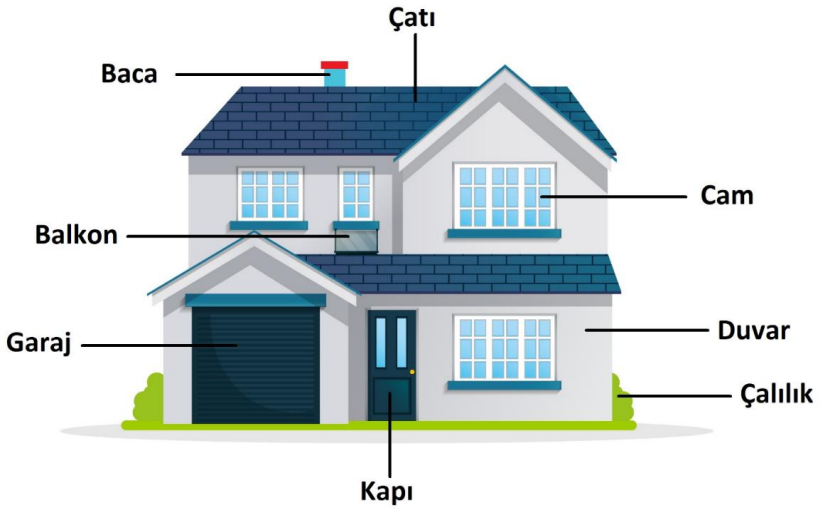
Araba örneğine tekrar bakalım. Bir arabanın hidrolik direksiyonu, dahili olarak birbirine bağlanmış çok sayıda bileşene sahip karmaşık bir sistemdir ve bu sistemdeki bileşenler aracı istenen yönde döndürmek için senkronize olarak çalışır. Motorun direksiyon simidine verdiği gücü bile kontrol eder. Ancak dış dünyaya yani bize ulaşan sadece bir arayüz mevcuttur ve karmaşıklığın geri kalanı gizlidir. Ayrıca, direksiyon ünitesi kendi içinde eksiksiz ve bağımsızdır. Başka bir mekanizmanın çalışmasını etkilemez.

Aşağıdaki gibi bir araba üretmek istediğinizi düşünün. Arabanın bitmiş hâlini oluşturmadan önce daha küçük alt parçaları oluşturmalısınız. Her küçük parça küçük bir fonksiyonu vardır, ancak hepsi bir araba üretmek için birleşir. Aşağıdaki görselde bir arabanın farklı parçaları gösterilmektedir. Her bölüm benzersiz bir amaca hizmet etmektedir.



Resim 27. *Veri kapsülleme araba örneği*

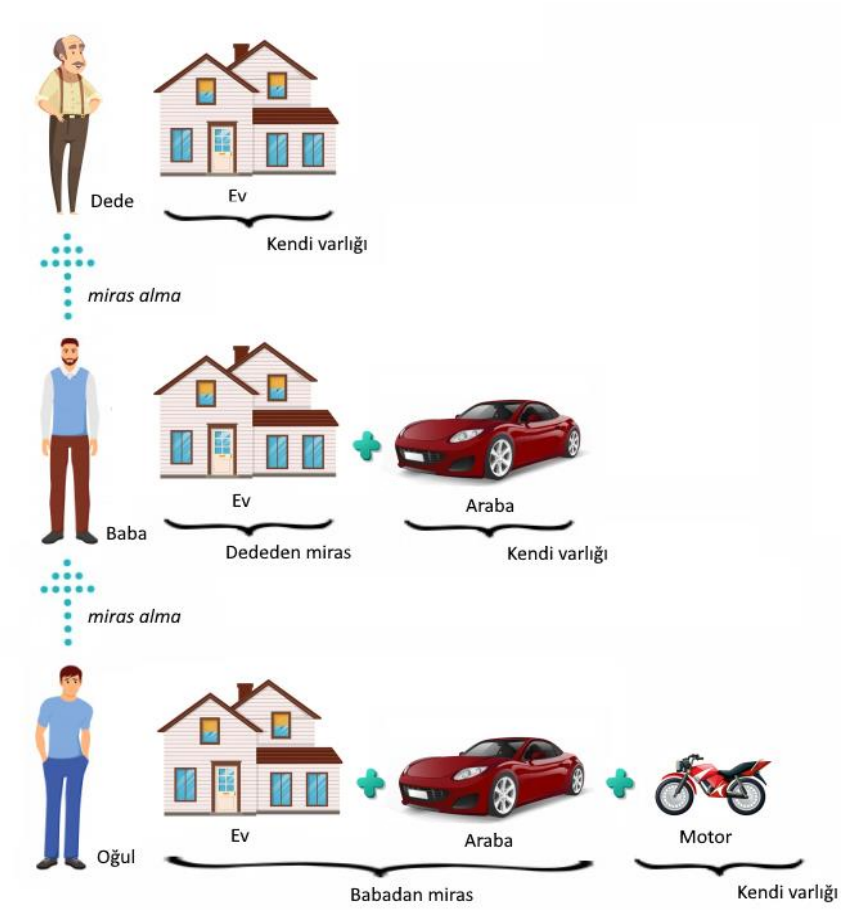
Aşağıdaki görselde ise bir evin farklı parçaları gösterilmektedir. Aynı şekilde burada da her bölüm benzersiz bir amaca hizmet etmektedir.



Resim 28. *Veri kapsülleme ev örneği*

3. Kalıtım

Kalıtım, bir nesnenin başka bir nesnenin belirli ya da tüm özelliklerini edinme mekanizmasıdır. Üst sınıfın özelliklerinin alt sınıfa aktarılmasını sağlayarak türetilmiş sınıflar oluşturur. Kodun yeniden kullanılabilirliğini sağladığı ve kod boyutunu azaltmaya yardımcı olduğu için nesne yönelimli programlamada çok önemli bir kavramdır.



Resim 29. *Kalıtım örneği*

Üyeleri temel bir sınıftan devralma yeteneği, temel sınıfta tanımlanan belirli ortak özellikleri paylaşan türetilmiş sınıfların oluşturulmasına izin verir.

UYARI:

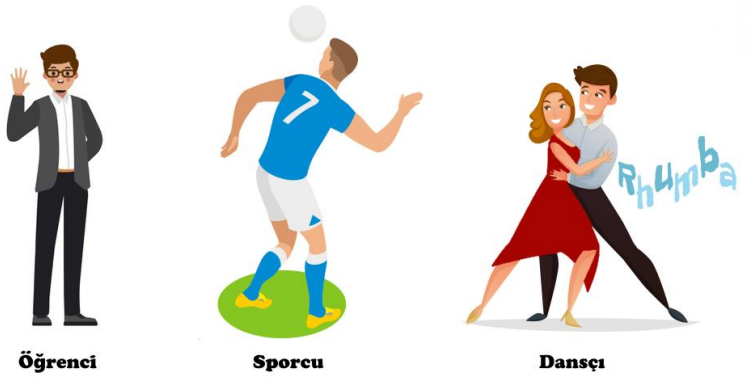
Sınıf örneklerini ve türetilmiş sınıfları birbiri ile karıştırmayın. Örnek, bir sınıfın kopyasıdır, türetilmiş sınıf ise türetildiği temel sınıfın özelliklerini miras alan yeni bir sınıftır.

Örneğin otomobiller, kamyonlar ve motosikletler dahil olmak üzere motorlu taşıtlarla ilgili bir uygulamanın gerektiğini varsayalım. Program, her türlü aracı temsil etmek için "taşıt" adlı bir sınıf kullanabilir. Arabalar, kamyonlar ve motosikletler taşıt türleri olduğundan, "taşıt" sınıfının alt sınıfları ile temsil edilecektir. "taşıt" sınıfı tüm taşıtlar için ortak olan plaka ve sahibi gibi genel bilgilere sahip olabilir. Bunlar tüm taşıtlarda ortak olan bilgilerdir. Ancak her bir taşıta özel bir değişken de eklenebilir. Araba sınıfı kaç kapılı bilgisini, kamyon sınıfı tekerlek sayısını ve motosiklet sınıfı da motosiklet sepeti olup olmadığı bilgisini alabilir.

4. Polimorfizm (Çok Biçimlilik)

Bir işleci veya fonksiyonu farklı şekillerde kullanma yeteneğine, başka bir deyişle işleçlere veya fonksiyonlara farklı anlam veya özellikler verme yeteneğine polimorfizm denir. Poli

birçok anlamına gelirken morf ise form anlamındadır. Yani bir nesnenin aynı sürece farklı şekillerde yanıt verebilme yeteneğidir. Tek bir fonksiyon veya kullanımdan farklı şekillerde fonksiyon gören bir operatöre polimorfizm denir. Daha iyi anlamak için aşağıdaki gerçek hayat örneğine bakalım. Bir kişi sınıfta kendini öğrenci, sahada bir sporcu ve dans kulübünde bir dansçı olarak temsil ediyor.



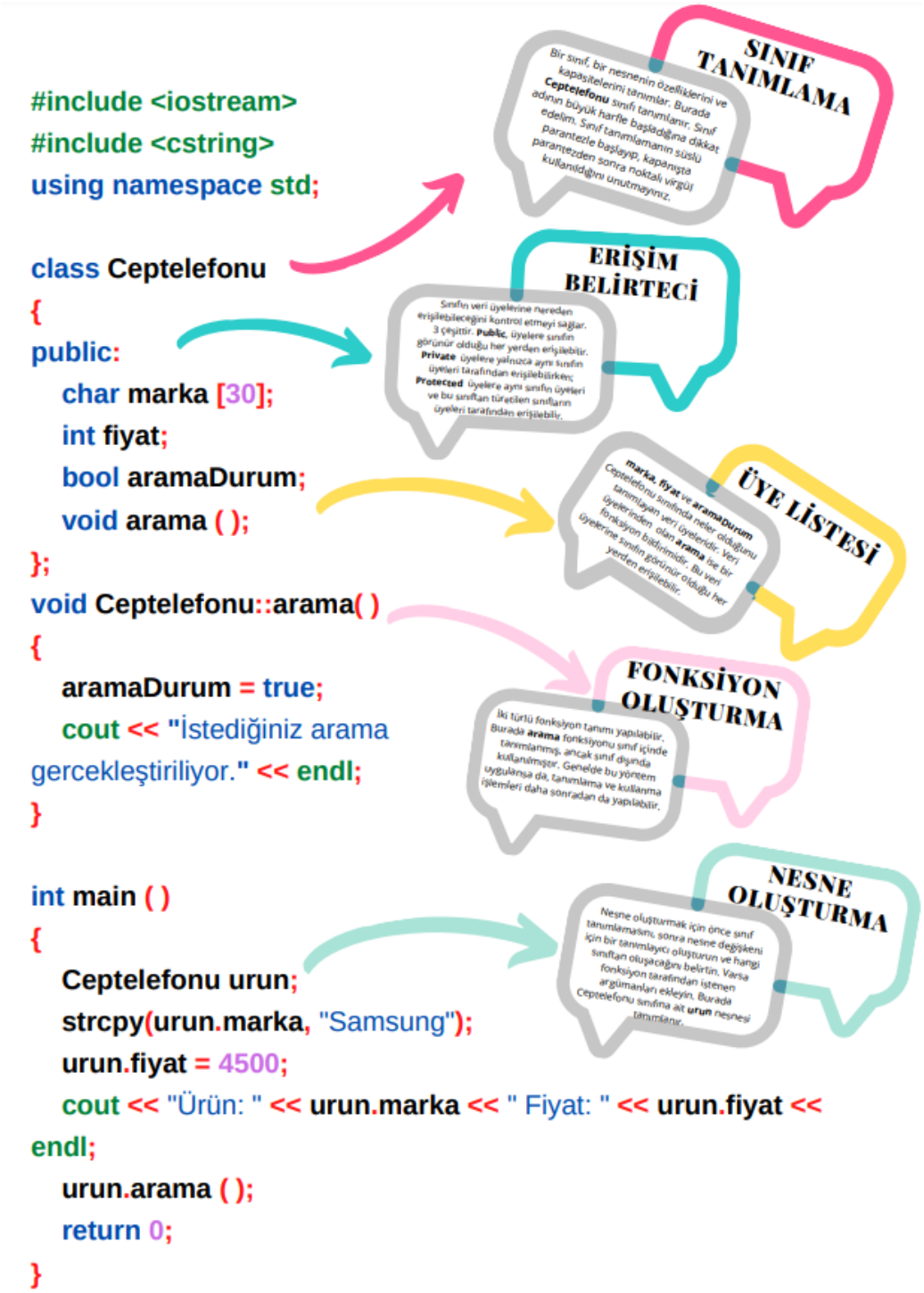
Resim 30. *Polimorfizm kişi örneği*

Aynı araba örneğine bakalım. Bir araçta vites iletim sistemi vardır. Dört ön vites ve bir arka vites vardır. Motor hızlandığında, hangi vitese geçildiğine bağlı olarak, araca farklı miktarda güç ve hareket verilir. Eylem vites değiştirme olarak aynıdır, ancak vites tipine bağlı olarak eylem farklı davranır. İşte bu durumda polimorfizm mantığı devreye girmiş olur. Polimorfizm mantığını aşağıdaki örneği kullanılarak da açıklayabiliriz. Aşağıdaki verilen nesnelerin doğasında *ses_cikar()* yöntemi vardır, ancak her canlı bunu farklı şekilde gerçekleştirir.

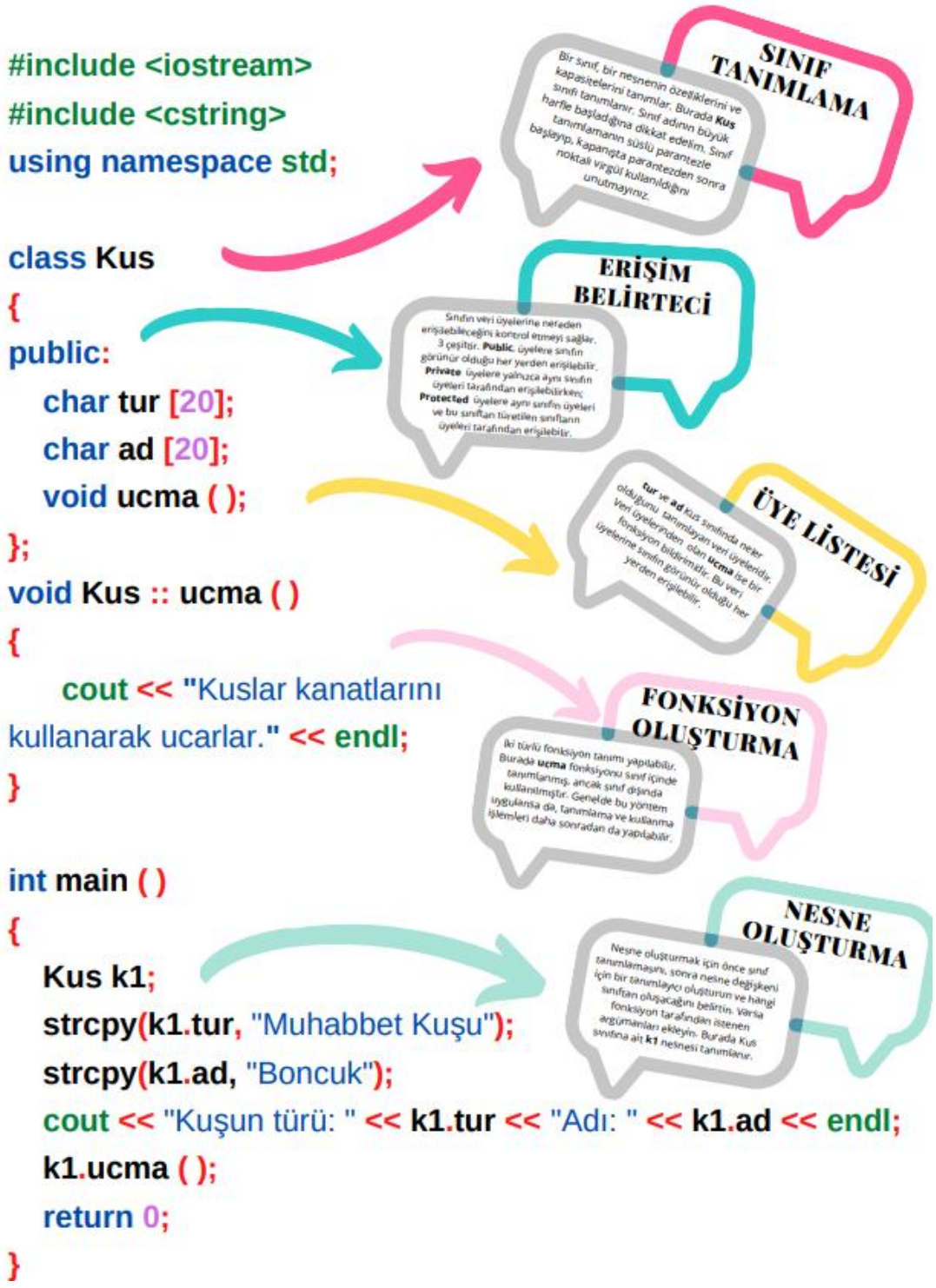
Resim 31. *Polimorfizm ses çıkarma örneği*

Hafta 6. Yüz Yüze: Ders Materyalleri

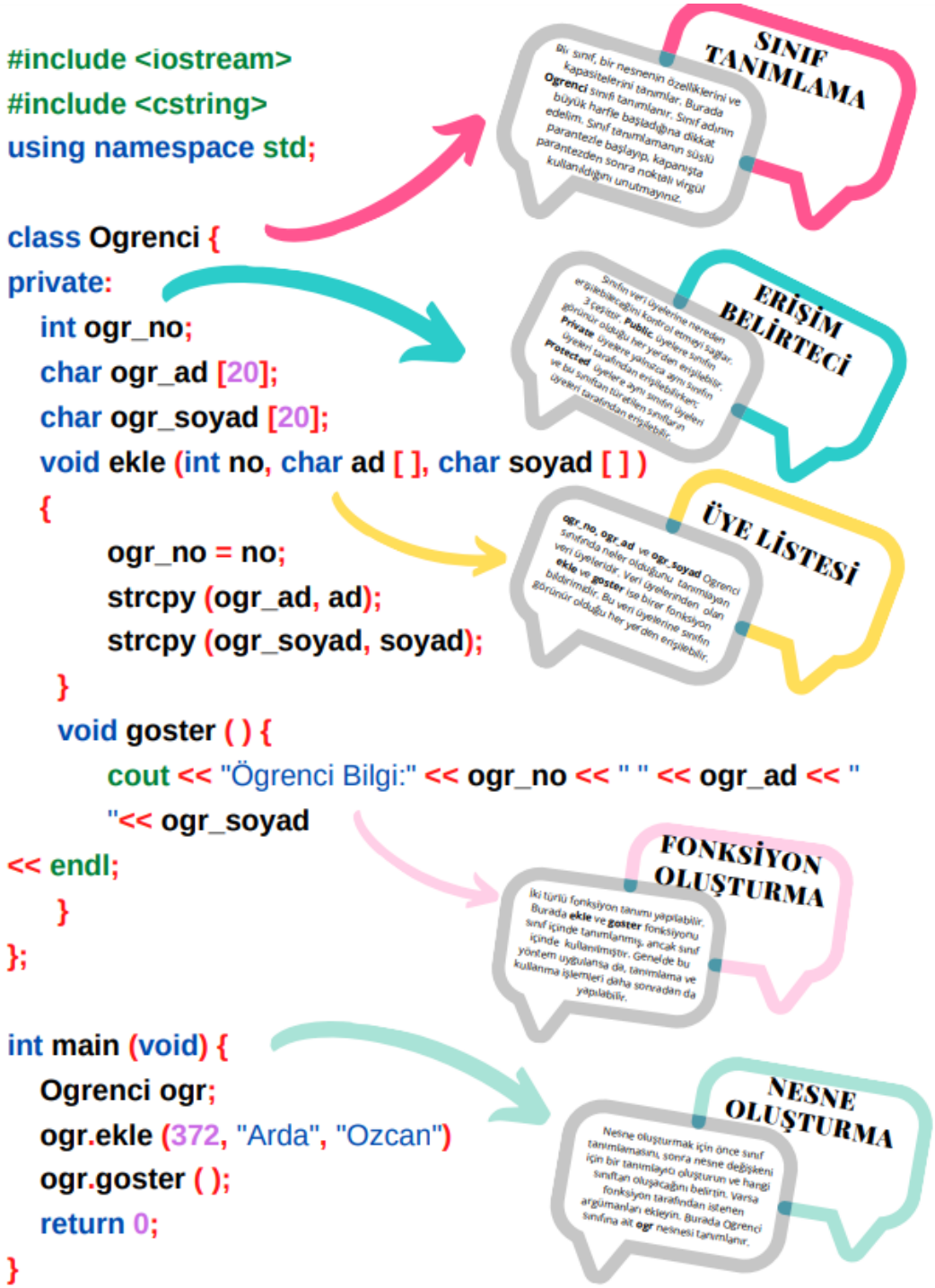
L6.A2.1. Grup Afişleri



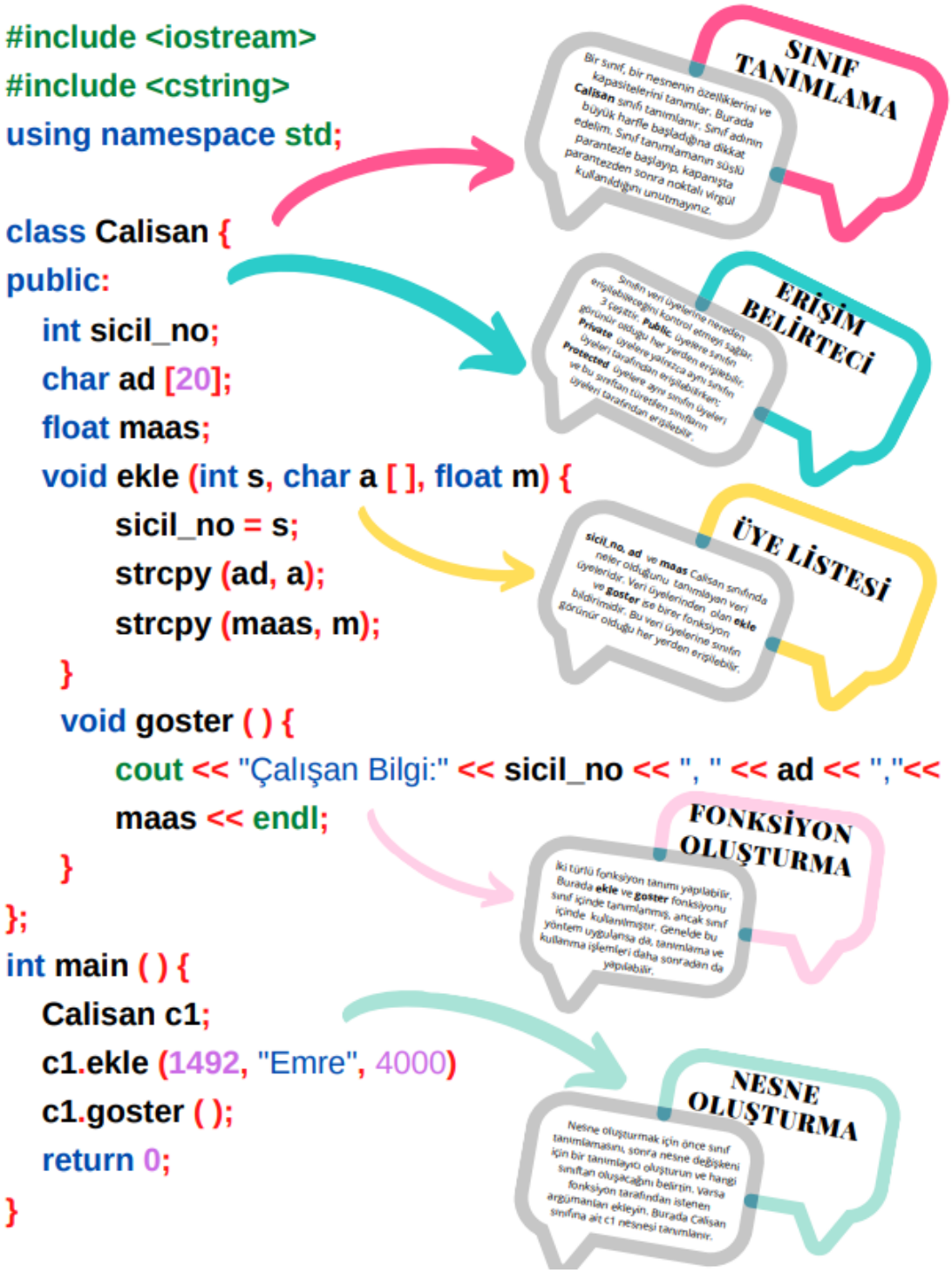
Resim 32. *Cep telefonu sınıfı afişi*



Resim 33. *Kuş sınıfı afişi*

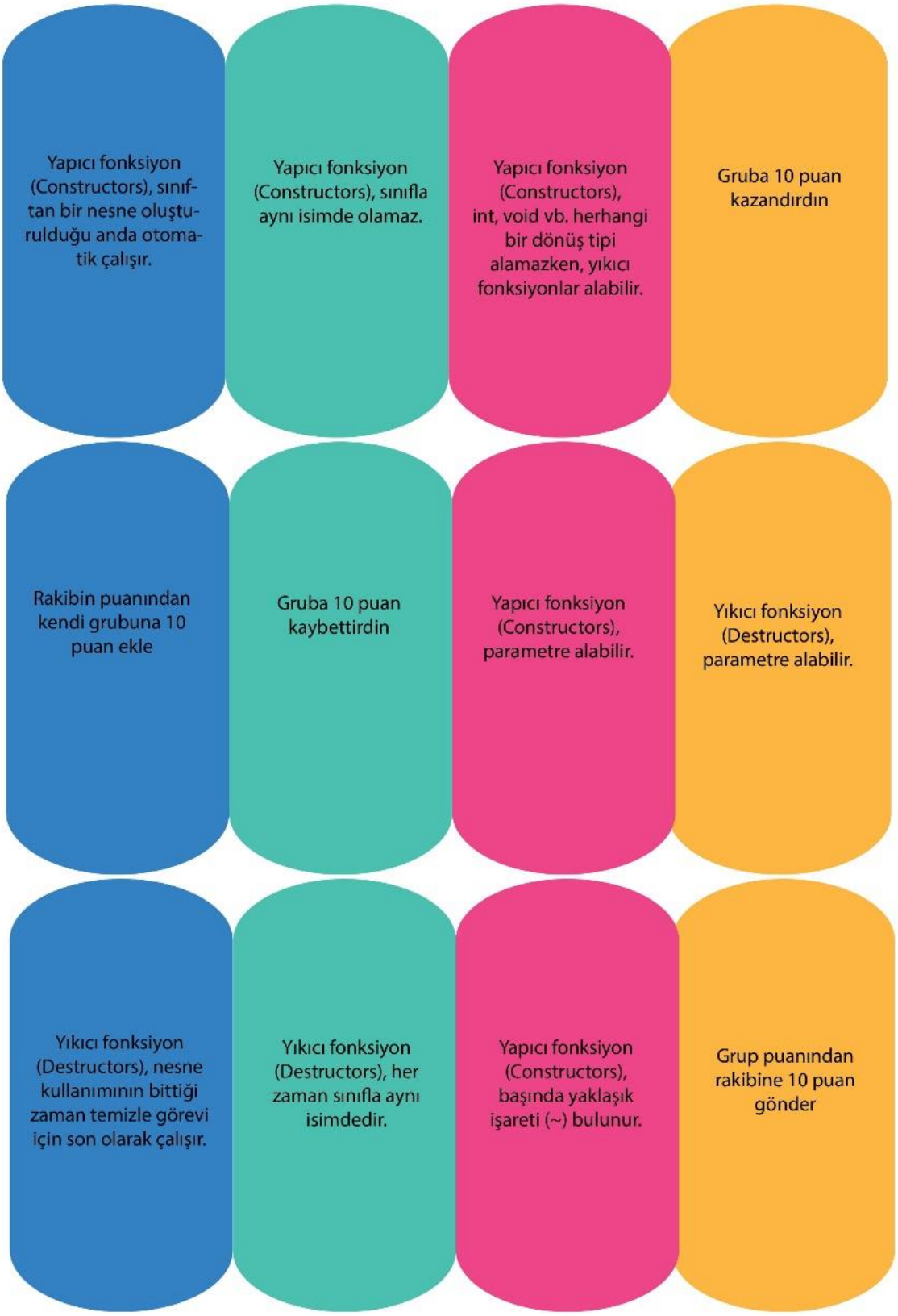


Resim 34. *Öğrenci sınıfı afişi*

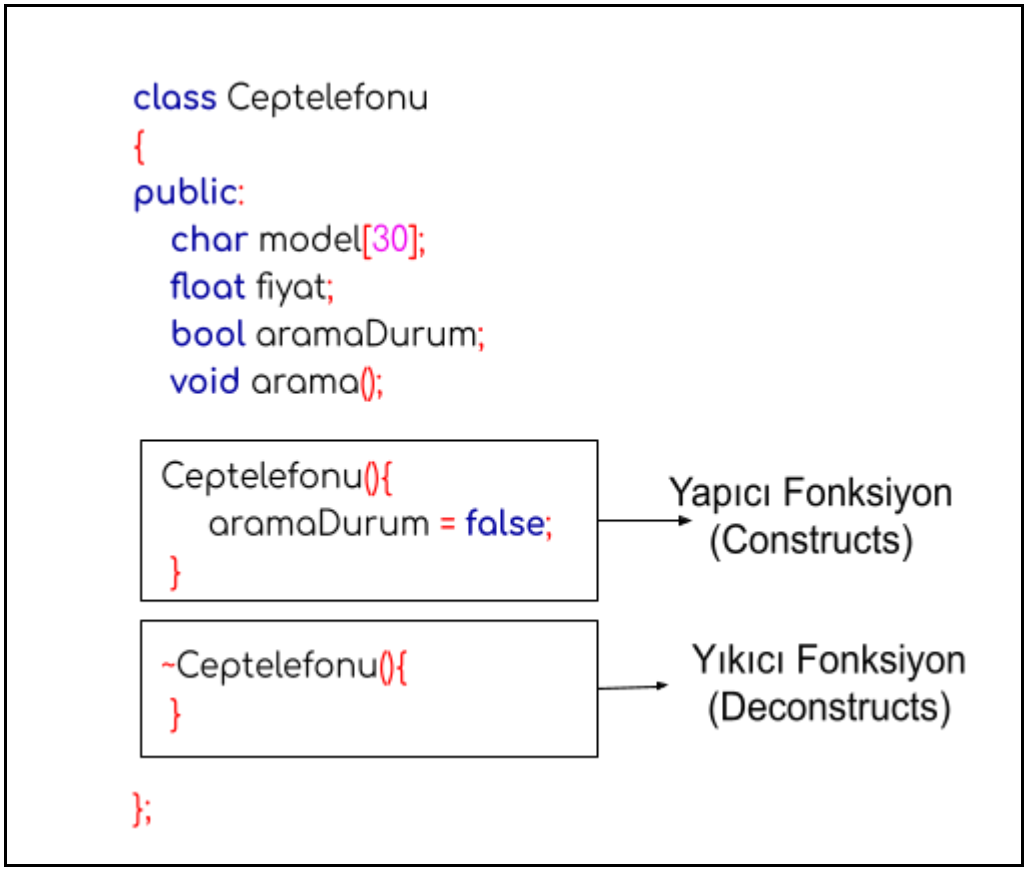


Resim 35. *Çalışan sınıfı afişi*

L6.A3.1. Yarışma Kartları



L6.A3.2. Kod Örneği



L6.A4.1. Grup Görev Kartları

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    char marka[30];
    char model[30];
    int yil;
    int fiyat;
};
int main() {
    Araba araba1;
    strcpy(araba1.marka, "Suzuki");
    araba1.fiyat = 225000;
    strcpy(araba2.model, "T-Roc");
    araba2.yil = 2020;
    cout << "Araba 1: " << araba1.marka << ", " << araba1.model << ", " << araba1.yil
<< ", " << araba1.fiyat << " \n";
    return 0;
}
```

Kodun Çıktısı:

Araba 1: Suzuki, Vitara, 2016, 225000

Araba 2: Volkswagen, T-Roc, 2020, 360000

Görev: Grup 1

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    char marka[30];
    char model[30];
    int hiz(int max_hiz);
};
int Araba::
int main() {
    strcpy(araba1.model, "Astra");
    cout << "Araba: " << araba1.marka << " " << araba1.model << " \n";
    cout << "Hiz: " << araba1.hiz(190); // Fonksiyonu parametre ile cagirin
    return 0;
}
```

Kodun Çıktısı:

```
Araba: Opel Astra
Hiz: 190
```

Görev: Grup 2

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

```
#include <iostream>
#include <cstring>
using namespace std;
class Araba {
public:
    string marka;
    string model;
    int yil;
    int fiyat;
    Araba(string) {
    }
};
int main() {
    Araba araba2("Volkswagen", "T-Roc", 2020, 360000);
    cout << araba1.marka << " " << araba1.model << " " << araba1.yil << " " << araba1.fiyat << "\n";
    cout << araba2.marka << " " << araba2.model << " " << araba2.yil << " " << araba2.fiyat << "\n";
    return 0;
}
```

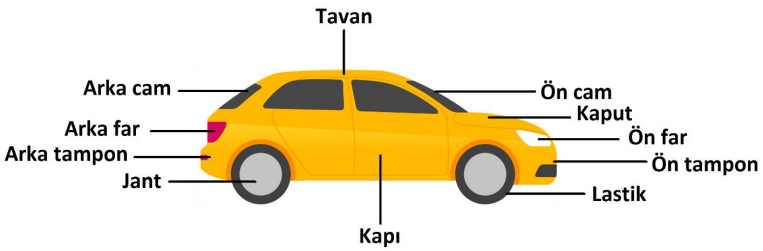
Kodun Çıktısı:

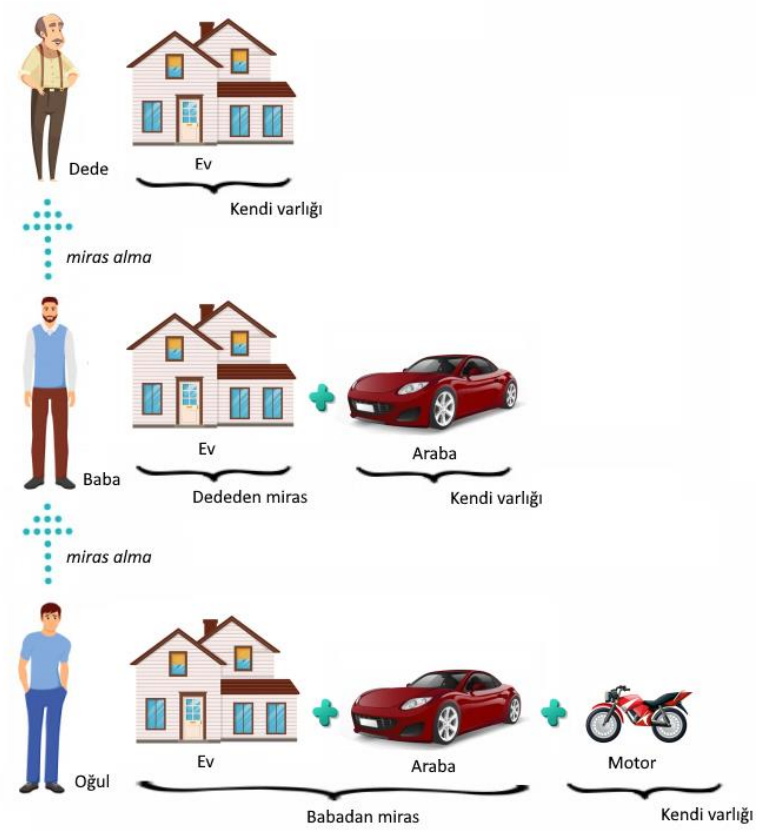
```
Suzuki Vitara 2016 225000
Volkswagen T-Roc 2020 360000
```

Görev: Grup 3

Yukarıdaki kod çıktısının aynısını ekranda görmek için verilen koddaki eksik satırları tamamlayınız.

L6.A5.1. Adam Asmaca Oyunu

	İpucu	Kelime
1.	  Dış dünyaya sadece gerekli bilgileri sağlama ve arka plan ayrıntılarını gizleme, yani ayrıntıları sunmadan programdaki gerekli bilgileri temsil etme anlamına gelir. Buradaki fikir, verilerin bir sınıf için uygulamanın içinde gizli olmasıdır. Örneğin bir arabanın nasıl sürüleceğini ve nasıl yakıt ekleneceğini biliyor olabiliriz, fakat bu bilgiler için motorun nasıl çalıştığını bilmek zorunda değiliz.	… … … … … … … … …
2.	 Sınıfları birbirinden ayrı tutmak ve birbirleriyle sıkı sıkıya bağlı olmalarını önlemek için kullanılır.	… … … … … … … … … …

	Oluşturduğunuz her sınıf, belirli bir şeyi veya kavramı temsil etmelidir. Birden çok sınıf, nesnelerin veya kavramların kombinasyonlarını temsil etmek için daha sonra bir araya gelir. Örneğin, yukarıdaki gibi bir arabanın bitmiş hâlini oluşturmadan önce daha küçük alt parçaları oluşturmalısınız. Her küçük parça küçük bir işlevi vardır, ancak hepsi bir araba üretmek için birleşir. Aşağıdaki görselde bir arabanın farklı parçaları gösterilmektedir. Her bölüm benzersiz bir amaca hizmet etmektedir.	
3.	 Bir nesnenin, başka bir nesnenin belirli ya da tüm özelliklerini edinme mekanizmasıdır. Başka bir ifadeyle veri üyelerini temel bir sınıftan devralma yeteneğidir. Üst sınıfın özelliklerinin alt sınıfa aktarılmasını sağlayarak türetilmiş sınıflar oluşturur. Kodun yeniden	… … … … … … …

	kullanılabilirliğini sağlar ve kod boyutunu azaltmaya yardımcıdır.	
4.	 Öğrenci  Sporcu  Dansçı Bir nesnenin farklı koşullarda farklı davranmasına izin verme durumudur. Başka bir ifadeyle tek bir işlev veya kullanımdan farklı şekillerde işlev gören bir operatördür. Örneğin bir kişi sınıfta kendini öğrenci, sahada bir sporcu ve dans kulübünde bir dansçı olarak temsil edebilir ya da bir adam sınıfta öğretmen, evde baba, markette müşteri gibi davranır. Burada tek bir kişi duruma göre farklı davranmaktadır.	… … … … … … … … … … …

Hafta 6. Çevrim İçi: Ders İçerikleri

Amaç

Bu haftaki çevrimiçi ders içeriğinin amacı, C++'ta nesne yönelimli programlama (OOP) 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 OOP 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 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. Nesne Yönelimli Programlama ile Basketbolcu 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: LÇ6.B1.1. Öğrenci Yönerge Metni

LÇ6.B1.2. Kod Bloğu

LÇ6.B1.2. 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 LÇ6.B1.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir basketbol koçu, oyuncularının bilgilerini bilgisayarda saklamak istemektedir. Bunun için bir Basketbolcu adlı sınıf (class) tasarlamanız beklenmektedir.

Bu sınıf içerisinde:

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

tutan üye değişkenler bulunmalıdır.

Ayrıca, sınıfın içinde:

- oyuncu bilgilerini ekrana yazdıran bir fonksiyon,

yer almalıdır. Programınızda bu sınıftan en az 2 basketbolcu 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: LÇ6.B2.1. Öğrenci Yönerge Metni

LÇ6.B2.2. Kod Bloğu

LÇ6.B2.2. 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 LÇ6.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

LÇ6.B1.1. Öğrenci Yönerge Metni

Göreviniz:

Bir basketbol koçu, oyuncularının bilgilerini bilgisayarda saklamak istemektedir. Bunun için bir Basketbolcu adlı sınıf (class) tasarlamanız beklenmektedir.

Bu sınıf içerisinde:

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

tutan üye değişkenler bulunmalıdır.

Ayrıca, sınıfın içinde:

- oyuncu bilgilerini ekrana yazdıran bir fonksiyon,

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

LÇ6.B1.2. Kod Bloğu

```
#include <iostream>
#include <string>
using namespace std;
class Basketbolcu
{
public:
    string ad_soyad;
    int forma_no;
    int basket_sayisi;
    Basketbolcu(string x_ad_soyad, int x_forma_no, int x_basket_sayisi){
        ad_soyad = x_ad_soyad;
        forma_no = x_forma_no;
        basket_sayisi = x_basket_sayisi;
    }
};
int main()
{
    Basketbolcu b1("Mirsad Turkcan", 5 , 21);
    Basketbolcu b2("Semih Erden", 9, 15);
    cout << b1.ad_soyad << " " << b1.forma_no << " " << b1.basket_sayisi
<< "\n";
    cout << b2.ad_soyad << " " << b2.forma_no << " " << b2.basket_sayisi
<< "\n";
    return 0;
}
```

LÇ6.B1.3. Kod Çıktısı

```
Mirsad Turkcan 5 21
Semih Erden 9 15
```

LÇ6.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

LÇ6.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;
}
```

LÇ6.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 (20 dk.)

B2. Eksik Kodları Dolduruyorum (20 dk.)

Ders Arası (10 dk.)

B3. Kütüphanelerdeki Bazı Fonksiyonları Kullanarak Kodluyorum (50 dk.)

Ders Arası (10 dk.)

B4. Neden Dosyalama İşlemleri Yaparız? (30 dk.)

B5. C++ Dilinde Dosyalama İşlemleri Yapıyorum (20 dk.)

Ders Arası (10 dk.)

B6. 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: 20 dk.

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

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: L7.B1.1 C++ Programlama Dilinde Kütüphaneler

L7.B1.2 Karakter Kütüphanesi

L7.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 "L7.B1.1. C++ Programlama Dilinde Kütüphaneler" öğrenciye ve eğitimcilere kütüphanelerin ne olduğuna yönelik derse giriş yapılması için hazırlanmıştır. Diğer iki materyalde ise noktalı boş olan yerleri öğrenci gruplarının doldurarak kütüphane ve o kütüphanedeki kullanılabilecek hazır fonksiyonları öğrencilerin gerek materyalle gerek akranlarıyla etkileşime girerek öğrenmesi amaçlanmaktadır. Eğitmenlerin bu etkinlikteki görevi öğrenci gruplarını sırayla gezerek ihtiyaç duydukları anda geri bildirimler vererek onları desteklemektir.

Eğitmene Öneriler: Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir. Cevaplar kırmızı renkte verilmiştir.

Fonksiyon	İşlevi	Örnek Kullanım	Sonuç
isalpha(c)	c karakteri eğer bir harf ise geriye true değilse false döndürür.	isalpha('2')	F
isdigit(c)	c karakteri eğer bir rakam ise geriye true değilse false döndürür.	isdigit('2')	T
isalnum(c)	c karakteri eğer bir rakam veya harf ise geriye true değilse false döndürür.	isalnum('*')	F
islower(c)	c karakteri eğer bir küçük harf ise geriye true değilse false döndürür.	islower ('d')	T
isupper(c)	c karakteri eğer bir büyük harf ise geriye true değilse false döndürür.	isupper('H')	T
tolower(c)	c karakterini küçük harfe çevirir.	tolower('E')	e
toupper(c)	c karakterini büyük harfe çevirir.	toupper('g')	G
strlen (s1)	s1 katarının uzunluğunu döndürür.	s1 = "Merhaba" strlen (s1)	7
strcpy (s1, s2)	s2 katarını s1 katarına kopyalar.	s1= "Merhaba" s2= "Dunya" strcpy (s1, s2)	Dunya
strcat (s1, s2)	s2 katarını s1 katarının sonuna ekler.	s1= "Merhaba" s2= "Dunya" strcat (s1, s2)	MerhabaDunya

strcmp (s1, s2)	s1 ve s2 aynı ise 0 değerini döndürür; Eğer alfabetik olarak s1, s2 metninden önce geliyorsa -1, sonra geliyorsa 1 değerini döndürür.	s1= "Merhaba" s2= "Dünya" strcmp (s1, s2)	M harfi D harfinden sonra geldiği için 1 değerini döndürür.

B2. Eksik Kodları Dolduruyorum

Süre: 20 dk.

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

K2. Program içinde katar kütüphanesi kullanarak sonucu test eder.

Materyaller: L7.B2.1. C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

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

Uygulama: Öğrenciler ikişerli gruplar hâlindedir. Eğitmenler "L7.B2.1. C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma" adlı materyali bilgisayar başında oturan öğrencilerden ÖYS'de açmalarını ve boşluk bulunan yerleri tahmin etmelerini ister. Eğitmen sınıfta öğrenci gruplarını dolaşarak onların ihtiyaç duydukları anda işlemsel bilgiyi verir. Ayrıca öğrencilerin görebilecekleri şekilde bilgisayarda kodları yazar. Öğrenci ihtiyaç duyduğu anda yazılan kodlara bakabilir.

Eğitmene Öneriler: Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir.

Görev 1: 'a' karakteri ile ilgili örnek kullanım ve ekran çıktısı aşağıdaki gibidir.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char c='a';

    if(isalpha(c))
        cout << "Bu bir harftir."<<endl;
    else
        cout << "Bu bir harf degildir."<<endl;

    if(isdigit(c))
        cout << "Bu bir rakamdir."<<endl;
    else
        cout << "Bu bir rakam degildir."<<endl;

    if(islower(c))
        cout << "Bu bir kucuk harftir."<<endl;
```

```
    if(isupper(c))
        cout << "Bu bir buyuk harftir."<<endl;
}
```

Görev 2: Katarlar üzerinde farklı fonksiyonların kullanımı.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str1[] = "Bugun hava cok guzel.";
    char str2[] = "Piknige gidelim.";
    char str3[50];
    int  uzunluk;

    cout << "str1 katari: " << str1 << endl;
    uzunluk = strlen(str1);
    cout << "str1 katari uzunlugu: " << uzunluk << endl << endl;

    cout << "str2 katari: " << str2 << endl;
    uzunluk = strlen(str2);
    cout << "str2 katari uzunlugu: " << uzunluk << endl << endl;

    strcpy(str3, str1);
    cout << "str3 katari: " << str3 << endl << endl;

    strcat(str1, str2);
    cout << "s1 katari: " << str1 << endl;

    uzunluk = strlen(str1);
    cout << "s1 katari uzunlugu: " << uzunluk << endl << endl;

    if(strcmp(str1,str3)==0)
        cout << "Iki katar birbirine esittir" << endl;
    else
        cout << "Iki katar birbirine esit degildir." << endl;
}
```

B3. 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: L7.B3.1. Kütüphaneleri ve Fonksiyonları Kullanma: Görevler

Hazırlık: Materyal ÖYS üzerinden erişime açılır.

Uygulama: Eğitmenler C++ programlama dilindeki bazı kütüphane ve hazır fonksiyonları kullanıp uygulaması için hazırlanan görev etkinliğinde en az bir tanesini öğrencilerin seçmesini sağlayarak, öğrencilerin bu derste kodlama yapmasını ister. Diğer görevler ise öğrencilere kodlanması için ev görevi olarak verilebilir. Öğrenciler kodlama esnasında verilen görevler için ilk derste verilen katar ve karakter kütüphaneleri ilgili materyalleri destekleyici bilgi olarak

kullanılırken, öğrencinin ihtiyaç duyduğu anda gerekli işlemsel bilgiyi eğitmen sınıfı dolaşarak ve kodlamayı öğrencilerin de görebileceği şekilde bilgisayarda kodlayarak sağlayabilir.

Eğitmene Öneriler:

Fonksiyon kodlama görevleri ve cevapları aşağıdaki gibidir.

Görev 1: Serkan yazdığı programda isim kısmına kullanıcının rakam girmesi durumunda programının "isminizde rakam olmaz" hatasını vermesini istiyor. Bunun için "isalpha" kodunda kullanan Serkan sizce nasıl bir kod yazmıştır?

Cevap 1:

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char ad[50];

    cout << "Adinizi giriniz:";
    cin >> ad;

    for(int i=0; i<strlen(ad);i++)
    {
        if(!isalpha(ad[i]))
        {
            cout << "Isminizde rakam olamaz!";
            return 0;
        }
    }
    cout << "Merhaba " << ad;
    return 0;
}
```

Görev 2: Ahmet yazacağı bir programın şifresinin otomatik olarak rastgele belirlenecek sekiz rakamdan oluşmasını istemektedir. Bunun için nasıl bir kod yazmalıdır?

Cevap 2:

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

int main()
{
    srand(time(0));

    int karakterSayisi = 8;
    char sifre[9];

    for(int i=0;i<karakterSayisi;i++)
    {
        char karakter;
        do
        {
            karakter = rand() % 255;
        }
```

```
        while(!isalnum(karakter));

        sifre[i] = karakter;
    }
    sifre[karakterSayisi] = '\0';
    cout << "Sifreniz: " << sifre;
    return 0;
}
```

B4. Neden Dosyalama İşlemleri Yaparız?

Süre: 30 dk.

Kazanımlar: K3. Dosyalama işleminin gerekliliğini açıklar.

K4. Dosya kütüphanesi kullanarak program geliştirir.

Materyaller: L7.B4.1. Dosyalama İşlemleri Görev Yaprağı

L7.B4.2. Dosyalama İşlemleri Afişi 1

L7.B4.3. Dosyalama İşlemleri Afişi 2

L7.B4.4. Dosyalama İşlemleri Afişi 3

L7.B4.5. Dosyalama İşlemleri

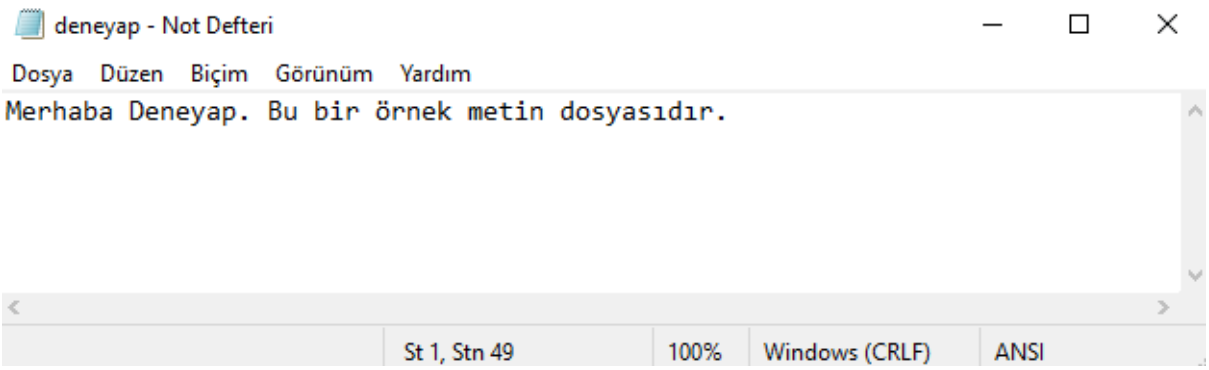
Hazırlık: Eğitmen birinci materyal hariç diğer tüm materyalleri ÖYS üzerinden erişime açar ve öğrencilerin göreceği şekilde projeksiyon ile yansıtır. Bu afiş öğrencilere destekleyici bilgi sağlamak için kullanılacaktır. Eğitmen dersin bu bölümü için hazırlanan materyal olan "L7.B4.1. Dosyalama İşlemleri Görev Yaprağı" adlı görev kâğıdını her gruba bir tane verecek şekilde 5 adet hâlinde çıktı alarak derse gelir.

Uygulama: Eğitmen öncelikle her bir öğrenciye "L7.B4.1. Dosyalama İşlemleri Görev Yaprağı" adlı görev kâğıdını dağıtır. Öğrencilerden bu ders için hazırlanan afişteki destekleyici bilgileri kullanarak bu görev kâğıdındaki boş yerleri doldurması istenir. Eğitmenlerin bu etkinlikteki görevi, öğrenci gruplarını sırayla gezerek onların ihtiyaç duydukları anda geri bildirimler, işlemsel bilgileri vererek öğrencileri desteklemesidir.

Eğitmene Öneriler: Afişlerde geçen bilgiler aşağıdaki gibi özetlenebilir. Öğrencilere verilen görevlerin cevaplarına ise bu bölümün sonunda ulaşılabilir.

DOSYALAMA

Programlama dilleri için dosya, verilerin kalıcı olarak saklanması için kullanılır. Dosya, program yardımıyla veya kullanıcılar tarafından oluşturularak depolama biriminde tutulur. Not defteri ile kolayca oluşturabiliriz.



Program verilerinin aksine, dosyadaki veriler bilgisayar kapansa bile silinmez. Bu sebeple ihtiyaç duyulan önemli bilgiler veya kullanıcılardan alınan bilgiler dosyalar yardımıyla tutulur. Genellikle "txt" uzantılı metin dosyaları kullanılır. "txt" uzantılı dosyalar hem kullanıcılar tarafından hemde program tarafından kolayca oluşturulabilir, okunabilir ve üzerine yazılabilir.

Var olan dosyalar üzerinde yapılacak metinsel işlemler okuma veya yazmadır. Dosya işlemleri ise, yeni dosya oluşturma ve mevcut dosyanın silinmesidir. Tüm bu işlemler C++ programlama dili ile hızlıca yapılmaktadır. Dosya işlemleri için C++ içerisinde üç temel sınıf bulunmaktadır.

ifstream	Okuma amaçlı açılacak dosya işlemleri için kullanılır.
ofstream	Yazma amaçlı açılacak dosya işlemleri için kullanılır.
fstream	Hem okuma hem de yazma amaçlı dosya işlemleri için kullanılır.

Dosyalar üzerinde yapılacak temel işlemler ve fonksiyonları ise aşağıdaki gibidir.

Dosyayı Aç	open()
Dosyadan veri oku	read()
Dosyaya veri yaz	write()
Dosyayı kapat	close()

Dosyayı açma sırasında eğer dosya mevcut değil ise, varsayılan olarak boş bir dosya oluşturulacaktır. Dosyayı farklı modlarda açabiliriz. Bu modlar;

Açıklama	mod
Normal dosya okuma modudur. Dosya en baştan okunmaya başlanır. Bu mod ifstream için varsayılan moddur.	ios::in
Normal dosya yazma modudur. Dosyaya en baştan yazılmaya başlanır. Bu mod ofstream için varsayılan moddur.	ios::out
Dosya yazma modudur. Dosyaya yazım işleminde, veriler dosyanın son karakterinden sonra eklenir.	ios::app
Dosya açıldığında içindeki tüm veriler silinir.	ios::trunc
Sadece dosya mevcut ise dosya açılacaktır. Eğer yoksa dosya oluşturulmayacaktır.	ios::nocreate

Öğrencilere verilen görevlerin cevapları aşağıdaki gibidir. Öğrenci gruplarının doldurdukları görev kâğıtlarının aşağıdaki gibi olması beklenir.

Dosya Açma Kipi	Dosya açılma kontrolü kodu	Dosya mevcutsa ne olur?	Dosya yoksa ne olur?
ifstream sınıfının varsayılan modu okuma için olan ios::in modudur. ofstream sınıfının varsayılan modu yazma için olan ios::out modudur. fstream ise varsayılan modu ios::in ve ios::out modlarıdır.	Bir dosyanın açılıp/açılmadığı "is_open()" fonksiyonu ile kontrol edilir. Eğer hatalı bir durum olduysa "0" değeri döndürecektir. Bu şekilde dosyanın düzgün biçimde açıldığı kontrol edilebilir.	Dosya mevcut ise açma moduna göre içerik silinebilir. ios::out modunda dosya içeriği silinecektir. ios::app modunda ise dosya içeriği korunacaktır.	Eğer dosya mevcut değil ise belirtilen isimde yeni dosya oluşturulacaktır. ios::nocreate modunda ise yeni dosya oluşturulmayacak sadece dosya mevcut ise açılacaktır.
Örnek Kod: fstream dosya; 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!";

B5. C++ Dilinde Dosyalama İşlemleri Yapıyorum

Süre: 20 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: L7.B4.5. Dosyalama İşlemleri

L7.B5.1. Kodlar Arasındaki Farkı Bulma

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan "L7.B4.5. Dosyalama İşlemleri" adlı afişi ÖYS üzerinden erişime açar ve öğrencilerin göreceği şekilde projeksiyon ile yansıtır. Bu afiş öğrencilere destekleyici bilgi sağlamak için kullanılacaktır. Eğitmen dersin bu bölümü için hazırlanan diğer materyal olan "L7.B5.1. Kodlar Arasındaki Farkı Bulma" adlı görev kâğıdını öğrencilerle her gruba bir tane olacak şekilde 5 adet çıktı alınır.

Uygulama: Eğitmenler sınıfı dörder kişilik beş gruba ayırır. Her grubun görebileceği şekilde bu ders için hazırlanan afişi sınıfın belirli bölgelerine asar ve afişi projeksiyon üzerinden yansıtır. Her gruba "L7.B5.1 Kodlar Arasındaki Farkı Bulma" adlı materyali dağıtmadan önce aşağıdaki gibi derse ister bilgisayar ister tahta üzerinden destekleyici bilgiyi sunarak başlar.

Destekleyici Bilgi:

C++ programlama dili üzerinde dosya işlemleri <fstream> kütüphanesi aracılığıyla yapılmaktadır. ifstream ve ofstream sınıfları bu amaçlar için kullanılmaktadır. Dosya okuma işlemleri için ifstream, dosyaya yazma işlemleri için ofstream kullanılır.

Boş dosya oluşturma:

ilk olarak kütüphaneyi projemize dahil ettik. Ardından dosya isimli bir nesne oluşturduk. Parantez içerisine de dosyanın ismini verdik. Eğer dosya yok ise projenin bulunduğu klasörde boş bir metin dosyası oluşturulmuş oldu.

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

İstersek yapıcı fonksiyonu kullanmadan dosya.open() fonksiyonu ile de dosyayı açabiliriz. Varolan dosyanın üstüne bilgi ekleyebiliriz bunun için dosya isminden sonra ikinci parametre olarak mod bilgisini (ios::app) vermemiz gerekmektedir.

Dosya içerisine yazı yazmak için;

```
dosya << "Merhaba Deneyap!";
```

Satırını ekleyelim. Böylece dosyamızın içerisine "Merhaba Deneyap!" yazmış olduk. Son olarak dosyamızı kullanmayı bitirmek için;

```
dosya.close();
```

yazarak dosyamızı kapatıyoruz. Dosyamızı kapatarak geçiçi hafızayı da temizlemiş oluyoruz.

Bu bilgiler ışığında dağıttığım materyal üzerindeki kod farklılıklarını grup arkadaşlarımızla inceleyip, daha sonra hep beraber üzerinde tartışalım.

Eğitmene Öneriler: Eğitmenlere bu derse yönelik öneriler yukarıda (uygulama başlığı altında) verilmiştir.

B6. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

Süre: 50 dk.

Kazanımlar: K5. Dosya okuma işlemlerini içeren program tasarlar.

K6. Dosya yazma işlemlerini içeren program tasarlar.

Materyaller: L7.B4.2. Dosyalama İşlemleri Afişi 1

L7.B4.3. Dosyalama İşlemleri Afişi 2

L7.B4.4. Dosyalama İşlemleri Afişi 3

L7.B4.5. Dosyalama İşlemleri

L7.B6.1. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum

Hazırlık: Eğitmen dersin bu bölümü için hazırlanan afişleri ÖYS üzerinden erişime açar. Eğitmen dersin bu bölümü için hazırlanan materyal olan "L7. B6.1. Dosyalama İşlemleri ile İlgili Verilen Görevleri Kodluyorum" adlı görev kâğıdını her iki öğrenciye bir kopya olacak şekilde 10 adet çıktı alır. Her bilgisayarda 2 öğrenci oturması sağlanarak verilen görevleri iki kişilik grupların yapması sağlanır. Code::Blocks uygulaması öğrencilerin kodlama yapabilmesi için hazır duruma getirilir.

Uygulama: Eğitmenler C++ programlama dilindeki dosyalama işlemleri için hazırlanan görev kâğıdındaki tüm etkinlikleri her öğrenci grubunun yapmasını ister. Öğrenciler kodlama esnasında verilen görevler için bir önceki derste hazırlanan dosyalama işlemleri ile ilgili afişleri 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:

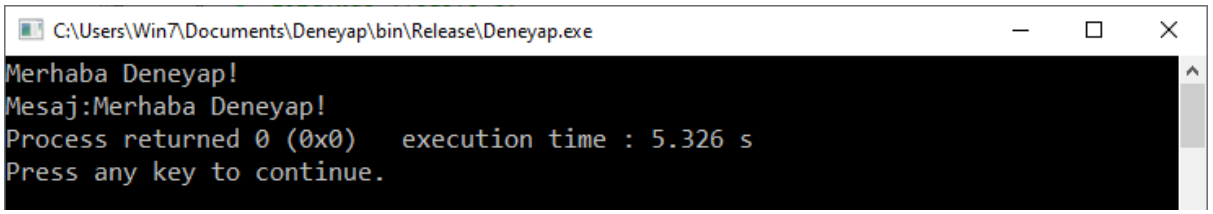
Dosyalama işlemleri ile ilgili verilen görevlerin cevapları aşağıdaki gibidir:

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 36. *Ekran çıktısı*

```
#include <iostream>
#include <fstream>
#include <string>
#include <locale.h>
using namespace std;
int main()
{
    string cumle;
    getline(cin,cumle);
    cout <<"Mesaj:" << cumle;
}
```

Cin komutu kullanıldığında hafızada tutulan önceki girişlerin temizlenmediği durumlarda sorun oluşabilmektedir. Bu yüzden cin.ignore() fonksiyonu kullanılarak giriş hafızası temizlenebilir.

Görev 2: C++ Programlama dili ile dosyalama kullanarak kendinize bir günlük oluşturun.

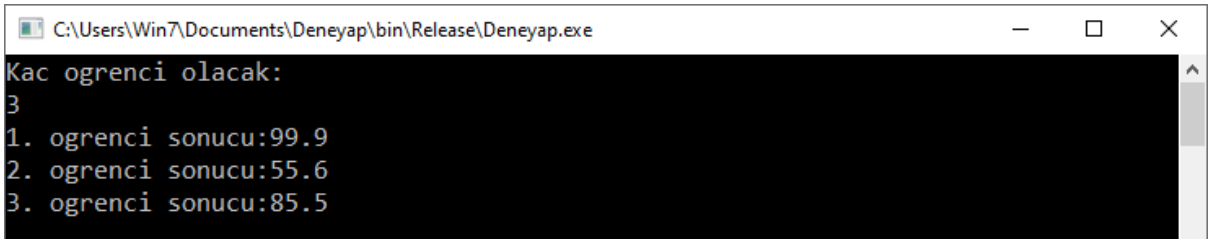
Cevap 2:

```
#include <iostream>
#include <fstream>
#include <string>
#include <locale.h>
using namespace std;
int main()
{
    setlocale(LC_ALL,"");
    int islem;

    cout << "1- Günlük Oku." << endl;
    cout << "2- Günlük Yaz." << endl;
    cin >> islem;

    if(islem==1)
    {
        ifstream dosya("gunluk.txt");
        if(!dosya.is_open())
        {
            cout << "Dosya Okunamadi!";
            return 0;
        }
        string satir;
        while(getline(dosya,satir))
            cout << satir <<endl;
        dosya.close();

    }
    else if(islem==2)
    {
        ofstream dosya("gunluk.txt",ios::app);
        if(!dosya.is_open())
        {
            cout << "Dosya Okunamadi!";
            return 0;
```

```
        }
        string satir;
        cout << "Ne yazmak istersin:\n";
        cin.ignore();
        getline(cin, satir);
        cout << "Kaydediliyor...";
        dosya << satir <<endl;
        dosya.close();
    }
}
```

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

Cevap 3:



Resim 37. *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

L7.B1.1. C++ Programlama Dilinde Kütüphaneler

UYARI:

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

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

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

L7.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)	…..

L7.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)	……..

L7.B2.1. C++ Programa Dilinde Bulunan Kütüphaneleri ve Fonksiyonları Kullanma

Kod	Kodun Çıktısı:

1)……..
2)……..
`using namespace std;`
 	
`int main()`
`{`
`char c='a';`
 	
`if(3.....(c))`
`cout << "Bu bir harftir."<<endl;`
`else`
`cout << "Bu bir harf degildir."<<endl;`
`if(4.....(c))`
`cout << "Bu bir rakamdir."<<endl;`
`else`
`cout << "Bu bir rakam degildir."<<endl;`
 	
`if(5.....(c))`
`cout << "Bu bir kucuk harftir."<<endl;`
 	
`if(6.....(c))`
`cout << "Bu bir buyuk harftir."<<endl;`
 	
`return 0;`
`}`
	Bu bir harftir.
	Bu bir rakam değildir.
	Bu bir küçük harftir.

`#include <iostream>`
1)……..
`using namespace std;`
`int main()`
`{`
`char str1[] = "Bugun hava cok guzel.";`
	:

```
    char str2[] = "Piknige gidelim.";

    char str3[50];

    int  uzunluk;


    cout << "str1 katari: " << str1 << endl;

    uzunluk = 2.....(str1);

    cout << "str1 katari uzunlugu: " << uzunluk << endl <<
endl;


    cout << "str2 katari: " << str2 << endl;

    uzunluk = strlen(str2);

    cout << "str2 katari uzunlugu: " << uzunluk << endl <<
endl;


    3.....(str3, str1);

    cout << "str3 katari: " << str3 << endl << endl;


    4.....(str1, str2);

    cout << "s1 katari: " << str1 << endl;


    uzunluk = 5.....(str1);

    cout << "s1 katari uzunlugu: " << uzunluk << endl <<
endl;


    if(6.....(str1,str3)==0)

        cout << "Iki katar birbirine esittir" << endl;

    else

        cout << "Iki katar birbirine esit değildir." <<
endl;

    return 0;

}
```

```
str1 katari: Bugun hava cok guzel.

str1 katari uzunlugu: 21


str2 katari: Piknige gidelim.

str2 katari uzunlugu: 16


str3 katari: Bugun hava cok guzel.


s1 katari: Bugun hava cok guzel.Piknige gidelim.

s1 katari uzunlugu: 37


Iki katar birbirine esit değildir.
```

L7.B3.1. Kütüphaneleri ve Fonksiyonları Kullanma: Görevler

Görev 1

Serkan yazdığı programda isim kısmına kullanıcının rakam girmesi durumunda programının "isminiz-de rakam olmaz" hatasını vermesini istiyor. Bunun için "isalpha" kodunda kullanan Serkan sizce nasıl bir kod yazmıştır?

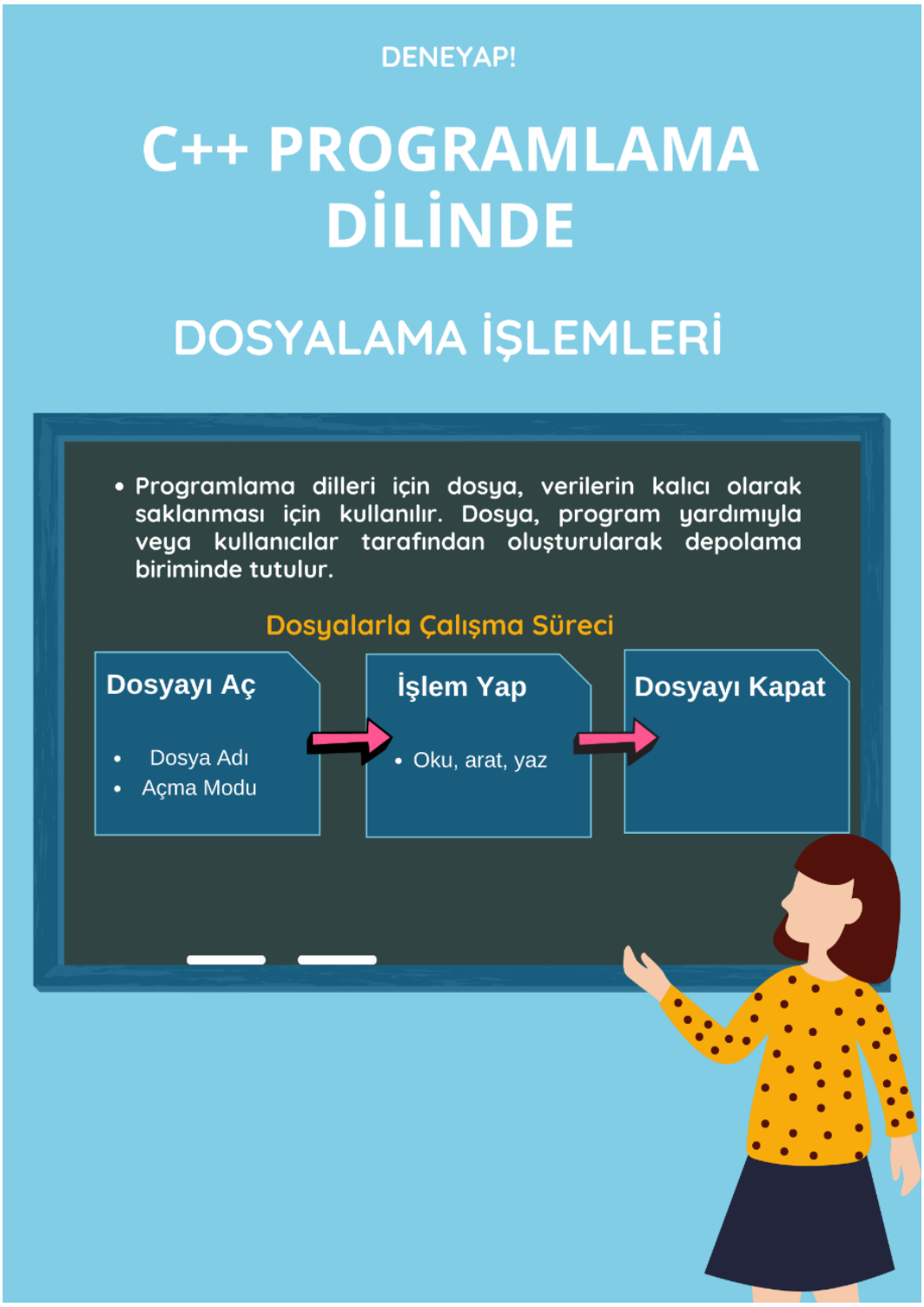
Görev 2

Ahmet yazacağı bir programın şifresinin otomatik olarak rastgele belirlenecek sekiz rakamdan oluşmasını istemektedir. Bunun için nasıl bir kod yazmalıdır?

L7.B4.1. Dosyalama İşlemleri Görev Yaprağı

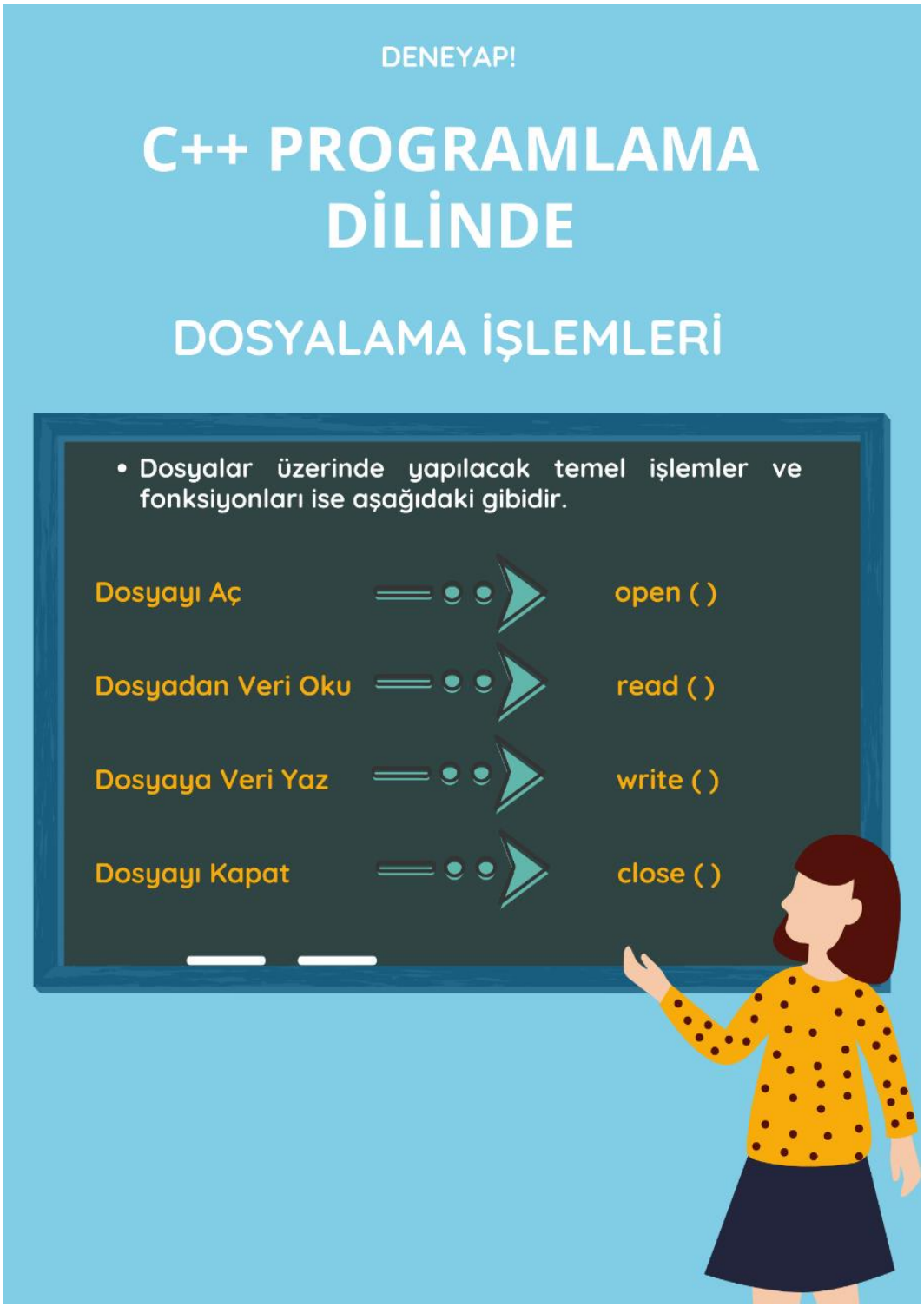
Dosya Açma Kipi	Dosya açılma kontrolü kodu	Dosya mevcutsa ne olur ?		Dosya yoksa ne olur?
ifstream sınıfının varsayılan modu okuma için olan …….. modudur. ofstream sınıfının varsayılan modu yazma için olan …… modudur. fstream ise varsayılan modu …… ve ……. modlarıdır.	Bir dosyanın açılıp/açılmadığı …….. fonksiyonu ile kontrol edilir. Eğer hatalı bir durum olduysa “0” değeri döndürecektir. Bu şekilde dosyanın düzgün biçimde açıldığı kontrol edilebilir.	Dosya mevcut ise açma moduna göre içerik silinebilir. …… modunda dosya içeriği silinecektir. ……. modunda ise dosya içeriği korunacaktır.		Eğer dosya mevcut değil ise belirtilen isimde yeni dosya oluşturulacaktır. …….. modunda ise yeni dosya oluşturulmayacak sadece dosya mevcut ise açılacaktır.
Örnek Kod: fstream dosya; dosya.open(“deneyap.txt”, ios::in | ios::out)	Örnek Kod: if(!dosya.is_open()) cout << “Dosya acilamadi!”;			Örnek Kod: dosya.open(“deneyap.txt”, ios::nocreate) if(!dosya.is_open()) cout << “Dosya mevcut değil!”;

L7.B4.2. Dosyalama İşlemleri Afişi 1



Resim 38. *Dosyalama işlemleri afişi 1*

L7.B4.3. Dosyalama İşlemleri Afişi 2

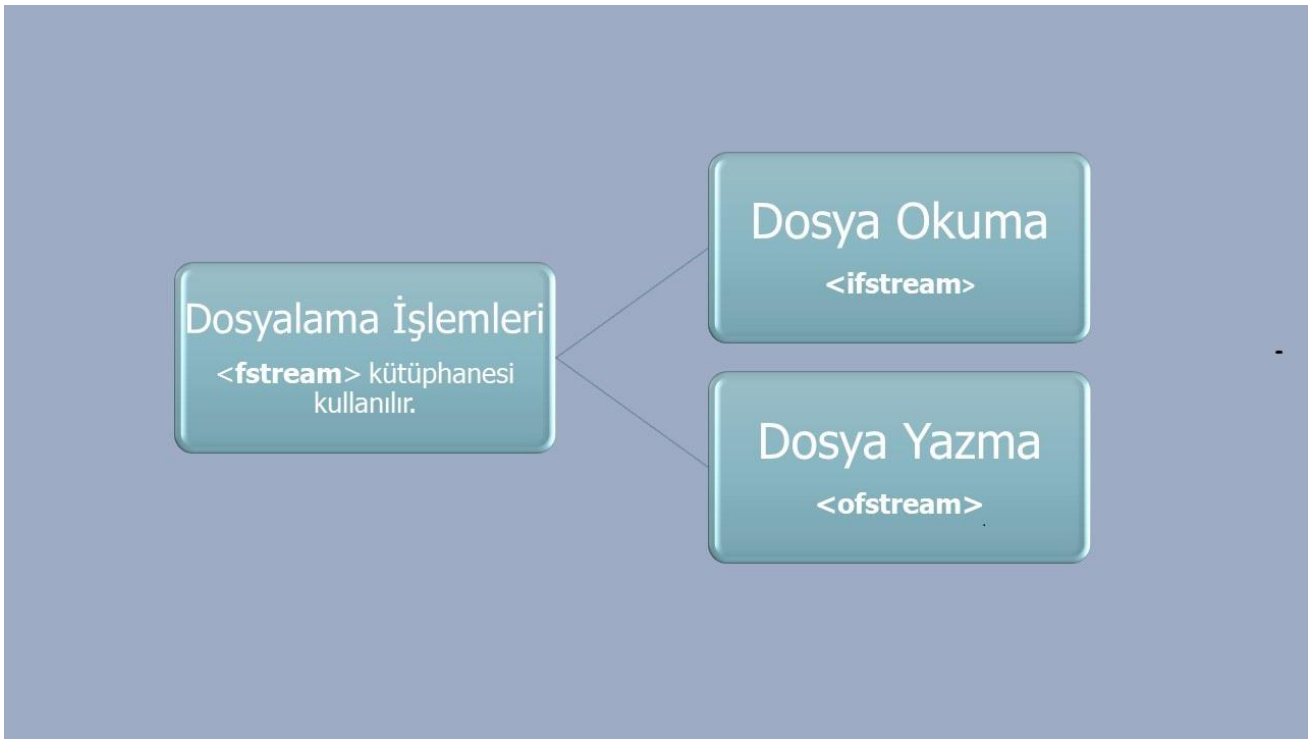


Resim 39. *Dosyalama işlemleri afişi 2*

L7.B4.4. Dosyalama İşlemleri Afişi 3

Resim 40. *Dosyalama işlemleri afişi 3*

L7.B4.5. Dosyalama İşlemleri



Resim 41. *Dosyalama işlemleri*

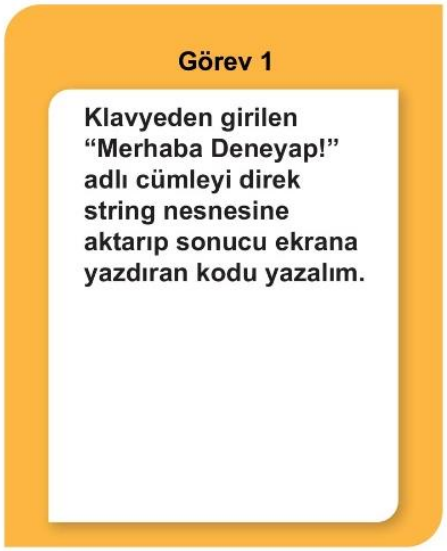
L7.B5.1. Kodlar Arasındaki Farkı Bulma

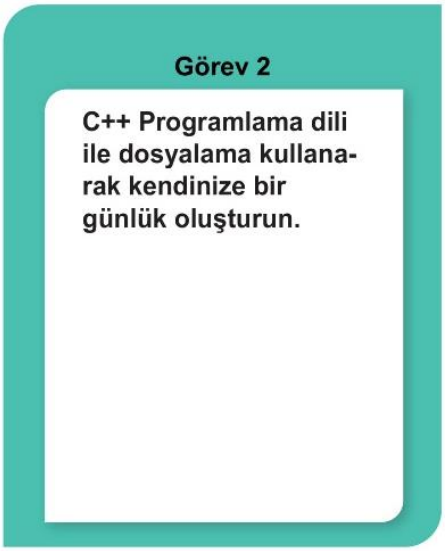
1. İşlem	2. İşlem

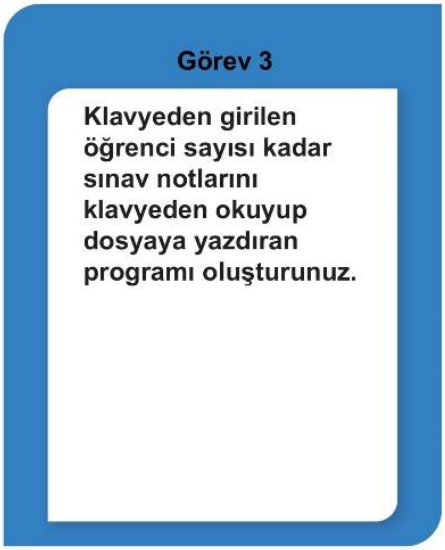
`#include <iostream>`
`#include <fstream>`
 	
`using namespace std;`
`int main()`
`{`
`ofstream dosya;`
`dosya.open("deneyap.txt");`
`dosya << "Merhaba Deneyap!" << endl;`
`dosya.close();`
`}`
	`#include <iostream>`
	`#include <fstream>`
	
	`using namespace std;`
	`int main()`
	`{`
	`ofstream dosya;`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

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

L7.B6.1. 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 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. E-Posta Adresinde İsim Soyisim Biçimlendirme Programı (45 dk.)

Ders Arası (10 dk.)

B2. Mesajlaşma Uygulaması Hazırlıyorum (25 dk.)

B3. Market Alışverişi Yapıyorum (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. E-Posta Adresinde İsim Soyisim Biçimlendirme Programı

Süre: 45 dk.

Materyaller: LÇ7.B1.1. Öğrenci Yönerge Metni

LÇ7.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 LÇ7.B1.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir kullanıcı e-posta adresinde adı ve soyadı "Ahmet BAYRaKtAR" olacak şekilde yazıldığını farketmiştir. Doğru biçimlendirme için soyadının yalnızca ilk harfi büyük, geri kalan harfleri küçük olacak şekilde düzenleyen bir program yazmak istemektedir. Bunun için nasıl bir kodlama yapmalıdır?

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.
- **25-45 dakika:** Kodların çevrim için platformda veya padlet ortamında paylaşılması, eğitmenin geri bildirim vermesi ve örnek çözümün eğitmen tarafından sunulması.

B2. Mesajlaşma Uygulaması Hazırlıyorum

Süre: 25 dk.

Materyaller: LÇ7.B2.1. Öğrenci Yönerge Metni

LÇ7.B2.2. 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 LÇ7.B2.1. Öğrenci Yönerge Metni'ne uygun bir C++ programı yazacaklardır: Bir mesajlaşma uygulamasında, kullanıcı "Bugun hava cok guzel. Azicik disari cikalim" adlı mesajın içinde yer alan herhangi bir harfin kaç kez geçtiğini merak etmektedir. Bunun için metin içerisinde istenilen harfi sayan bir fonksiyon yazınız.

"Not: Karakter olarak istediğiniz bir değer gönderebilirsiniz..."

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. Market Alışverişi Yapıyorum

Süre: 25 dk.

Materyaller: LÇ7.B2.1. Öğrenci Yönerge Metni

LÇ7.B2.2. 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: 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.

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.

Hafta 7. Çevrim İçi: Ders Materyalleri

LÇ7.B1.1. Öğrenci Yönerge Metni

Göreviniz:

Bir kullanıcı e-posta adresinde adı ve soyadı "Ahmet BAYRaKtAR" olacak şekilde yazıldığını farketmiştir. Doğru biçimlendirme için soyadının yalnızca ilk harfi büyük, geri kalan harfleri küçük olacak şekilde düzenleyen bir program yazmak istemektedir. Bunun için nasıl bir kodlama yapmalıdır?

LÇ7.B1.2. Kod Bloğu

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char mesaj[] = "Ahmet BAYRaKtAR";
    int kelimeSayisi = 0;
    bool bosluk = false;
    for(int i=0; i<strlen(mesaj);i++)
    {
        if(bosluk)
            mesaj[i] = tolower(mesaj[i]);
        if(mesaj[i] == ' ')
        {
            bosluk = true;
            mesaj[i+1] = toupper(mesaj[i+1]);
            i++;
        }
    }
    cout << mesaj ;
 return 0;
}
```

LÇ7.B2.1. Öğrenci Yönerge Metni

Göreviniz: Bir mesajlaşma uygulamasında, kullanıcı “Bugun hava cok guzel. Azicik disari cikalim” adlı mesajın içinde yer alan herhangi bir harfin kaç kez geçtiğini merak etmektedir. Bunun için metin içerisinde istenilen harfi sayan bir fonksiyon yazınız.

“Not: Karakter olarak istediğiniz bir değer gönderebilirsiniz...”

LÇ7.B2.2. Kod Bloğu

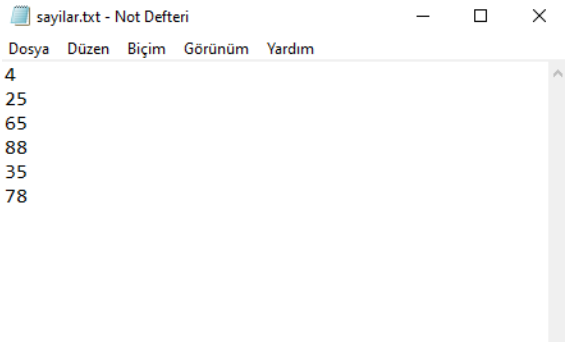
```
#include <iostream>
#include <cstring>
using namespace std;
int karakter_say(char metin[], char karakter)
{
    int sayac = 0;
    for(int i=0;i<strlen(metin);i++)
    {
        if(tolower(metin[i])==karakter)
            sayac++;
    }
    return sayac;
}
int main()
{
    char mesaj[] = "Bugun hava cok guzel. Azicik disari cikalim.";
    char karakter = tolower('B');
    cout << "Bu cumlede "<< karakter_say(mesaj,karakter) <<" adet "<<
karakter <<  " vardir.";
    return 0;
}
```

LÇ7.B3.1. Öğrenci Yönerge Metni

Göreviniz:

Bir market alışverişinde ürün fiyatlarının bir dosyada saklandığını varsayalım. Bu dosyadan fiyatları okuyarak toplam alışveriş tutarını ekrana yazdıran bir program yazınız.

Not: Kodu yazmadan önce "sayilar.txt" isimli bir dosya oluşturarak içerisine ürün fiyatları ekleyiniz.



LÇ7.B3.2. Kod Bloğu

```
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <string>
using namespace std;
int main()
{
    int toplam = 0;
    ifstream dosya("sayilar.txt");
    string satir;
    if(dosya.is_open())
    {
        cout << "Sayilar:"<<endl;
        while(getline(dosya,satir))
        {
            toplam += atoi(satir.c_str());
            cout << satir << endl;
        }
        cout << "Toplam: "<< toplam;
    }
    else
        cout << "dosya Okunamadi!";
}
```

Ekran Çıktısı: