Todor Dimov

BİLGİSAYAR-BİLİŞİM
II. (ikinci)
Sınıf Lise Eğitimi

Ulusal Ders Kitapları Komisyonunun 22.07.2026 tarihli ve 26-843/1 sayılı Kararı ile lise eğitiminde II. (ikinci) sınıf Bilgisayar-Bilişim dersi ders kitabı olarak onaylanmış ve kullanıma sunulmuştur.

Üsküp, 2026

BİLGİSAYAR-BİLİŞİM

II. (ikinci) Sınıf Lise Eğitimi

назив на оригинал:
Информатика
за II (втора) година гимназиско образование

Yayınevi:
ÜTHŞ «STUDENTSKİ SERVİS» Ltd. Şti. – Üsküp

Yazar:
Todor Dimov

Hakemler
Prof. Dr. Marija Kalendar
Prof. Dr. Goran Jakimovski
Jaklina Miloşeva

Editör:
Simona Ristovska

Dil Editörü:
Doç. Dr. Melek Nuredini

Türkçeye Çeviren:
Doç. Dr. Melek Nuredini

Tasarım ve Teknik Hazırlık:
ÜTHŞ «STUDENTSKİ SERVİS» Ltd. Şti. – Üsküp
Baskı:

ÜTHŞ «STUDENTSKİ SERVİS» Ltd. Şti.
Pero Nakov Caddesi No:112 – Üsküp

Baskı Adedi:
200 adet

CIP - Каталогизација во публикација
Национална и универзитетска библиотека «Св. Климент Охридски», Скопје

004(075.3)=512.161

DIMOV, Todor
Bilgisayar-Bilişim II. (ikinci) Sınıf lise Eğitimi / Todor Dimov ; [türkçeye Çeviren Melek Nuredini]. - Üsküp : Studentski servis, 2026. - 330 стр. : илустр. ; 26 см

Превод на делото: Информатика за II (втора) година гимназиско образование

ISBN 978-608-4777-52-6

COBISS.MK-ID 69472773

İÇİNDEKİLER

Teknolojinin günlük yaşamın ayrılmaz bir parçası olduğu bir çağda yaşamaktayız. Bilgisayarlar, internet, mobil cihazlar ve dijital hizmetler, daha hızlı iletişim kurmamıza, öğrenmemize, çalışmamıza ve üretmemize olanak sağlamaktadır. Tüm bu teknolojilerin temelinde bilişim alanındaki bilgi ve beceriler bulunmaktadır. Bilişim günümüzde yalnızca bilgisayar kullanımı anlamına gelmemektedir. Bilişim, çağdaş teknolojilerin işleyiş biçiminin anlaşılmasını, verilerin işlenmesini ve düzenlenmesini, program geliştirilmesini, dijital içeriklerin oluşturulmasını ve bilgisayar sistemleri aracılığıyla problemlerin çözülmesini kapsamaktadır.

Bu ders kitabı, lise eğitiminin ikinci yılına yönelik Bilişim dersi öğretim programına uygun olarak hazırlanmış olup öğrencilerin dijital topluma başarılı bir şekilde katılım için gerekli olan güncel bilgi ve uygulamalı becerileri edinmelerine yardımcı olmayı amaçlamaktadır. Ders kitabındaki içerikler, bilişim alanındaki çağdaş teknolojileri, programlamayı, veri tabanlarını, çoklu ortamı, bilgisayar grafiklerini ve web geliştirmeyi kapsamaktadır. Bilgilerin pratikte uygulanmasına özel önem verilmiş, örnekler, görevler ve öğrencileri düşünmeye, araştırmaya ve kendi çözümlerini geliştirmeye teşvik eden etkinliklere yer verilmiştir.

Ders kitabının amacı yalnızca yeni bilgiler sunmak değil, aynı zamanda mantıksal düşünme, yaratıcılık, dijital okuryazarlık ve yaşam boyu öğrenmeye hazırlık becerilerini geliştirmektir. Teknolojinin sürekli değiştiği bir dönemde en önemli beceri, öğrenebilmek, değişen koşullara uyum sağlayabilmek ve bilgiyi yeni durumlarda uygulayabilmektir.

Bu ders kitabının bilişim dünyasını keşfetme sürecinde yararlı bir rehber olmasını ve öğrencileri araştırmaya, üretmeye ve kendi fikirlerini geliştirmeye devam etmeleri konusunda motive etmesini umuyoruz. Bilişim yalnızca sınıfta öğrenilen bir ders değildir. Bilişim, çağdaş dünyanın dilidir ve geleceği şekillendirmek için bir araçtır.

KONU 1

1.1

BİLİŞİMDE ÇAĞDAŞ TEKNOLOJİLER VE DİJİTAL DÖNÜŞÜM

Değişimlerin büyük bir hızla gerçekleştiği bir çağda yaşamaktayız ve bu değişimlerin başlıca itici gücü bilişimdir. Günümüzde "teknoloji" terimi artık yalnızca sıradan bilgisayarları ve temel programları ifade etmemekte, çalışma, iletişim kurma ve yaşam biçimimizi köklü bir şekilde değiştiren karmaşık bir yenilikler sistemini ifade etmektedir.

Bilişimde çağdaş teknolojiler kavramı, büyük miktardaki verilerin hızlı bir şekilde işlenmesini, depolanmasını ve analiz edilmesini sağlayan en yeni ve gelişmiş araçları, donanım aygıtlarını, yazılım çözümlerini ve ağ sistemlerini kapsamaktadır. Bu teknolojiler, geçmişteki basit ve statik veri tabanı yönetiminden etkin ve akıllı bir ekosisteme geçişi temsil etmektedir. Söz konusu teknolojiler, büyük miktarlardaki verilerin analiz edilmesine ve gerçek zamanlı olarak akıllı kararlar alınmasına olanak sağlamakta, dijital dünya ile fiziksel dünyayı birbirine bağlamaktadır.

Çağdaş teknolojiler sürekli gelişmekte ve insan yaşamının tüm alanlarına entegre olmaktadır. Bu durum, dijital dönüşüm olarak adlandırılan olgunun ortaya çıkmasını sağlamaktadır. Dijital dönüşüm, dijital teknolojinin bir toplumun eğitim, ekonomi, üretim veya iş dünyası gibi tüm alanlarına entegre edilmesi ve bu alanların işleyiş biçimini temelden değiştirmesi sürecidir.

Bilişim dünyası günümüzde birkaç temel sütun tarafından yönlendirilmektedir:

- Yapay zekâ (AI) ve Büyük Dil Modelleri (LLM'ler): İnsanlara benzer düzeyde metin, kod ve fikirleri işleyebilen ve üretebilen gelişmiş sistemlerdir (ör. GPT modelleri, Gemini, Claude). Bu sistemler milyarlarca veriyi birkaç saniye içinde analiz edebilmektedir.
- İleri robotik: Bilişim, yapay zekâ ve makine mühendisliğini bir araya getiren teknolojidir. Bunlar, karmaşık fiziksel görevleri yerine getirebilen, çevreden öğrenebilen ve gerçek zamanlı olarak değişikliklere tepki verebilen otonom veya yarı otonom makinelerdir.
- Nesnelerin İnterneti (IoT): İnternete bağlı olan ve kendi aralarında iletişim kuran fiziksel cihazlardan (akıllı telefonlar, ev aletleri, endüstriyel sensörler) oluşan bir ağdır.
- Bulut teknolojileri (Cloud Computing): Verilerin yerel bir bilgisayar yerine uzak sunucularda depolanmasını ve işlenmesini sağlayan teknolojilerdir (ör. Google Drive, Netflix).

Bu teknolojilerin günlük yaşamımızdaki uygulamaları oldukça yaygındır. Bunların uygulanma olanakları pratik olarak sınırsızdır. Bu teknolojilerin pratik uygulamaları en yaygın olarak şu alanlarda görülmektedir:

- Yaratıcı çalışma ve kişisel asistanlar. Büyük Dil Modellerinin yardımıyla günümüzde herkes anında erişilebilen bir "dijital beyne" sahiptir. Bu modeller, kompozisyon yazma, dilleri gerçek zamanlı olarak çevirme, yeni beceriler öğrenme ve bilgisayar kodu oluşturma konularında yardımcı olmaktadır.
- Evlerde ve endüstride akıllı otomasyon. Robotik ve IoT aracılığıyla günlük yaşamda, evi kendi kendine haritalandıran robotik elektrikli süpürgeler kullanmaktayız. Endüstride ise robotik kollar hassas işlemleri, paketlemeyi ve insanlar açısından tehlikeli fiziksel görevleri yerine getirerek insanlara yönelik riski ortadan kaldırmaktadır.

- Eğitim ve tıp. Büyük Dil Modelleri, açıklamaları öğrencinin bilgi düzeyine göre uyarlayan kişisel öğretmenler olarak kullanılmaktadır. Tıp alanında ise (Da Vinci sistemi) gibi cerrahi robotlar, doktorların uzaktan son derece hassas ameliyatlar gerçekleştirmesine olanak sağlamaktadır.
- Elektronik hizmetler (E-hizmetler). Tek tıklamayla fatura ödeme, çevrim içi bankacılık ve yapay zekâ sayesinde 7/24 hizmet veren akıllı "sohbet botları" aracılığıyla müşteri desteği sağlanmaktadır.
- Ulaşım, en uygun güzergâhın seçilmesine yardımcı olan navigasyon sistemlerinin kullanıldığı alandır.
- Tarım, hava koşullarının izlenmesi ve sulamanın gerçekleştirilmesi amacıyla akıllı sistemlerin uygulandığı alandır.

1.1.1 AVANTAJLAR, ZORLUKLAR VE TOPLUM ÜZERİNDEKİ ETKİ

Her büyük devrimde olduğu gibi, dijital dönüşüm de beraberinde ciddi faydalar ve toplumun üstesinden gelmesi gereken zorluklar getirmektedir.

Olumlu Yönler	Olumsuz yönler
Verimlilik ve üretkenlik: Büyük Dil Modelleri ve robotlar, manuel ve tekrarlayan görevleri üstlenerek insanın yaratıcı düşünmeye daha fazla zaman ayırmasını sağlar.	İşlerin kaybedilmesi: Otomasyon, robotik ve yapay zekâ, belirli mesleklerin (fabrika işçilerinden yöneticilere kadar) yerini alabilir ve bu durum hızlı bir şekilde yeniden mesleki eğitim alınmasını gerektirir.
İş güvenliği: Robotlar, insan hayatının tehlikeye girdiği tehlikeli ortamlarda (madenler, derin denizler, nükleer santraller) çalışabilir.	Bağımlılık ve bilişsel tembellik: Yazma ve düşünme süreçlerinde dil modellerine aşırı güvenilmesi, özellikle gençlerde eleştirel düşünme ve yaratıcılık becerilerinin azalmasına neden olabilir.
Bilgiye erişim ve kapsayıcılık: Gelişmiş LLM çevirmenleri sayesinde dil engelleri ortadan kalkmakta ve bilgilerin herkes tarafından erişilebilir olması sağlanmaktadır.	Dezenformasyonun yayılması: Büyük Dil Modelleri, ikna edici görünen yanlış bilgileri (sözde "halüsinasyonlar") istemeden üretebilir veya propaganda amacıyla kötüye kullanılabilir.
Daha iyi tanı ve bakım: Yapay zekâ ile tıbbi robotik teknolojilerinin bir araya gelmesi, insanların daha uzun ve daha kaliteli bir yaşam sürmesine olanak sağlamaktadır.	Siber güvenlik ve etik sorunlar: Robotun bir hata yapması durumunda kim sorumlu olacaktır? Dil modellerinin beslendiği kişisel veriler nasıl korunmalıdır?

Çağdaş teknolojilerin günlük yaşamda akılcı bir şekilde uygulanması gerekmektedir. Aşağıdaki örnekler, teknolojilerin yararlı bir şekilde nasıl kullanılabileceğini açıklamaktadır.

- Yaşam boyu eğitim için uyarlanabilir öğrenme. Klasik kurslar yerine, yeniden mesleki eğitim almak veya yeni beceriler (diller, programlama) öğrenmek amacıyla, öğrenme hızına uyum sağlayan bulut tabanlı açık eğitim platformlarının kullanılması daha pratiktir.

- Sağlığın akıllı yönetimi (IoT ve Teletıp). Uyku, nabız ve fiziksel aktiviteyi izlemek amacıyla akıllı saatlerin ve uygulamaların kullanılması ve bu verilerin rutin kontroller için fiziksel ziyaretler yerine dijital platformlar aracılığıyla aile hekimiyle paylaşılması.

- Yapay zekâ destekli araçlarla rutin görevlerin otomasyonu. İş e-postalarının hızlı bir şekilde hazırlanması, toplantı notlarının yapılandırılması veya büyük metin arşivlerinde arama yapılması gibi görevlerde yapay zekâ araçlarının "bilişsel asistanlar" olarak kullanılması ve böylece yaratıcı problem çözmeye daha fazla zaman ayrılması.

- Dijital cihaz kullanımının bilinçli olarak sınırlandırılması. Dikkat süresinin azalmasına karşı bir önlem olarak, çalışma saatlerinde sosyal medya erişimini sınırlayan uygulamaların kullanılması ve böylece odaklanarak çalışma becerisinin korunması.

SONUÇ:

Bilişim alanındaki çağdaş teknolojiler, fiziksel dünyayı dönüştüren robotikten entelektüel emeği dönüştüren Büyük Dil Modellerine kadar güçlü araçlardır. Bunların toplum üzerindeki etkisi, tamamen onları nasıl kullandığımıza bağlıdır. Bu teknolojilerin büyük potansiyeli, yaşamı kolaylaştırmak amacıyla kullanılmalı, aynı zamanda olumsuz sonuçlardan korunmak için eleştirel düşünme ve etik kurallara ilişkin farkındalık geliştirilmelidir.

BİLİYOR MUSUNUZ?

Biliyor musun, günümüzde bir akıllı telefon, Ay'a gerçekleştirilen ilk uzay görevlerinde kullanılan bilgisayarlardan daha yüksek işlem gücüne sahiptir?

BİLGİ KONTROLÜ SORULARI

1. Bilişimde "çağdaş teknolojiler" kavramı nasıl tanımlanır?
2. Günümüzde dünyada en yaygın kullanılan üç çağdaş teknoloji örneğini sıralayınız.
3. Çağdaş teknolojiler ile dijital dönüşüm süreci arasındaki ilişkiyi kendi sözlerinizle açıklayınız.
4. Nesnelerin İnterneti (IoT) nasıl çalışır ve günlük yaşamımızı internete nasıl bağlar?
5. Büyük Dil Modelleri neden bazen "halüsinasyon" olarak adlandırılan yanlış bilgiler üretir ve bu ne anlama gelir?
6. Gün içerisinde sizin veya akıllı evinizin IoT teknolojisini kullandığı bir durumu açıklayınız.
7. Robotların fabrikalarda yaygın olarak kullanılmasının olumlu ve olumsuz yönlerini karşılaştırınız. Kim kazanır, kim kaybeder?
8. Bulut teknolojilerinin günümüzde verilerin saklanma ve paylaşılma biçimi üzerindeki etkisini geçmişteki yöntemlerle (sabit diskler, USB bellekler) karşılaştırarak analiz ediniz.
9. Bir tıbbi robot ameliyat sırasında hata yaparsa etik ve hukuki sorumluluğu kim üstlenmelidir — programcı, mühendis veya doktor? Görüşünüzü gerekçelendiriniz.
10. Yapay zekânın hızlı gelişiminin insanlarda sosyal izolasyona ve yaratıcılığın

azalmasına mı yol açtığını, yoksa verimliliği mi artırdığını değerlendiriniz.

11. Okulunuzda veya şehrinizdeki gerçek bir sorunu çözebilecek, IoT veya yapay zekâ tabanlı yeni bir "akıllı cihaz" fikri öneriniz.
12. Geleneksel olarak kâğıt üzerinde çalışan bir işletmenin güvenli bir şekilde dijital dönüşüm gerçekleştirmesi için kısa bir eylem planı (3–4 adım) yazınız.

ARAŞTIRMA VE UYGULAMA ÇALIŞMALARI

1. Bir gün boyunca kullandığınız on çağdaş teknolojinin listesini hazırlayınız. Her bir teknolojinin hangi alanda kullanıldığını ve günlük yaşamınızı nasıl kolaylaştırdığını belirtiniz.
2. Bir kamu kurumu (okul, kütüphane, banka, hastane veya belediye) seçiniz ve vatandaşlara sunduğu dijital hizmetleri araştırınız. Kısa bir rapor hazırlayınız.
3. Yirmi yıl önceki bir okul gününün nasıl olduğunu düşününüz ve günümüzle karşılaştırınız. Çağdaş teknolojilerin sonucunda ortaya çıkan en az beş değişikliği belirtiniz.
4. Sınıf arkadaşlarınızla bir grup oluşturarak akıllı telefonların günlük yaşamda kullanılmasının avantajlarını ve olası dezavantajlarını tartışınız. Sonuçları yazınız.
5. Gelecek üzerinde büyük bir etkisi olacağını düşündüğünüz bir çağdaş teknolojiyi araştırınız. Nasıl çalıştığını ve hangi alanlarda uygulanabileceğini açıklayan kısa bir sunum hazırlayınız.

SONUÇ:

Çağdaş teknolojiler, günlük yaşamı kolaylaştıran çağdaş teknik ve bilişim çözümleridir.

Dijital dönüşüm, dijital teknolojilerin toplumun çeşitli alanlarına uygulanması sürecidir.

Çağdaş teknolojiler, eğitim, sağlık, sanayi, ulaşım, ticaret ve iletişim alanlarında kullanılmaktadır.

Çağdaş teknolojilerin kullanımı, çok sayıda yarar sağlamakla birlikte bazı zorlukları da beraberinde getirmektedir.

Çağdaş teknolojilerin sorumlu ve güvenli kullanımı, kişisel verilerin ve dijital gizliliğin korunmasına katkıda bulunmaktadır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Belirli bir uygulamanın "bir sonraki anda ne görmek istediğinizi nasıl bildiği?" sorusunun yanıtını bulmak için araştırınız. Bu sorunun yanıtını keşfetmek, ileri bilişim dünyasında geleceğinize açılan kapıyı aralamaktadır.

1.2

SİBER GÜVENLİK VE KİŞİSEL VERİLERİN KORUNMASI

Siber güvenlik, bilgi güvenliğinin giderek daha önemli hâle geldiği çağdaş dijital dünyada temel bir unsurdur. İnternetin ve dijital hizmetlerin kullanımının sürekli artmasıyla birlikte kişisel veriler çeşitli risklere ve kötüye kullanımlara maruz kalmaktadır. Bu nedenle, gizliliğin korunması ve verilerin uygun şekilde güvence altına alınması, her birey ve kuruluş için büyük önem taşımaktadır. Bilgisayar korsanı saldırıları, virüsler ve kimlik avı gibi dijital tehditler, bilgi güvenliği açısından ciddi bir tehlike oluşturmaktadır. Bu tehditlerin anlaşılması ve temel koruma ilkelerinin uygulanması, dijital teknolojilerin daha güvenli bir şekilde kullanılmasını sağlamaktadır. Bu nedenle siber güvenlik, çağdaş toplumda yaşamın sürdürülebilmesi için gerekli bir beceri hâline gelmiştir.

BU KONUYU TAMAMLADIĞINIZDA...

- Bilgi aramak için interneti kullanmayı;
- Dijital platformlar aracılığıyla iletişim kurmayı;
- Farklı kullanıcı hesaplarını kullanmayı;
- Dijital içerikleri paylaşmayı;
- Kişisel verilerin korunması gerektiğini fark etmeyi bileceksiniz.

DÜŞÜNME SORULARI

1. Bir hafta içerisinde kaç kullanıcı hesabı kullanıyorsunuz?
2. İnterneti kullanırken hangi kişisel verilerinizi paylaştığınızı hiç düşündünüz mü?
3. Tanımadığınız bir kişi sizden şifrenizi veya kişisel bilgilerinizi isterse nasıl davranırsınız?
4. İnternette yayımlanan tüm bilgilerin başkalarıyla paylaşılması gerekir mi?
5. Dijital hizmetleri kullanırken dikkatli olmak neden önemlidir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Kişisel ve dijital verilerin korunmasının neden önemli olduğunu açıklarsınız.
- Temel koruma önlemlerini (güçlü parolalar, antivirüs programları, internetin dikkatli kullanımı) açıklarsınız.
- Verilen bir durumda internetin güvenli kullanımına ilişkin kuralları uygularsınız (sosyal medya veya e-posta kullanırken verilerinizi korumak amacıyla).
- Güvenli ve güvensiz çevrim içi davranışları ayırt eder ve verilerin uygun şekilde korunmamasının olası sonuçlarını analiz edersiniz.
- Kişisel dijital güvenliğin geliştirilmesine yönelik yöntemler önerirsiniz.

Çağdaş bilişim teknolojileri hızlı iletişimi, çok sayıda bilgiye erişimi ve çeşitli dijital hizmetlerin kullanımını mümkün kılmaktadır. Günlük yaşamda elektronik posta, sosyal ağlar, öğrenme platformları, elektronik bankacılık ve çeşitli mobil uygulamalar kullanılmaktadır. Bunların kullanımı

sırasında ad ve soyadı, telefon numarası, elektronik posta adresi, parolalar ve diğer bilgiler gibi kişisel veriler sıklıkla girilmektedir. Bu nedenle dijital teknolojilerin güvenli kullanımı her kullanıcı için giderek daha önemli hâle gelmektedir.

Çağdaş teknolojilerin gelişmesiyle birlikte veri güvenliğini ve kullanıcıların gizliliğini etkileyebilecek çeşitli riskler de ortaya çıkmaktadır. Kişisel bilgilerin çalınmasına, kullanıcı hesaplarına yetkisiz erişim sağlanmasına ve zararlı yazılımların yayılmasına yönelik sürekli girişimler bulunmaktadır. Bu riskleri azaltmak için siber güvenliğin temel kurallarının bilinmesi ve dijital teknolojilerin günlük kullanımında kişisel verilerin korunmasına yönelik önlemlerin uygulanması gerekmektedir.

1.2.1 SİBER GÜVENLİK NEDİR VE NEDEN ÖNEMLİDİR?

İnternete her bağlandığında, ister sosyal ağları görüntülüyor, ister e-posta gönderiyor, ister çevrim içi oyunlar oynuyor ol, aslında dijital dünyaya açılan kapıyı aralamış olursun. Ancak hırsızlardan korunmak için evinin kapısını kilitlediğin gibi, dijital alanının güvenliğini de sağlaman gerekir.

Siber güvenlik, bilgisayar sistemlerinin, ağların ve dijital verilerin çeşitli tehditlerden ve kötüye kullanımlardan korunmasını sağlayan önlemler, kurallar ve uygulamalar bütünüdür. Günümüz toplumunda çok sayıda faaliyet bilişim teknolojilerine ve internete bağlıdır. Elektronik posta, sosyal ağlar, dijital öğrenme platformları, elektronik bankacılık ve çeşitli mobil uygulamaların kullanılması, dijital ağlar üzerinden sürekli bilgi alışverişini gerektirmektedir. Bu nedenle, söz konusu sistemlerin güvenliği, kullanıcıların ve verilerinin korunmasında önemli bir rol oynamaktadır.

Siber güvenlik yalnızca büyük şirketlerin ve kurumların korunmasıyla ilgili değildir, aynı zamanda her bireysel kullanıcıyı da ilgilendirmektedir. Akıllı telefonların, tabletlerin ve bilgisayarların günlük kullanımı, kişisel bilgilerin korunması konusunda belirli bir sorumluluk gerektirmektedir. Uygun önlemlerin uygulanmasıyla verilere yetkisiz erişim, kimlik hırsızlığı veya kullanıcı hesaplarının kötüye kullanılması gibi riskler azaltılabilir. Siber güvenliğin temel ilkelerinin bilinmesi, çağdaş teknolojilerin daha güvenli kullanılmasına katkıda bulunmaktadır.

1.2.2 KİŞİSEL VERİLER VE DİJITAL MAHREMİYET

Kişisel veriler, bir kişinin kimliğinin belirlenmesini sağlayabilecek bilgilerdir. Bu gruba ad ve soyadı, ikamet adresi, telefon numarası, elektronik posta adresi, kimlik numarası, fotoğraflar ve belirli bir kullanıcıyla ilişkilendirilebilecek diğer birçok bilgi dâhildir. Dijital toplumda internet hizmetleri, mobil uygulamalar ve elektronik platformlar kullanılırken her gün çeşitli kişisel veriler paylaşılmaktadır. Bu nedenle, kişisel verilerin korunması dijital güvenliğin önemli bir parçasıdır.

Dijital mahremiyet, her kullanıcının hangi bilgileri paylaşacağına ve bu bilgileri kiminle paylaşacağına kendisinin karar verme hakkıdır. İnternet kullanılırken sosyal ağlarda veya diğer dijital platformlarda hangi verilerin paylaşılacağı dikkatle seçilmelidir. Kişisel bilgilerin aşırı paylaşılması çeşitli kötüye kullanımlara yol açabilir ve kullanıcının güvenliğini tehlikeye atabilir. Dijital hizmetlerin sorumlu bir şekilde kullanılması, mahremiyetin daha iyi korunmasına ve kişisel verilerin kötüye kullanılma olasılığının azaltılmasına katkıda bulunmaktadır.

1.2.3 İNTERNETIN GÜVENLI KULLANIMINA İLIŞKIN KURALLAR

İnternette güvende olmak için ileri düzeyde bilişim bilgisine sahip olmak gerekmez. Dijital hijyenin üç altın kuralını uygulamak yeterlidir:

- **Güçlü parolalar oluşturmak:** Parola, ilk savunma hattıdır. Zayıf bir parola (örneğin 123456 veya ime123) bilgisayar korsanları tarafından birkaç saniye içinde ele geçirilebilir. Güçlü bir parola en az 12 karakterden oluşmalı, büyük ve küçük harfler, rakamlar ve özel karakterlerin (ör. @, #, $, %) bir kombinasyonunu içermelidir. Ayrıca, farklı hesaplar için aynı parola kesinlikle kullanılmamalıdır.
- **Antivirüs yazılımı kullanmak ve güncel tutmak:** Antivirüs programları, cihazı tarayan, şüpheli yazılımları (virüsler, truva atları, casus yazılımlar) tespit eden ve zarar vermeden önce engelleyen dijital koruyucular olarak işlev görür. Yeni tehditleri tanıyabilmesi için bu yazılımın düzenli olarak güncellenmesi önemlidir.
- **İnterneti dikkatli kullanmak:** Bu, internette nereye tıkladığınızın farkında olmayı gerektirir. Bilinmeyen web sitelerinden dosya indirmekten kaçınınız, hızlı kazanç veya ücretsiz ödüller vaat eden şüpheli reklamlara tıklamayınız ve hassas verileri girerken halka açık, güvenli olmayan Wi-Fi ağlarına (kafeler veya parklardaki ağlar gibi) bağlanmayınız.

1.2.4 EN YAYGIN DİJİTAL TEHDİTLER VE RİSKLER

İnternetin ve çağdaş teknolojilerin gelişmesiyle birlikte kullanıcıların güvenliğini etkileyebilecek çeşitli dijital tehditler ortaya çıkmaktadır. En yaygın tehditler arasında sahte mesajlar ve internet dolandırıcılıkları, zararlı yazılımlar, kullanıcı hesaplarına yetkisiz erişim ve kişisel verilerin çalınması yer almaktadır. Bazı durumlarda saldırganlar kullanıcıların güvenini kazanmaya ve onları parolalarını veya diğer hassas bilgilerini kendi istekleriyle açıklamaya yönlendirmeye çalışmaktadır. Bu tür girişimler çoğunlukla elektronik posta, mesajlar veya sahte internet siteleri aracılığıyla gerçekleştirilmektedir.

Doğrudan saldırıların yanı sıra dijital teknolojilerin dikkatsizce kullanılmasından kaynaklanan riskler de bulunmaktadır. Zayıf parolaların kullanılması, doğrulanmamış kaynaklardan dosya indirilmesi veya şüpheli mesajların açılması, bilgisayar sistemlerinin ve kişisel verilerin güvenliğinin tehlikeye girmesine yol açabilir. Olası tehditlerin tanınması ve internet hizmetlerinin dikkatli bir şekilde kullanılması, dijital kimliğin korunması ve olası risklerin azaltılması açısından önemli bir adımdır.

1.2.5 İNTERNETİN GÜVENLİ KULLANIMINA İLİŞKİN KURALLAR

İnternetin güvenli kullanımı, kullanıcıların ve verilerinin korunmasına katkıda bulunan çeşitli kuralların ve alışkanlıkların uygulanmasını gerektirir. Temel önlemlerden biri, harf, rakam ve özel karakterlerin birleşiminden oluşan güçlü parolaların kullanılmasıdır. Aynı parolanın birden fazla farklı hizmette kullanılmaması ve parolaların düzenli olarak değiştirilmesi yararlıdır. Kullanıcı hesabına erişim için ek bir doğrulama gerektiren çok faktörlü kimlik doğrulamanın kullanılması da ek güvenlik sağlamaktadır.

İnternetin sorumlu kullanımı, bilgi paylaşırken ve dijital hizmetleri kullanırken dikkatli davranmayı da gerektirir. İnternet sitelerinin güvenilirliği kontrol edilmeli, elektronik postalar dikkatli bir şekilde açılmalı ve kişisel veriler tanımadık kişilerle paylaşılmamalıdır. İşletim sistemlerinin ve programların düzenli olarak güncellenmesi, yeni güvenlik tehditlerine karşı daha iyi koruma sağlamaktadır. İyi dijital alışkanlıkların ve eleştirel düşünme becerisinin geliştirilmesiyle kullanıcılar, çağdaş bilişim teknolojilerini kullanırken güvenliklerini önemli ölçüde artırabilirler.

Siber güvenliği anlamanın en iyi yolu, onu günlük yaşamda karşılaştığımız durumlarda uygulamaktır.

Örnek 1: Sosyal ağların kullanımı (TikTok, Instagram, Snapchat...)

Sosyal ağlar, çok fazla bilginin kolayca paylaşılabildiği ortamlardır. Bu durumda verilerin korunması için aşağıdaki kurallar uygulanmalıdır:

- **Gizli profil (Private Account): Gi**zlilik ayarlarını, yalnızca kabul ettiğin arkadaşlarının paylaşımlarını, fotoğraflarını ve konumunu görebileceği şekilde değiştir.
- **Konum paylaşımında ("check-in") dikkatli olunuz:** Gerçek zamanlı olarak kesin konumunuzu paylaşmayınız. Bu, kötü niyetli kişilere tam olarak nerede bulunduğunuzu (veya o anda evde olmadığınızı) gösterebilir.
- **Paylaşmadan önce düşününüz:** Her fotoğraf veya bilgi (örneğin kimlik kartının fotoğrafı, ev adresi veya okul bilgisi), silseniz bile internette kalıcı olarak bulunabilir.

Örnek 2: Elektronik posta (e-posta) kullanımı

E-posta, saldırganların kendilerini yasal bir kuruluş (banka, sosyal ağ, posta hizmeti vb.) gibi **göst**ererek parolalarınızı veya banka kartı bilgilerinizi çalmaya çalıştıkları kimlik avı (phishing) saldırılarının en yaygın hedeflerinden biridir. Bu durumda korunmak için aşağıdaki kurallar uygulanmalıdır:

1. **Bilinmeyen göndericileri dikkate almama**k: Bilinmeyen bir adresten acil yanıt vermenizi isteyen bir e-posta alırsanız (örneğin "Hesabınız silinecek, parolanızı doğrulamak için buraya tıklayın"), e-postayı hemen siliniz.
2. **Bağlantıları ve ekli dosyaları kontrol etmek:** Tanımadığınız kişilerden gelen ve .exe veya .zip gibi uzantılara sahip ekli dosyaları kesinlikle açmayınız, bunlar bilgisayarınıza otomatik olarak virüs yükleyebilir.

Örnek 3: İki faktörlü kimlik doğrulamayı (2FA) etkinleştirme: Bu, ek bir güvenlik katmanı sağlar. E-posta hesabına giriş yapılırken sistem, parolanın yanı sıra cep telefonunuza SMS yoluyla gönderilen bir kodu da ister. Böylece bir kişi parolanızı öğrenmiş olsa bile cep telefonunuza erişmeden hesabınızı kullanamaz.

1.2.6 DİJİTAL ÖĞRENCİ VE SİBER GÜVENLİK

Sıradan bir okul günü boyunca öğrenci çok sayıda dijital hizmet kullanır. Öğrenme platformlarına giriş yapar, internette bilgi arar, çeşitli uygulamalar aracılığıyla iletişim kurar ve elektronik posta kullanır. Tüm bu etkinlikler sırasında çeşitli veriler paylaşılır ve hizmetlere erişim için çoğunlukla kullanıcı hesapları ve parolalar kullanılır. Bu hizmetlerin güvenli bir şekilde kullanılması, kişisel verilerin korunmasına ve kötüye kullanım olasılığının azaltılmasına katkıda bulunur.

Bir öğrencinin ödül kazandığına dair bir mesaj alması ve bilinmeyen bir internet sitesine kullanıcı adı, parola ve banka hesabı bilgilerini girmesinin istenmesi buna örnek olarak verilebilir. Gönderen ve internet sitesinin adresi kontrol edilmezse kullanıcı hesabının kötüye kullanılması söz konusu olabilir. Dikkatli davranmak, bilginin kaynağını kontrol etmek ve kişisel verileri tanımadık kişilerle paylaşmaktan kaçınmak, dijital güvenliği artırmak için basit ancak önemli önlemlerdir.

Çevrim içi davranışlarımız, güvensiz internet ortamında karşılaşacağımız sonuçları büyük ölçüde belirler. Dijital dünyada attığımız her adım bir iz bırakır. Günlük yaşamın hızlı temposu nedeniyle çoğu zaman internetin yalnızca bir oyun alanı olmadığını, aynı zamanda gizli risklerle dolu bir ortam olduğunu unuturuz. Güvenli internette gezinme ile olası bir dijital felaket arasındaki fark çoğu zaman kendi kararlarımıza ve alışkanlıklarımıza bağlıdır.

1.2.7 GÜVENLİ VE GÜVENSİZ ÇEVRİM İÇİ DAVRANIŞLAR

Kendinizi korumak için öncelikle riskli durumları tanımayı bilmeniz gerekir. Çevrim içi davranışlar genel olarak iki kategoriye ayrılabilir:

Güvensiz Davranış (Risk)	Güvenli davranış (Koruma)
Sosyal ağlara veya elektronik bankacılığa giriş yapmak için açık, halka açık Wi-Fi ağlarını (parolasız) kullanmak.	Hassas hesaplara erişirken mobil internet (4G/5G) veya güvenli bir ev ağı kullanmak.
Tanımadığınız kişilerden gelen arkadaşlık isteklerini kabul etmek ve onlarla özel bilgilerinizi paylaşmak.	Yalnızca gerçek hayatta tanıdığınız kişilerle iletişim kurmak ve gizliliğinizi korumak.
Ücretsiz ödüller, telefonlar veya "çekilişler" vaat eden cazip bağlantılara tıklamak (Kimlik avı – Phishing).	Şüpheli mesajları görmezden gelmek ve ödüllerle ilgili bilgileri resmî internet sitelerinden kontrol etmek.
Korsan oyun, film veya programları güvenilir olmayan yazılımlardan ve torrent sitelerinden indirip yüklemek.	Yazılımları yalnızca resmî internet sitelerinden ve çevrim içi mağazalardan (Google Play, App Store, resmî siteler) indirmek.

Dikkatsiz davranışların ve yetersiz korumanın sonuçları ciddi ve uzun süreli olabilir. Kişisel verileriniz (parolalar, fotoğraflar, kimlik numaraları) başka kişiler tarafından ele geçirilirse aşağıdaki durumlar ortaya çıkabilir:

- **Kimlik hırsızlığı:** Bilgisayar korsanları verilerinizi ve fotoğraflarınızı kullanarak sahte profiller oluşturabilir, sizin adınıza dolandırıcılık yapabilir veya para talep ederek arkadaşlarınızı ya da aile üyelerinizi şantajla tehdit edebilir.
- **Maddi zarar:** Kimlik avı mesajları veya güvenli olmayan ödeme işlemleri aracılığıyla saldırganlar, banka hesaplarına veya kredi kartlarına erişim sağlayabilir ve hesabınızdaki parayı birkaç dakika içinde çekebilir.
- **Fidye yazılımı (Ransomware) bulaşması:** Cihaza sıradan bir virüs yerine, tüm belgelerinizi, fotoğraflarınızı ve projelerinizi kilitleyen ve bunları geri vermek için sizden fidye talep eden bir yazılım yüklenebilir.
- **Duygusal sağlığın ve itibarın zarar görmesi:** Gizliliğin ihlal edilmesi ve özel konuşmaların veya fotoğrafların kamuya açık şekilde paylaşılması (siber zorbalık), kişinin sosyal yaşamında büyük strese, utanca ve uzun vadeli sonuçlara yol açabilir.

Korunmayı geliştirmek karmaşık teknik beceriler gerektirmez, bunun yerine iyi dijital alışkanlıkların edinilmesi gerekir. Hemen uygulayabileceğiniz bazı somut öneriler şunlardır:

1. **İki faktörlü kimlik doğrulamayı (2FA) etkinleştiriniz:** Bu en önemli adımdır. Instagram, TikTok, Google vb. hesaplarınıza giriş yaparken parolanın yanı sıra telefonunuza gönderilen ek bir kod da gerekecektir. Bir kişi parolanızı bilse bile hesabınıza giriş yapamayacaktır.
2. **Parola yöneticisi (password manager) kullanınız:** Her yerde aynı kolay parolayı kullanmak yerine (bu büyük bir hatadır), her hesabınız için farklı ve karmaşık parolalar oluşturacak ve bunları güvenli bir şekilde saklayacak bir uygulama kullanınız.
3. **Düzenli olarak yedekleme (Backup) yapınız:** Haftada veya ayda bir kez önemli belgelerinizi ve fotoğraflarınızı harici bir sabit diske, USB belleğe veya güvenli bir bulut depolama alanına (Cloud) aktarınız. Böylece cihazınız çalınsa veya bozulsa bile verileriniz güvende kalır ve yeniden kullanılabilir.

4. **Sistemi düzenli olarak güncelleyiniz**: Telefonunuzun veya bilgisayarınızın işletim sistemi için gelen güncelleme bildirimlerini ertelemeyiniz. Bu güncellemeler genellikle bilgisayar korsanlarının saldırılarda kullandığı yeni keşfedilmiş güvenlik açıklarını gideren "güvenlik yamaları" içerir.

BİLİYOR MUSUNUZ?

Güçlü bir parolanın yeterince uzun olması ve büyük-küçük harfler, rakamlar ve özel karakterlerden oluşan bir kombinasyon içermesi gerektiğini biliyor musunuz? Farklı dijital hizmetlerde farklı parolaların kullanılması, kullanıcı hesaplarının güvenliğini önemli ölçüde artırır.

ARAŞTIR

Okulunuzun, arkadaşlarınızın veya ailenizin internetin güvenli kullanımına ilişkin hangi kuralları uyguladığını araştırınız. Elde ettiğiniz sonuçlara dayanarak, kişisel verilerin ve dijital mahremiyetin daha iyi korunmasına katkıda bulunan en az on kuraldan oluşan bir liste hazırlayınız.

SONUÇ:

Siber güvenlik, bilgisayarların karmaşıklığına değil, bizim davranışlarımıza bağlıdır. Güçlü parolalar ve antivirüs yazılımları kullanarak, her tıklamada biraz dikkatli davranarak dijital dünyanın güvenli ve sorumlu kullanıcıları hâline geliriz. İnternet ortamı, bizim ne kadar dikkatli olduğumuz ölçüde güvenlidir. Tehlikelere ilişkin farkındalığımızı geliştirerek ve basit koruma yöntemlerini uygulayarak kendi güvenliğimizin sorumluluğunu üstlenir ve dijital yaşamımızın kontrolünü kendi ellerimize alırız.

BİLGİ KONTROLÜ SORULARI

1. Siber güvenliğin çağdaş toplumda önemli bir rolü olmasının nedeni nedir?
2. Hangi bilgiler kişisel veri olarak kabul edilir?
3. Kullanıcının dijital mahremiyeti nasıl tehlikeye atılabilir?
4. İnternet ve dijital hizmetlerin kullanımında en yaygın riskler nelerdir?
5. İyi dijital alışkanlıklar kişisel verilerin daha iyi korunmasına nasıl katkı sağlar?

ARAŞTIRMA VE UYGULAMA ÇALIŞMALARI

1. İnternette bir kullanıcı hesabı oluştururken en sık kullanılan kişisel verilerin bir listesini hazırlayınız. Bu verilerin korunmasının neden önemli olduğunu açıklayınız.
2. En bilinen internet platformlarının kullanıcı hesaplarını korumak için uyguladığı kuralları ve hangi kişisel verileri topladıklarını ve işlediklerini araştırınız. Kısa bir rapor hazırlayınız.
3. İnternetin dikkatsizce kullanılmasının kişisel verilerin kötüye kullanılmasına yol açabileceği üç durumu düşününüz ve açıklayınız.

4. Sınıf arkadaşlarınızla grup halinde sosyal medyada fotoğraf ve kişisel bilgileri paylaşmanın avantajlarını ve risklerini tartışınız. Sonuçları yazınız.
5. Okulunuzdaki öğrenciler için yararlı olabilecek, interneti güvenli ve sorumlu bir şekilde kullanmaya yönelik on tavsiyeden oluşan kısa bir rehber hazırlayınız.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Siber güvenlik, bilişim alanında en hızlı gelişen alanlardan biridir. Çok sayıda devlet kurumu, şirket, banka, hastane ve eğitim kurumu, verilerini ve hizmetlerini korumak için her gün modern güvenlik sistemleri kullanmaktadır. Bu alanla ilgili farklı meslekler bulunmaktadır. Bunlar arasında siber güvenlik analisti, ağ güvenliği uzmanı, dijital tehdit araştırmacısı ve dijital adli bilişim uzmanı yer almaktadır. Modern teknolojilerin gelişmesiyle birlikte, dijital alanın korunmasına katkıda bulunacak uzmanlara duyulan ihtiyaç sürekli artmaktadır.

Doğru cevabı işaretleyiniz.

1. Siber güvenlik şu alanla ilgilidir:

a) Bilgisayarların yapımı,

b) Dijital sistemlerin ve verilerin korunması,

c) Cep telefonlarının onarılması,

ç) Bilgisayar oyunlarının yapımı.

2. Kavramları açıklayınız.

Kavram	Açıklama
Kişisel veriler	_____
Dijital mahremiyet	_____
Siber güvenlik	_____

3. Tamamlayınız.

Güvenli parola _____ olmalıdır.

4. Güçlü ve güvenli bir parolanın içerdiği üç temel özellik nelerdir?

5. "Kimlik avı" (phishing) saldırısının nasıl işlediğini kendi sözlerinizle açıklayınız ve temel amacının ne olduğunu belirtiniz.

6. Tanımadığınız bir göndericiden, ücretsiz bir video oyunu olduğu belirtilen .exe uzantılı bir dosyanın ekli olduğu bir e-posta alırsanız, korunmak için hangi somut adımları atarsınız?

7. Bir kafede halka açık Wi-Fi ağı kullanmak ile kendi mobil internetinizi (4G/5G) kullanmayı karşılaştırınız. Ağ seçimi kişisel verilerinizin güvenliğini nasıl etkiler?

8. Sosyal ağlarda kimlik hırsızlığının olası sonuçlarını değerlendiriniz. Sizce bir genç için sonuçlar finansal açıdan mı, yoksa duygusal/sosyal açıdan mı daha tehlikelidir? Görüşünüzü gerekçelendiriniz.

9. Okulunuzdaki öğrencilerin verilerini korumak amacıyla bir "Güvenli Çevrim İçi Davranış Kuralları" içermesi gereken üç temel kural öneriniz.

1.3

KRİPTOGRAFİ

Her gün çok sayıda mesaj gönderiyor, internet üzerinden alışveriş yapıyor ve farklı platformlara giriş yapıyoruz. Peki, bu verilerin çalınmadan veya okunmadan dünya genelinde nasıl "seyahat ettiğini" hiç düşündünüz mü? Bunun cevabı, geçmişi çok eskiye dayanan, ancak günümüzde son derece modern bir bilim dalı olan kriptografide yatmaktadır.

Kriptografi, dijital dünyada bilgilerin yetkisiz erişime karşı korunmasını sağlayan güvenli iletişim bilimidir. Temelinde, normal ve okunabilir metnin matematiksel algoritmalar kullanılarak anlaşılmaz bir koda dönüştürüldüğü şifreleme işlemi bulunur. Mesajın alıcı tarafından yeniden anlaşılabilir hâle gelmesi için bunun tersi olan şifre çözme işlemi kullanılır. Anahtarların kullanılma biçimine bağlı olarak kriptografi, simetrik kriptografi ve asimetrik kriptografi olarak ikiye ayrılır. Simetrik kriptografide, verilerin şifrelenmesi ve şifresinin çözülmesi için aynı gizli anahtar kullanılır. Buna karşılık, asimetrik kriptografide iki farklı anahtardan oluşan bir anahtar çifti kullanılır: herkesin erişebildiği açık anahtar şifreleme için, yalnızca alıcının sahip olduğu özel anahtar ise şifre çözme için kullanılır.

BU KONUYU TAMAMLADIĞINIZDA...

- Kişisel verilerin neden korunması gerektiğini açıklarsınız;
- İnternetin güvenli ve güvensiz kullanımını ayırt edersiniz;
- Dijital mahremiyetin ne olduğunu açıklarsınız;
- Dijital teknolojilerin güvenli kullanımına ilişkin örnekler verirsiniz;
- Bilgi alışverişi sırasında ortaya çıkabilecek olası riskleri analiz edersiniz.

DÜŞÜNME SORULARI

1. Bir arkadaşınıza, başkalarının okuyamayacağı gizli bir mesajı nasıl gönderirdiniz?
2. Geçmişte insanların bilgilerini korumaya ihtiyaçları var mıydı?
3. Günümüzde hangi bilgilerin gizli kalmasını isterdiniz?
4. Sizce bankalar ve internet mağazaları kullanıcılarının verilerini nasıl koruyor?
5. Gönderilen mesajı yalnızca yetkili kişinin okuyabilmesi mümkün müdür?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Kriptografi kavramını, şifreleme, anahtar ve gizli mesaj gibi temel kavramlarla tanımlarsınız;
- Kriptografinin günlük hayattaki belirli bir durumda kullanımını fark edersiniz;
- Verilerin korunmasında kriptografinin nasıl kullanıldığını açıklarsınız;
- Korunan ve korunmayan bilgileri ayırt edersiniz;
- Kriptografinin mahremiyet ve güvenlik açısından önemini açıklarsınız.

İletişim ve bilgi alışverişi, insanların günlük yaşamının her zaman önemli bir parçası olmuştur. Tarihin farklı dönemlerinde, gizli kalması ve yalnızca gönderildiği kişiler tarafından anlaşılması gereken mesajların gönderilmesine ihtiyaç duyulmuştur. Askerî planlar, diplomatik anlaşmalar ve önemli devlet bilgileri, içeriklerinin gizliliğini korumak amacıyla çoğu zaman çeşitli yöntemlerle korunmuştur. Bilgilerin korunmasına duyulan ihtiyaç, mesajların şifrelenmesi için farklı tekniklerin geliştirilmesine yol açmıştır.

Modern teknolojilerin gelişmesiyle birlikte kriptografi günlük yaşamın önemli bir parçası hâline gelmiştir. Günümüzde kriptografi yalnızca askerî ve devlet sistemlerinde değil, aynı zamanda bankacılık hizmetlerinin, elektronik postanın, internet mağazalarının ve mobil uygulamaların kullanımında da kullanılmaktadır. Kriptografik yöntemler sayesinde hassas bilgiler güvenli bir şekilde paylaşılabilir ve yetkisiz erişime karşı korunabilir.

1.3.1 KRİPTOGRAFİ NEYİ İFADE EDER?

Kriptografi, bilgilerin yetkisiz erişime karşı korunmasıyla ilgilenen bir bilim dalıdır. Kriptografinin temel amacı, belirli bir mesajın veya verinin yalnızca gönderildiği kişiler tarafından okunabilmesini sağlamaktır. Bu amaçla, anlaşılır mesajın başkaları tarafından kolayca anlaşılamayacağı bir biçime dönüştürülmesini sağlayan çeşitli yöntemler kullanılır. Mesajı gönderen ve alan kişinin, içeriği doğru şekilde okuyabilmek için gerekli bilgilere sahip olması gerekir. Bu şekilde, gizli bilgilerin kötüye kullanılma ihtimali azaltılır.

Kriptografi günümüzde günlük yaşamın birçok alanında kullanılmaktadır. Elektronik bankacılık, internet mağazaları, mobil uygulamalar ve çeşitli dijital hizmetler kullanılırken kişisel veriler ve finansal bilgiler kriptografik yöntemlerle korunur. Ayrıca elektronik posta, dijital sertifikalar ve çok sayıda internet hizmeti, bilgilerin korunması için çeşitli tekniklerden yararlanır. Kriptografi sayesinde kullanıcılar verileri daha güvenli bir şekilde paylaşabilir ve modern dijital hizmetleri kullanabilirler. Kriptografiyi anlayabilmek için dört temel kavramı bilmemiz gerekir: şifreleme, şifre çözme, kriptografik anahtar ve gizli mesaj.

1.3.2 ŞİFRELEME, ŞİFRE ÇÖZME VE KRİPTOGRAFİK ANAHTAR

Şifreleme, anlaşılır bir mesajın yetkisiz erişime karşı korunması amacıyla anlaşılmaz bir biçime dönüştürülmesi işlemidir. Okunabilen mesaja açık mesaj, şifreleme işlemi sonucunda elde edilen mesaja ise şifreli mesaj denir. Bu mesaj, rastgele harf ve işaretlerden oluşan bir dizi gibi görünür. Birisi mesajı internet üzerinden aktarılırken ele geçirse bile okuyamadığı için onun açısından tamamen kullanışsızdır. Örneğin, "Merhaba!" kelimesi "Xy9#z!" gibi anlaşılmaz bir koda dönüştürülebilir. Bu işlem, karmaşık matematiksel kurallar (algoritmalar) kullanılarak gerçekleştirilir.

Mesajı alan kişinin, mesajı yeniden ilk hâline dönüştürebilmesi için gerekli bilgilere sahip olması gerekir. Mesajın anlaşılır hâline geri dönüştürülmesi işlemine şifre çözme denir.

Başarılı bir şifreleme ve şifre çözme işlemi için kriptografik anahtar kullanılır. Anahtar, mesajın doğru şekilde korunmasını ve yeniden okunabilmesini sağlayan kurallar veya veriler bütünüdür. Basit sistemlerde aynı anahtar hem şifreleme hem de şifre çözme için kullanılabilirken, daha modern sistemlerde bu iki işlem için farklı anahtarlar kullanılır. Uygun anahtar bilinmeden, şifreli mesajın yetkisiz kişiler tarafından anlaşılması çok daha zordur.

1.3.3 MODERN TEKNOLOİNİN UYGULANMASI

Sezar Şifresi

Şifrelemenin en bilinen örneklerinden biri Sezar şifresidir. Bu yöntemde mesajdaki her harf, alfabede birkaç sıra sonra bulunan başka bir harfle değiştirilir. Harfler üç sıra kaydırıldığında A harfi Ç, B harfi D, C harfi E ile değiştirilir ve bu şekilde devam eder. Kaydırma kuralını bilen kişi mesajı kolayca okuyabilirken, bu kuralı bilmeyen kişiler için mesaj rastgele bir harf dizisi gibi görünür.

Sezar şifresinin kuralı basit olduğu ve kolayca çözülebildiği için günümüzde önemli bilgilerin korunmasında pratik bir kullanım alanı yoktur. Bununla birlikte, kriptografinin temel fikirlerini basit bir şekilde açıklaması nedeniyle büyük bir eğitimsel öneme sahiptir. Bu yöntem sayesinde şifrelemenin, şifre çözmenin ve kriptografik anahtar kullanımının nasıl işlediği kolayca anlaşılabilir.

Örnek

Harflerin üç sıra kaydırıldığını düşünelim.

Kaynak mesaj	Şifreli mesaj
A	Ç
B	D
C	E
Ç	F

Mesaj İNTERNET ise, **şifreleme** işleminden sonra seçilen kaydırma kuralına göre yeni bir harf dizisi elde edilir.

Şifreleme işlemi (harf harf):
İ (+3 sıra aşağıda) = K
N (+3 sıra aşağıda) = P
T (+3 sıra aşağıda) = V
E (+3 sıra aşağıda) = Ğ
R (+3 sıra aşağıda) = T
N (+3 sıra aşağıda) = P
E (+3 sıra aşağıda) = Ğ
T (+3 sıra aşağıda) = V
Gizli mesaj (anahtar 3) şu harf dizisi olacaktır: KPVĞTPĞV

Modern Kriptografi

Bilgisayar sistemlerinin ve internetin gelişmesiyle birlikte kriptografi, geçmişe kıyasla çok daha geniş bir kullanım alanı kazanmıştır. Günümüzde çok sayıda dijital hizmet, kullanıcılar ile bilişim sistemleri arasında alışverişi yapılan bilgileri korumak için modern kriptografik yöntemler kullanmaktadır. Kullanıcı hesabına giriş yaparken, elektronik posta gönderirken, internet mağazalarını veya elektronik bankacılığı kullanırken hassas bilgiler çeşitli kriptografik yöntemlerle korunmaktadır. Bu sayede verilerin yetkisiz kişiler tarafından okunması veya değiştirilmesi ihtimali azaltılmaktadır.

Modern kriptografide bilgilerin korunması için farklı yöntemler uygulanmaktadır. Bazı sistemlerde mesajların şifrelenmesi ve şifrelerinin çözülmesi için aynı kriptografik anahtar kullanılırken, bazı sistemlerde iki farklı anahtar kullanılır. Yöntemin seçimi, gerekli güvenlik düzeyine ve uygulanacağı alana bağlıdır. Modern kriptografik algoritmalar, yüksek düzeyde koruma sağlamak ve kullanıcılar ile bilişim sistemleri arasında güvenli iletişime olanak tanımak amacıyla geliştirilmiştir.

Kriptografinin Günlük Yaşamda Kullanımı

Kriptografi, insanların günlük olarak kullandığı birçok dijital hizmetin bir parçasıdır. Elektronik bankacılık kullanırken, internet üzerinden ödeme yaparken veya çeşitli internet sitelerine erişirken kişisel ve finansal veriler kriptografik mekanizmalarla korunur. Mobil iletişim uygulamaları, mesajların yalnızca görüşmeye katılan kişiler tarafından okunabilmesini sağlamak için kriptografiden yararlanır. WhatsApp, Viber ve Signal uygulamaları uçtan uca (end-to-end) şifreleme olarak adlandırılan kriptografiyi kullanır. Bu, mesajın telefonunuzda şifrelendiği ve ancak arkadaşınıza ulaştığında şifresinin çözüldüğü anlamına gelir. Şirketler (örneğin Meta) bile görüşmelerinizi okuyamaz.

Ayrıca birçok elektronik hizmet, kullanıcıların ve internet sitelerinin kimliğinin doğrulanmasını sağlayan dijital sertifikalar kullanır. Kriptografi yalnızca gizli servisler tarafından kullanılan bir uygulama değildir. Kriptografi hayatımızın her alanında kullanılmaktadır.

Kriptografinin kullanılması, modern bilişim sistemlerinde güvenliğin ve güvenin artırılmasına katkı sağlar. Kriptografik yöntemler sayesinde kullanıcılar çeşitli dijital hizmetleri daha güvenli bir şekilde kullanabilirler. Bununla birlikte, kriptografinin genel dijital güvenliğin yalnızca bir parçası olduğunu anlamak önemlidir. Daha yüksek düzeyde koruma sağlamak için güvenli parolalar kullanmak, bilgileri dikkatli bir şekilde paylaşmak ve bilişim sistemlerini düzenli olarak güncellemek gibi diğer önlemlerin de uygulanması gerekir.

Bir öğrenci, bir gün içinde farkında olmadan birçok kez kriptografi kullanabilir. Elektronik günlüğe giriş yaparken, bir mobil uygulama üzerinden mesaj gönderirken, internetten ürün satın alırken veya elektronik bankacılık kullanırken veriler modern kriptografik yöntemlerle korunur. Bu sayede kişisel bilgilerin kötüye kullanılma ihtimali azaltılır.

Bir kullanıcı internet mağazasına erişip ödeme bilgilerini girdiğinde, kriptografik sistemler bu bilgilerin kullanıcı ile sunucu arasında güvenli bir şekilde iletilmesini sağlar. Bu sayede hassas veriler düz metin olarak gönderilmez, bunun yerine yetkisiz kişiler tarafından okunması çok daha zor olan korumalı bir biçimde iletilir.

Korumalı ve korumasız bilgiler nasıl ayırt edilir?

Bir öğrenci olarak, kullandığın bilgilerin güvenli olup olmadığını hemen ayırt edebilmen çok önemlidir:

- **Korumalı bilgiler:** Bunlar şifrelenmiş ve yalnızca yetkili kişilerin erişebildiği verilerdir. Bunları tarayıcıdaki kilit simgesinden ve HTTPS protokolünden (S harfi Secure / Güvenli anlamına gelir) kolayca anlayabilirsin. Bu tür sitelerde parola veya kişisel bilgi girmek daha güvenli bir bağlantı üzerinden gerçekleştirilir.
- **Korumasız bilgiler:** Bunlar HTTP protokolü üzerinden (kilit simgesi olmadan) düz metin olarak iletilen herkese açık bilgilerdir. Ayrıca sosyal medyadaki herkese açık profillerde, açık forumlarda veya şifresiz bir kafedeki halka açık Wi-Fi ağı kullanılırken paylaşılan bilgiler de korumasız kabul edilir. Bir miktar teknik bilgiye sahip kötü niyetli kişiler bu bilgileri ele geçirebilir.

Kriptografi olmadan modern dijital dünyanın işleyişini sürdürmesi mümkün olmazdı. Kriptografinin önemi iki temel nedenden kaynaklanır:

- **Mahremiyetin korunması:** Bize dijital dünyada kişisel bir alana sahip olma imkânı sağlar. Arkadaşlarımızla, ailemizle veya doktorlarla iletişim kurmamıza, özel düşüncelerimizin, fotoğraflarımızın ve sırlarımızın yalnızca bize ait olduğunu bilmemize olanak tanır.
- **Dijital güvenliğin sağlanması:** Devlet kurumlarını, hastaneleri, okulları ve bankaları siber saldırılardan korur. Kriptografi, kötü niyetli kişilerin başkalarının e-günlükteki notlarını değiştirmesini, kimliklerini çalmasını veya bütün bir devletin sistemlerini devre dışı bırakmasını önler.

SONUÇ:

Kriptografi, dijital yaşamlarımızın görünmez koruyucusudur. Temelini anlamak, kendi dijital izimizi nasıl koruyacağını bilen daha bilinçli kullanıcılar olmamıza yardımcı olur.

BİLİYOR MUSUNUZ?

Biliyor musun, internet sitesinde adresin yanında bir kilit simgesi gördüğünde, cihazın ile site arasındaki iletişim modern kriptografik teknolojilerle korunur?

ARAŞTIR

İnternette farkında olmadan kriptografi kullandığınız günlük etkinlikleri araştırınız. En az beş örnekten oluşan bir liste hazırlayınız ve kriptografinin her birindeki rolünü açıklayınız.

BİLGİLERİ TEKRARLAMA SORULARI

1. Tekrar Soruları
2. Kriptografi neyi ifade eder?
3. Şifreleme ile şifre çözme arasındaki fark nedir?
4. Kriptografik anahtarın rolü nedir?
5. Sezar şifresinin günümüzde eğitim açısından önemi neden vardır?
6. Kriptografi hangi günlük dijital hizmetlerde uygulanmaktadır?
7. İnsanların bilgilerini korumaya neden ihtiyaçları vardır?
8. Şifreleme ve şifre çözme arasındaki fark nedir?
9. Kriptografik anahtarın bilgilerin korunmasındaki rolü nedir?
10. Sezar şifresi günümüzde neden daha çok eğitim açısından önem taşımaktadır?
11. Kriptografi günümüzde hangi dijital hizmetlerde günlük olarak uygulanmaktadır?

ARAŞTIRMA VE UYGULAMA ÇALIŞMALARI

1. Sezar şifresinin kuralını kullanarak sınıf arkadaşın için gizli bir mesaj oluştur. Daha sonra mesajları değiş tokuş edin ve şifrelerini çözmeye çalışın.
2. Farkında olmadan kriptografi kullandığın günlük etkinlikleri araştır. En az beş örnek belirt.
3. Sezar şifresini bilgilerin korunmasında kullanılan modern yöntemlerle karşılaştır. En az üç farklılık belirt.
4. Bankaların, internet mağazalarının ve mobil uygulamaların neden kriptografi kullandığını düşün. Kendi görüşlerini içeren kısa bir metin hazırla.
5. Dijital sertifikanın ne olduğunu araştır ve internet üzerinden güvenli iletişim için neden önemli olduğunu açıkla.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Kriptografi, bilgilerin korunması bilimidir.

Şifreleme, anlaşılır mesajı korunan bir biçime dönüştürür.

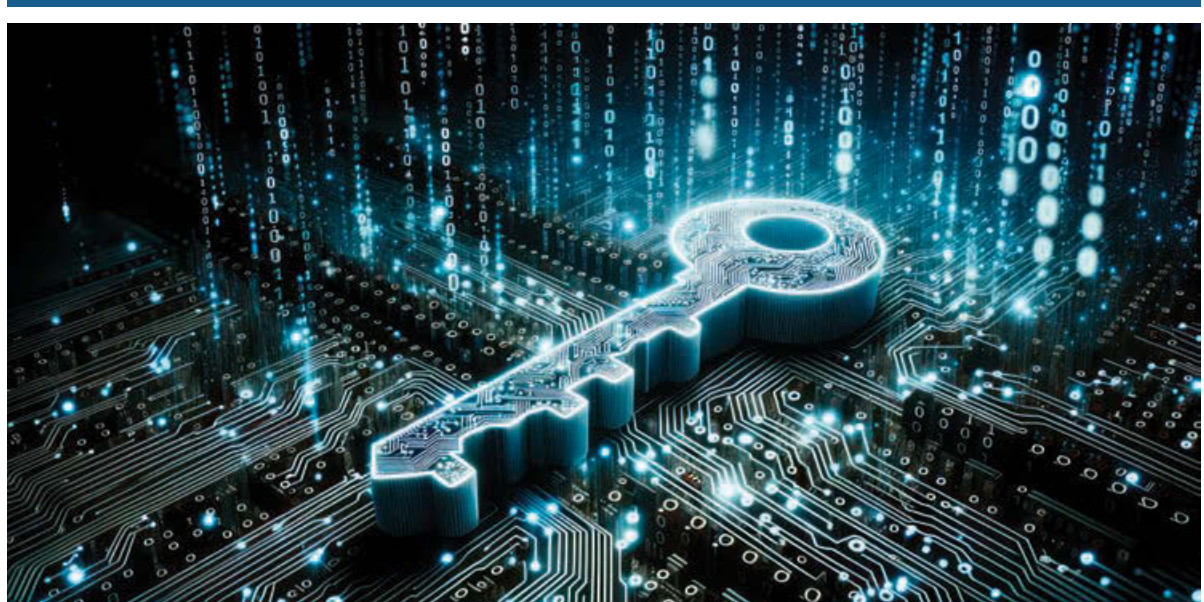
Şifre çözme, mesajın yeniden okunmasını sağlar.

Kriptografik anahtar, bilgilerin korunmasında önemli bir role sahiptir.

Kriptografi, modern bilişim sistemlerinde ve günlük dijital hizmetlerde geniş bir kullanım alanına sahiptir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern kriptografi, bilişimin en önemli alanlarından biridir ve internetin işleyişinde önemli bir role sahiptir. Mesajların ve finansal verilerin korunmasının yanı sıra, kriptografik yöntemler belgelerin dijital olarak imzalanmasında, dijital sertifikaların oluşturulmasında, bilgisayar ağlarının korunmasında ve blockchain teknolojisinin işleyişinde uygulanmaktadır. Modern bilişim sistemlerinin gelişmesiyle birlikte kriptografi, bilgilerin daha yüksek düzeyde güvenliğini ve gizliliğini sağlamak amacıyla sürekli olarak geliştirilmektedir.



BİLGİNİZİ KONTROL EDİN

1. Doğru cevabı işaretleyiniz.

Kriptografi şu amaçla kullanılır:

a) Bilgisayarları hızlandırmak;

b) Bilgileri korumak;

c) Bilgisayar ağlarını onarmak;

ç) Mobil uygulamalar geliştirmek.

2. Kriptografi kavramını tanımlayınız. Tanımınızda mutlaka şu üç temel kavramı kullanınız: şifreleme, anahtar ve gizli mesaj

3. Doğru veya yanlış.

Şifreleme ve şifre çözme aynı işlemdir.

4. Kavramları açıklayınız.

Kavram	Açıklama
Şifreleme	_____
Şifre çözme	_____
Kriptografik anahtar	_____

5. Boşluğu doldurunuz.

En bilinen basit şifreleme yöntemlerinden biri _____.

6. Kriptografi, internet üzerinden aktarılırken verilerin korunmasına pratik olarak nasıl yardımcı olur? Normal metnin şifreli metne ve tekrar normal metne dönüştürülmesi sürecini açıklayınız.

7. Günlük yaşamınızdan (örneğin iletişim uygulamalarını kullanırken veya internetten alışveriş yaparken) farkında olmadan kriptografi kullandığınız belirli bir durumu belirtiniz ve o anda verilerinizin başına ne geldiğini açıklayınız.

8. İnternette korunan ve korunmayan bilgileri karşılaştırınız. Öğrenciler, bir internet sitesinin verilerini koruyup korumadığını web tarayıcısındaki hangi dijital işaretlerden (semboller veya protokoller) hemen anlayabilir?

9. Kriptografinin toplum açısından önemini değerlendiriniz: Kriptografi dijital dünyadan aniden ortadan kalksaydı vatandaşların gizliliğine ve kurumların (bankalar ve hastaneler gibi) güvenliğine ne olurdu? Sonuçlarını açıklayınız.

10. Daha küçük yaştaki öğrencilere verilerini korumanın neden önemli olduğunu açıklamanız gerektiğini düşününüz. İnternette korunan ve korunmayan bilgileri ayırt etmeye dayanan, güvenli internet kullanımı için üç basit "altın kural" oluşturunuz.

PROBLEM ÇALIŞMASI

Bir arkadaşına internet üzerinden gizli bir mesaj göndermen gerektiğini düşün. Mesajın şifrelenmesinin neden önemli olduğunu ve kriptografinin sağladığı faydaların neler olduğunu açıkla.

1.4

TEKNOLOJISI – MODERN DÜNYADA GÜVENİLİR DİJİTAL KAYITLAR

Blockchain teknolojisi, dijital kayıtların güvenli bir şekilde saklanmasını ve takip edilmesini sağlayan modern bir bilişim çözümüdür. Kriptografik yöntemler ve verilerin düzenlenme biçimi sayesinde yapılan her değişiklik kaydedilir ve kolayca kontrol edilebilir. Bu özellikleri nedeniyle blockchain teknolojisi modern toplumun farklı alanlarında kullanılmakta ve kullanım alanı sürekli genişlemektedir.

BU KONUYU TAMAMLADIĞINIZDA...

- Kriptografinin ne olduğunu açıklarsınız;
- Şifreleme ve şifre çözme arasındaki farkı ayırt edersiniz;
- Kriptografik anahtarın rolünü açıklarsınız;
- Bilgilerin korunmasına ilişkin örnekler verirsiniz;
- Modern bilişim sistemlerinde kriptografinin kullanımını fark edeceksiniz.

DÜŞÜNME SORULARI

1. Önemli bilgiler, yapılan her değişikliğin kaydedileceği şekilde nasıl saklanabilir?
2. Bir veritabanının tek bir merkezi yönetici olmadan çok sayıda kullanıcı tarafından kullanılması mümkün müdür?
3. Bir kayıtta belirli bir değişikliği kimin ve ne zaman yaptığının bilinmesi neden önemlidir?
4. Kripto paralar için kullanılan bir teknoloji başka alanlarda da kullanılabilir mi?
5. Modern bilişim teknolojileri, dijital hizmetlere duyulan güvenin artmasına nasıl katkıda bulunabilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Blockchain teknolojisinin ne olduğunu açıklarsınız;
- Bloğun yapısını ve blokların birbirine bağlanmasını açıklarsınız;
- Merkeziyetsizlik ilkesini açıklarsınız;
- Blockchain teknolojisinin kullanımına örnekler verirsiniz.

Günümüz toplumunda her gün güvenli bir şekilde saklanması ve yetkisiz değişikliklere karşı korunması gereken çok sayıda dijital kayıt oluşturulmaktadır. Bankacılık işlemleri, mülkiyet bilgileri, sağlık kayıtları, sözleşmeler ve çeşitli elektronik belgeler doğru ve güvenilir olması gereken bilgilerdir. Bu kayıtlar kolayca değiştirilebilir veya silinebilirse ciddi sorunlar ve anlaşmazlıklar ortaya çıkabilir.

1.4.1 BLOCKCHAIN TEKNOLOJİSİ NEDİR?

Modern toplumda önemli bilgilerin güvenli bir şekilde saklanmasına ve gerektiğinde doğrulanabilmesine ihtiyaç duyulmaktadır. Gayrimenkul satın alımı, para transferi, diploma düzenlenmesi veya sözleşme yapılması gibi işlemlerde gerçekleştirilen faaliyetlere ilişkin doğru kayıtların bulunması gerekmektedir. Kayıtların kolayca değiştirilebilmesi veya silinebilmesi durumunda büyük sorunlar ve yanlış anlaşılmalar ortaya çıkabilir. Bu nedenle modern bilişim teknolojileri, dijital kayıtların güvenli bir şekilde saklanmasını ve izlenmesini sağlayan çözümler sunmaktadır.

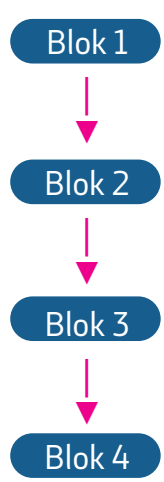
Blockchain teknolojisi, verilerin düzenlenmesi ve saklanmasına yönelik özel bir yöntemdir. Bilgiler tek bir merkezi kayıt üzerinde saklanmak yerine, birbiriyle bağlantılı birden fazla blok hâlinde düzenlenir. Her yeni blok yeni bilgiler içerir ve zincir içerisinde önceki blokla bağlantılıdır. Bu düzenleme ve kriptografik yöntemlerin kullanılması sayesinde veriler korunur ve gerçekleştirilen her değişiklik kayıt altına alınır. Bu nedenle blockchain teknolojisi, çeşitli türlerdeki dijital kayıtların saklanması için güvenilir bir yöntem sunmaktadır.

Bloklar, İşlemler ve Merkeziyetsizlik

Blockchain teknolojisinin temel unsuru bloktur. Blok, belirli bilgilerin saklandığı dijital bir bütündür. Sistemin kullanım amacına bağlı olarak blok finansal işlemlere, mülkiyet değişikliklerine, yapılan sözleşmelere veya diğer önemli kayıtlara ilişkin verileri içerebilir. Yeni bir kayıt oluşturulduğunda, önceki blokla bağlantı kuran yeni bir blok oluşturulur ve bu blok zincirin bir parçası hâline gelir. Bu şekilde sistemde gerçekleşen tüm olayların ardışık bir geçmişi oluşturulur.

Blockchain teknolojisi çoğunlukla merkeziyetsizlik kavramıyla ilişkilendirilmektedir. Geleneksel sistemlerde genellikle verileri saklayan ve kontrol eden tek bir merkezi yönetici bulunmaktadır. Blockchain sistemlerinde ise kayıtlar birbirine bağlı birden fazla katılımcının erişimine açık olabilir ve gerçekleştirilen her yeni değişiklik kayıt altına alınarak doğrulanabilir. Bu tür bir organizasyon, kayıtların geçmişinin korunması ve kolaylıkla izlenebilmesi sayesinde sistemde daha fazla şeffaflık ve güven oluşmasına katkı sağlamaktadır.

Blokların Birbirine Bağlanmasının Gösterimi



Her yeni blok, önceki blokla bağlantı kurarak ortak kayıt zincirinin bir parçası hâline gelir.

Bloklar Nerede Saklanır?

Blockchain teknolojisinin en önemli özelliklerinden biri, verilerin saklanma biçimidir. Tüm kayıtların tek bir merkezi bilgisayarda veya sunucuda tutulması yerine, blockchain sisteminin kopyaları, sistemin işleyişine katılan birden fazla bağlı bilgisayarda saklanabilir. Bu bilgisayarlara düğüm (İng. nodes) adı verilir ve her birinde blok zincirinin bir kopyası bulunur. Sisteme yeni bir blok eklendiğinde, bu bilgi kaydedilir ve ağdaki tüm katılımcıların sistemlerinde güncellenir. Bu organizasyon biçimi sayesinde veriler tek bir depolama konumuna bağlı kalmaz ve güvenlikleri ile erişilebilirlikleri önemli ölçüde artırılır.

Blokların birden fazla bağlı bilgisayarda saklanması, sistemin güvenilirliğine ve dayanıklılığına katkıda bulunur. Bilgisayarlardan birinin çalışmayı durdurması veya teknik bir sorun ortaya çıkması durumunda, diğer bilgisayarlar saklanan verilere erişmeye devam eder. Aynı zamanda her yeni değişikliğin sistemde kayıt altına alınması gerekir, bu da kayıtların doğruluğunun daha kolay denetlenmesini sağlar. Bu şekilde blockchain teknolojisi, bilgilerin güvenli bir şekilde saklanmasını sağlamakta ve ağı kullanan katılımcılar arasında güven oluşturmaktadır.

1.4.2 BLOCKCHAIN TEKNOLOJİSİNİN MODERN DÜNYADAKİ YERİ

Blockchain teknolojisinin en bilinen kullanım alanlarından biri kripto paralardır. Kripto paralar, işlemlerin güvenli bir şekilde kaydedilmesi için blockchain teknolojisini kullanan dijital değişim araçlarıdır. Her işlem bir bloğa kaydedilir ve kayıt zincirinin bir parçası hâline gelir. Bu sayede işlemlerin geçmişi izlenebilir ve yetkisiz değişikliklerin yapılma olasılığı azaltılabilir. Bitcoin, en bilinen kripto paralardan biridir, ancak günümüzde bu teknolojiyi kullanan çok sayıda farklı sistem de bulunmaktadır. Blockchain teknolojisi yalnızca kripto paralar için kullanılmamaktadır. Sağlık, lojistik, eğitim, kamu yönetimi ve çeşitli dijital kayıtların yönetimi gibi alanlarda da uygulama alanı bulmaktadır.

Blockchain Teknolojisinin Avantajları ve Olanakları

Blockchain teknolojisi, dijital kayıtların saklanması ve işlenmesi sırasında çeşitli avantajlar sağlamaktadır. Blokların birbirleriyle bağlantılı olması sayesinde gerçekleştirilen her değişiklik kayıt altına alınmakta ve doğrulanabilmektedir. Bu durum, sistemde daha fazla şeffaflık ve güven oluşmasına katkıda bulunmaktadır. Ayrıca kriptografik yöntemlerin kullanılması, bilgilerin daha iyi korunmasını sağlamakta ve veriler üzerinde yetkisiz değişiklik yapılma olasılığını azaltmaktadır.

Çok sayıdaki avantajının yanı sıra blockchain teknolojisinin uygulanması bazı zorlukları da beraberinde getirmektedir. Teknolojinin başarılı bir şekilde kullanılabilmesi için uygun teknik altyapı ve uygulandığı sistemlerin iyi bir şekilde organize edilmesi gerekmektedir. Ayrıca farklı uygulama alanlarının farklı ihtiyaç ve gereksinimleri bulunmaktadır. Bu nedenle teknolojinin seçimi, kullanım amacına bağlıdır. Bilişim teknolojilerinin gelişmesi ve dijital dönüşümün ilerlemesiyle birlikte blockchain teknolojisinin giderek daha fazla modern bilişim sistemi ve hizmetinde uygulanması beklenmektedir.

1.4.3 MODERN TEKNOLOJİNİN UYGULAMADAKİ KULLANIMI

Blockchain Teknolojisi ve Kadastro Kayıtları

Birçok ülkede kadastro hizmetleri, arazilerin ve taşınmazların mülkiyetine ilişkin kayıtları tutmaktadır. Bir taşınmazın satın alınması, satılması, miras yoluyla devredilmesi veya bağışlanması sırasında her değişikliğin doğru bir şekilde kayıt altına alınması gerekmektedir. Kayıtların güncel olmaması veya kolayca değiştirilebilmesi durumunda mülkiyet konusunda anlaşmazlıklar ortaya çıkabilir ve ciddi hukuki sorunlar meydana gelebilir. Bu nedenle farklı ülkeler, kadastro verilerinin güvenliğini ve şeffaflığını artırmayı sağlayacak çağdaş bilişim çözümlerini araştırmaktadır.

Afrika'da kadastro kayıtlarının ve arazi kayıtlarının yönetiminde blockchain teknolojisinin uygulanmasına yönelik girişimler bulunmaktadır. Bu tür projelerin geliştirildiği ülkelerden biri Gana'dır. Burada çağdaş bilişim çözümleri, mülkiyet kayıtlarının iyileştirilmesi için bir olanak olarak değerlendirilmektedir. Blockchain teknolojisi sayesinde bir taşınmaz üzerindeki tüm değişikliklerin geçmişi saklanabilir, böylece her satış, miras yoluyla devir veya diğer mülkiyet devri türleri kalıcı olarak kayıt altına alınmış olur. Bu yaklaşım, verilere duyulan güvenin artırılmasına ve kötüye kullanım olasılığının azaltılmasına katkıda bulunmaktadır.

BİLİYOR MUSUNUZ?

Blockchain teknolojisinin günümüzde yalnızca kripto paralar için kullanılmadığını biliyor musun? Finans sektörünün yanı sıra sağlık, eğitim, ulaşım, lojistik alanlarında ve dijital belgelerin düzenlenmesinde de uygulanmakta veya test edilmektedir.

ARAŞTIR

Blockchain teknolojisinin uygulandığı üç alanı araştırınız. Her bir alan için aşağıdakileri belirtiniz:

- Hangi bilgilerin kayıt altına alındığını;
- Blockchain teknolojisinin hangi faydaları sağladığını;
- Teknolojinin veri güvenliğinin artırılmasına ne şekilde katkıda bulunduğunu.

TEKRARLAMA SORULARI

1. Blockchain teknolojisi nedir?
2. Blockchain sisteminde blok nedir?
3. İşlem nedir?
4. Merkeziyetsizlik ne anlama gelir?
5. Kripto paralar neden blockchain teknolojisinin en bilinen uygulamalarından biridir?
6. Blockchain teknolojisinin uygulanabileceği en az üç alan belirtiniz.

İŞ BİRLİĞİNE DAYALI VE UYGULAMALI ETKİNLİKLER

1. Okulunuzda elektronik diplomaların saklanması gerektiğini düşününüz. Blockchain teknolojisinin bu diplomaların kayıt altına alınmasına nasıl yardımcı olabileceğini açıklayınız.
2. Seçtiğiniz bir kripto parayı araştırınız ve kullanım alanı ile temel özellikleri hakkında kısa bir rapor hazırlayınız.
3. Klasik bir veri tabanı ile blockchain sistemini bilgilerin saklanma biçimi açısından karşılaştıracağınız bir tablo hazırlayınız.
4. Kuzey Makedonya Cumhuriyeti'nde blockchain teknolojisinin hangi alanlarda uygulanabileceğini düşününüz. En az üç örnek belirtiniz ve seçiminizi gerekçelendiriniz.
5. Blockchain teknolojisinin kadastro kayıtlarının iyileştirilmesine ve taşınmazların mülkiyetinin yönetilmesine nasıl katkıda bulunabileceğini araştırınız.

SONUÇ:

Blockchain teknolojisi, dijital kayıtların güvenli bir şekilde düzenlenmesini ve saklanmasını sağlayan bir yöntemdir. Blockchain sisteminin temel unsurları bloklar ve bu blokların bir zincir hâlinde birbirine bağlanmasıdır. İşlem, sisteme eklenen bir kaydı ifade eder. Merkeziyetsizlik, kayıtların birden fazla katılımcı tarafından erişilebilir ve doğrulanabilir olmasını sağlar. Kripto paralar, blockchain teknolojisinin en bilinen uygulamalarından biridir. Bu teknoloji, modern toplumun birçok farklı alanında da uygulanmaktadır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Blockchain teknolojisi sürekli gelişmekte ve yeni uygulama alanlarında kullanılmaya başlanmaktadır. Finans sektörünün yanı sıra tedarik zincirlerinde ürünlerin izlenmesi, dijital sertifikaların düzenlenmesi, tıbbi verilerin yönetilmesi ve fikri mülkiyetin korunması amacıyla da kullanılmaktadır. Birçok ülke ve kuruluşta, çeşitli bilişim sistemlerinin güvenliğini, şeffaflığını ve güvenilirliğini artırmak amacıyla bu teknolojinin uygulanmasına yönelik yeni yöntemler araştırılmaktadır.

BİLGİNİZİ KONTROL EDİN

1. Doğru cevabı yuvarlak içine alınız.

Blockchain teknolojisi en çok aşağıdakilerden hangisi için kullanılır:

a) Fotoğrafların işlenmesi;
b) Dijital kayıtların güvenli bir şekilde saklanması ve izlenmesi;
c) İşletim sistemlerinin geliştirilmesi;
ç) Bilgisayar ağlarının onarılması.

2. Doğru mu, yanlış mı?

Kripto paralar, blockchain teknolojisinin tek kullanım alanıdır.

3. Kavramları açıklayınız.

Kavram	Açıklama
Blok	_____
İşlem	_____
Merkeziyetsizlik	_____

4. Tamamlayınız.

Blockchain teknolojisinin en bilinen uygulaması _____.

5. Blockchain ağında bir dijital işlemin gerçekleştirilmesi sırasında merkeziyetsizlik ilkesi, geleneksel bankacılık ödemeleriyle karşılaştırıldığında nasıl uygulanmaktadır?

6. Veri güvenliğinde merkeziyetsizliğin rolünü analiz ediniz. Bir blok zincire bağlandıktan sonra daha önce kaydedilmiş bir işlemin silinmesi veya değiştirilmesi neden neredeyse imkânsızdır?

7. Blockchain teknolojisinin en yaygın uygulaması olan kripto paraların avantajlarını ve dezavantajlarını değerlendiriniz. Sizce gelecekte geleneksel paraların yerini tamamen alabilirler mi? Görüşünüzü gerekçelendiriniz.

8. Kripto paralar dışında, blockchain teknolojisinin değiştirilemezlik ve merkeziyetsizlik özelliklerinden yararlanarak gerçek bir toplumsal sorunun çözümünde nasıl kullanılabileceğine ilişkin bir fikir öneriniz. Seçimlerde oy kullanma veya telif haklarının korunması gibi alanlardan yararlanabilirsiniz.

DURUMSAL ETKİNLİK

Bir yerleşim yerindeki evlerin mülkiyet kayıtlarının tutulması gerektiğini düşününüz. Blockchain teknolojisinin bu verilerin güvenliğini ve güvenilirliğini artırmaya nasıl katkıda bulunabileceğini açıklayınız.

1.5

ROBOTİK VE OTOMASYON

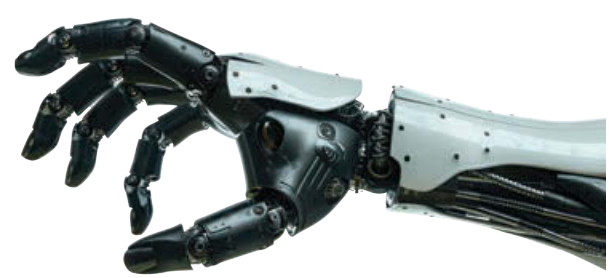
Modern bilişim teknolojileri, çeşitli görevleri kendi başlarına yerine getirebilen makinelerin oluşturulmasını mümkün kılmaktadır. Bilgisayar sistemleri, sensörler ve otomatik kontrol alanındaki gelişmeler sayesinde robotlar günümüzde günlük yaşamın birçok alanında ve sanayide kullanılmaktadır. Robotlar hassas, tekrarlayan veya tehlikeli görevleri yerine getirerek insanlara yardımcı olmakta ve çalışma süreçlerinin iyileştirilmesine katkıda bulunmaktadır.

BU KONUYU TAMAMLADIĞINIZDA...

- Modern teknolojilere örnekler vermeyi;
- Bilişim sistemlerinin bilgileri nasıl işlediğini açıklamayı;
- Dijital teknolojilerin günlük yaşamda kullanım alanlarını tanımayı;
- Farklı süreçlerin otomasyonuna ilişkin örnekler vermeyi;
- Modern teknolojilerin toplumun gelişimine nasıl katkıda bulunduğunu açıklamayı bileceksiniz.

DÜŞÜNME SORULARI

1. Robotlar, daha önce insanlar tarafından gerçekleştirilen görevleri yerine getirebilir mi?
2. Robotların kullanıldığını hangi mesleklerde gördünüz?
3. Bir robot günlük yaşamda insanlara nasıl yardımcı olabilir?
4. Robotlar her zaman insan görünümünde midir?
5. Bir robota hangi görevleri vermek isterdiniz?



AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Robotik kavramını açıklamayı.
- Robotların kullanım alanlarına örnekler vermeyi.
- Otomasyonun rolünü anlamayı.
- Robot kullanımını modern teknolojilerle ilişkilendirmeyi.

Robotik, robotik sistemlerin tasarımı, üretimi ve uygulanmasıyla ilgilenen modern bir bilim ve teknoloji alanıdır. Gelişimi bilişim, elektronik, makine mühendisliği ve otomasyon alanlarıyla ilişkilidir. Günümüzde robotlara fabrikalarda, hastanelerde, tarımda, ulaşımda, evlerde ve daha birçok alanda rastlanabilmektedir. Robotların kullanılması sayesinde birçok görev daha hızlı, daha hassas ve daha güvenli bir şekilde gerçekleştirilmektedir.

1.5.1 ROBOTİK KAVRAMI

Robotik, robotların oluşturulması ve uygulanmasıyla ilgilenen bilimsel ve teknik bir alandır. Robot, bir program ve çevresinden aldığı bilgiler aracılığıyla önceden belirlenmiş görevleri yerine getirebilen bir makinedir. Robotlar çalışmaları sırasında hareket etmelerini, nesneleri tanımalarını ve belirli faaliyetleri gerçekleştirmelerini sağlayan çeşitli sensörler, kontrol sistemleri ve mekanik parçalardan yararlanır. Kullanım amaçlarına bağlı olarak robotların görünüşleri ve sahip oldukları yetenekler farklılık gösterebilir.

Robotların mutlaka insan görünümünde olması gerekmez. Günümüzde kullanılan robotik sistemlerin büyük bir bölümü, gerçekleştirdikleri görevlere uygun biçimde tasarlanmıştır. Sanayide çoğunlukla ürünlerin montajında kullanılan robotik kollar, evlerde temizlik robotları, tarımda ise ekinlerin izlenmesi ve işlenmesi için kullanılan otomatik makineler bulunmaktadır. Farklı robot türleri, insanlara yardımcı olmak ve görevlerin daha hızlı ve daha hassas bir şekilde gerçekleştirilmesini sağlamak amacıyla geliştirilmektedir.

1.5.2 OTOMASYON VE ROBOTLARIN KULLANIMI

Otomasyon, belirli görevlerin insanın doğrudan katılımı en aza indirilerek veya tamamen ortadan kaldırılarak otomatik şekilde gerçekleştirildiği süreçtir. Robotik sistemler, çok sayıda tekrarlayan faaliyeti yüksek hız ve doğrulukla gerçekleştirebildikleri için otomasyonun önemli bir parçasıdır. Üretimde robotlar ürünleri monte etmekte, kaynak, boyama ve paketleme işlemlerini gerçekleştirmekte, depolarda ise malların sınıflandırılması ve taşınmasına yardımcı olmaktadır.

Robotların kullanımı yalnızca sanayiyle sınırlı değildir. Tıpta, doktorların karmaşık ameliyatlar gerçekleştirmesine yardımcı olan robotik sistemler kullanılmaktadır. Tarımda robotlar, ekinlerin durumunu izleyebilmekte ve daha hassas sulama yapılmasına katkıda bulunabilmektedir. Ulaşım alanında otonom araçlar ve dronlar geliştirilmekte, evlerde ise temizlik ve yaşam alanlarının bakımı için robotlardan yararlanılmaktadır. Bilişim alanındaki gelişmeler sayesinde robotlar, modern toplumdaki giderek daha fazla faaliyetin bir parçası hâline gelmektedir.

1.5.3 TIBBİ ROBOTLAR, DRONLAR VE MODERN ROBOTİK SİSTEMLER

Robotik alanındaki gelişmeler, yaşamın ve çalışma hayatının farklı alanlarında insanlara yardımcı olan özel amaçlı robotların geliştirilmesini mümkün kılmaktadır. Tıp alanında, doktorların karmaşık ve hassas cerrahi müdahaleler gerçekleştirmesine olanak sağlayan robotik sistemler kullanılmaktadır. Bu sistemler ayrıca hastaların rehabilitasyonuna, tıbbi ekipmanların taşınmasına ve hastanelerde ilaçların ulaştırılmasına yardımcı olmaktadır. Yüksek hassasiyetleri sayesinde tıbbi robotlar, sağlık hizmetlerinin iyileştirilmesine ve belirli işlemler sırasında hata olasılığının azaltılmasına katkıda bulunmaktadır.

Modern robotik sistemlerin özel bir grubunu dronlar oluşturmaktadır. Bunlar uzaktan kontrol edilebilen veya önceden belirlenmiş görevleri otomatik olarak yerine getirebilen insansız hava araçlarıdır. Dronlar fotoğraf çekme ve görüntü kaydetme, tarım alanlarını gözlemleme, trafiği izleme, arama ve kurtarma çalışmalarının yürütülmesi ve çeşitli ürünlerin teslim edilmesi gibi amaçlarla kullanılmaktadır. Bazı ülkelerde dronlardan tıbbi malzemelerin ulaşılması zor bölgelere taşınmasında da yararlanılmaktadır. Bu tür sistemlerin gelişimi, robotiğin modern toplumda geniş bir kullanım alanına sahip olduğunu göstermektedir.

1.5.4 ROBOTLARIN OTOMOTİV ENDÜSTRİSİNDE KULLANIMI

Modern otomobil üretim fabrikalarında çok sayıda iş süreci, endüstriyel robotlar yardımıyla gerçekleştirilmektedir. Robotik kollar, kaynak yapma, boyama, montaj ve ağır parçaları yüksek hız ve hassasiyetle taşıma işlemlerini gerçekleştirir. Robotlar kesintisiz olarak çalışabilir ve aynı görevleri aynı kalitede tekrar tekrar yerine getirebilir. Bu da üretimin artırılmasına katkı sağlar.

Robotlar işlerin büyük bir bölümünü gerçekleştirse de insanların yerini tamamen almazlar. Robotların programlanması, bakımı ve kontrolü için farklı bilgi ve becerilere sahip uzman kişilere ihtiyaç vardır. Bu şekilde robotik teknolojisi, bilişim, elektronik ve mühendislikle ilişkili modern mesleklerin gelişmesi için yeni fırsatlar yaratmaktadır.

1.5.5 ROBOTİĞİN AVANTAJLARI VE SINIRLILIKLARI

Robotik, modern toplum için çok sayıda fayda sağlamaktadır. Robotlar yüksek hassasiyet, hız veya dayanıklılık gerektiren görevleri yerine getirebilir. Sanayide çeşitli ürünlerin üretimine yardımcı olmakta, tıp alanında sağlık hizmetlerinin iyileştirilmesine katkıda bulunmakta, tarımda ise ekinlerin daha verimli bir şekilde izlenmesini ve işlenmesini sağlamaktadır. Robotlar madenler, doğal afetlerden etkilenen bölgeler veya uzay görevleri gibi tehlikeli ortamlarda çalışabilir ve böylece insanlar açısından oluşabilecek riskleri azaltabilir. Robotların kullanılması, çeşitli çalışma süreçlerinde verimliliğin ve kalitenin artırılmasına katkıda bulunmaktadır.

Çok sayıda avantajına rağmen robotik sistemlerin bazı sınırlılıkları da bulunmaktadır. Bu sistemler insanlar tarafından oluşturulan programlara ve kurallara göre çalışır ve yaratıcılık, empati ve ahlaki muhakeme gibi birçok insani yeteneğin yerini kendi başlarına alamaz. Robotların çalışabilmesi için düzenli bakım, teknik destek ve bunları programlayıp kontrol edecek uzmanlara ihtiyaç vardır. Robotiğin gelişimi aynı zamanda güvenlik, sorumluluk ve otomasyonun farklı meslekler üzerindeki etkisi konusunda yeni sorular ortaya çıkarmaktadır. Bu nedenle robotik sistemler, insanların yaşam ve çalışma koşullarının iyileştirilmesine katkıda bulunacak şekilde geliştirilip uygulanmalıdır.

1.5.6 ROBOTİK VE GELECEK

Robotik alanındaki gelişmeler büyük bir hızla devam etmekte ve gelecekte robotik sistemlerin günlük yaşamda daha yaygın hâle gelmesi beklenmektedir. Akıllı şehirlerde otonom araçlar, kamusal alanların bakımı için robotlar ve trafiğin yönetilmesine yönelik otomatik sistemler geliştirilmektedir. Tıp alanında doktorlara hastaların teşhis ve tedavisinde yardımcı olacak robotlar geliştirilirken, tarımda ekinlerin durumunu izleyecek ve çeşitli faaliyetleri otomatik olarak gerçekleştirecek sistemler geliştirilmektedir.

Robotiğin gelişimi, bilişimin ve yapay zekânın gelişimiyle yakından ilişkilidir. Modern robotlar, çevrelerinden bilgi toplamak için çeşitli sensörlerden ve bu bilgileri işlemek için bilgisayar sistemlerinden yararlanmaktadır. Bu sayede robotlar nesneleri tanıyabilmekte, engellerden kaçınabilmekte ve görevlerini yerine getirirken basit kararlar verebilmektedir. Gelecekte robotlar ile yapay zekânın daha fazla iş birliği yapması ve modern toplumun gelişimi için yeni olanaklar oluşturması beklenmektedir.

1.5.7 KURTARMA OPERASYONLARINDA ROBOTLAR

Depremler, yangınlar ve diğer doğal afetlerden sonra kurtarma ekipleri çoğu zaman tehlikeli çalışma koşullarıyla karşı karşıya kalmaktadır. Bu tür durumlarda, insanların hareket etmesinin zor

BİLİYOR MUSUNUZ?

Okyanusların derinliklerinde ve Mars'ın yüzeyinde, insanların ulaşmasının zor olduğu yerlerde veri toplayan ve araştırmalar gerçekleştiren özel robotların çalıştığını biliyor musun?

ARAŞTIR

Robotların kullanıldığı bir alan seçiniz (tıp, sanayi, tarım, ulaşım, uzay veya kurtarma operasyonları). Aşağıdaki konuları açıklayacağınız kısa bir rapor hazırlayınız:

- ne tür robotların kullanıldığı;
- hangi görevleri yerine getirdikleri;
- ne gibi faydalar sağladıkları;
- hangi zorluklarla karşılaştıkları.

BİLGİ KONTROLÜ SORULARI:

1. Robotik nedir?
2. Otomasyon nedir?
3. Tüm robotlar insan görünümünde midir? Cevabınızı gerekçelendiriniz.
4. Robotlar günlük yaşamın ve çalışma hayatının hangi alanlarında kullanılmaktadır?
5. Robotlar üretimin ve hizmetlerin iyileştirilmesine nasıl katkıda bulunmaktadır?
6. Robotik sistemlerin kullanımının en önemli avantajları nelerdir?
7. Robotik sistemlerin kullanımında hangi sınırlılıklar ve zorluklar bulunmaktadır?

İŞ BİRLİĞİNE DAYALI VE UYGULAMALI ETKİNLİKLER

1. Dört farklı robot türünü (endüstriyel robotlar, tıbbi robotlar, ev robotları ve dronlar) karşılaştıracağınız bir tablo hazırlayınız. Her biri için aşağıdakileri belirtiniz:

- Kullanım alanı;
- Gerçekleştirdiği görevler;
- Kullanımının sağladığı faydalar.

2. Modern tarımda robotların nasıl kullanıldığını araştırınız. Kısa bir rapor hazırlayınız ve gerçekleştirebilecekleri en az üç faaliyeti belirtiniz.

3. Okulunuzun öğrencilere ve öğretmenlere yardımcı olacak bir robot satın aldığını düşününüz. Aşağıdakileri açıklayınız:

- Hangi görevleri gerçekleştireceği;
- Hangi teknolojileri kullanacağı;
- Okulun işleyişini nasıl iyileştireceği.

4. Robotların tıp alanında veya kurtarma operasyonlarında kullanımına ilişkin kısa bir sunum hazırlayınız. Örnekler veriniz ve sağladıkları faydaları açıklayınız.

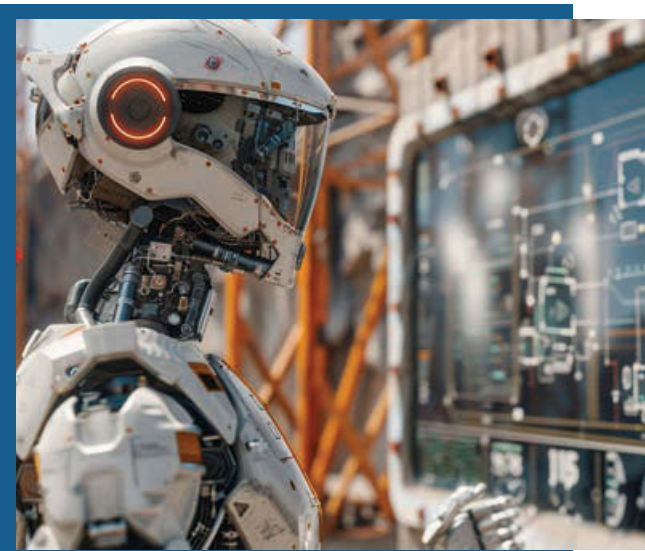
5. Gelecekte hangi mesleklerde robotların kullanılabileceğini düşününüz. En az beş örnek belirtiniz ve insanlara hangi şekilde yardımcı olabileceklerini açıklayınız.

SONUÇ:

Robotik, robotların oluşturulması ve uygulanmasıyla ilgilenen bilimsel ve teknik bir alandır. Robotların insan görünümünde olması zorunlu değildir ve farklı görevlere uyarlanabilirler. Otomasyon, belirli faaliyetlerin otomatik olarak gerçekleştirilmesini sağlar. Robotlar sanayi, tıp, tarım, ulaşım, evler ve daha birçok alanda kullanılmaktadır. Robotik sistemler tehlikeli, ağır ve tekrarlayan görevleri yerine getirebilir. Robotiğin gelişimi, bilişim ve modern teknolojilerin gelişimiyle yakından ilişkilidir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Son yıllarda, çeşitli görevleri yerine getirirken insanlarla iş birliği yapabilen yeni nesil robotlar geliştirilmektedir. Bu sistemler, çevrelerini daha iyi tanıyabilmek için sensörler, kameralar ve modern bilişim çözümlerinden yararlanmaktadır. Endüstriyel robotların yanı sıra yaşlılara yardımcı olan robotlar, otonom araçlar, okyanusları ve uzayı araştıran robotlar ve bilimsel deneylere katılan robotlar da geliştirilmektedir. Robotiğin gelecekte de modern toplumun gelişiminde önemli bir role sahip olması beklenmektedir.



BİLGİNİZİ KONTROL EDİN

1. Doğru cevabı yuvarlak içine alınız.

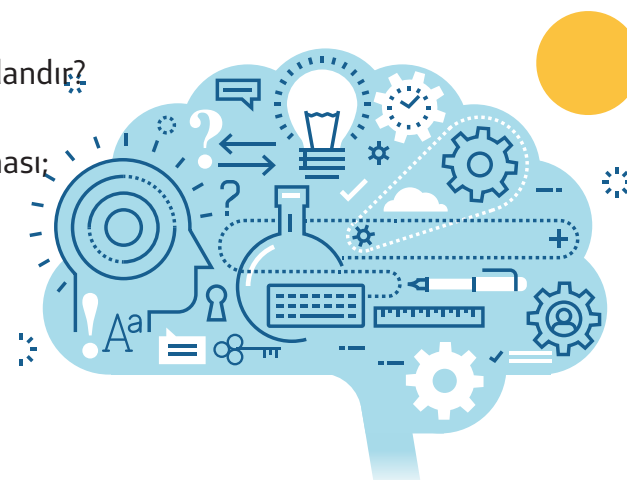
Robotik, aşağıdakilerden hangisiyle ilgilenen bir alandır?

a) Elektrik enerjisi üretimi;

b) Robotik sistemlerin oluşturulması ve uygulanması;

c) Gezegenlerin incelenmesi;

ç) Bilgisayarların onarılması.



2. Doğru mu, yanlış mı?

Tüm robotlar insan görünümündedir.

3. Tamamlayınız.

Belirli görevlerin otomatik olarak gerçekleştirildiği sürece _____ adı verilir.

4. Kavramları açıklayınız.

Kavram	Açıklama
Robot	_____
Robotik	_____
Otomasyon	_____

DURUMSAL ETKİNLİK

Bir depoda her gün ağır yüklerin taşınması gerektiğini düşününüz. Robotik sistemlerin bu çalışmaya nasıl yardımcı olabileceğini ve hangi faydaları sağlayabileceğini açıklayınız.

1.6

YAPAY ZEKÂ VE MAKİNE ÖĞRENMESİ

İnsanlar günlük yaşamlarında konuşmayı tanıyabilen, fotoğrafları analiz edebilen, içerik önerilerinde bulunabilen ve karar verme süreçlerine yardımcı olabilen çeşitli dijital hizmetleri kullanmaktadır. Mobil telefonlar kullanıcının yüzünü tanıyabilmekte, internet platformları kullanıcıların ilgi alanlarına göre film ve müzik önermekte, navigasyon sistemleri ise belirli bir konuma ulaşmak için en uygun rotayı seçmektedir. Bu olanaklar, modern bilişim teknolojilerinin gelişmesinin ve yapay zekânın uygulanmasının bir sonucudur.

BU KONUYU TAMAMLADIĞINIZDA...

- modern teknolojilere örnekler vermeyi;
- robotların çevrelerinden elde ettikleri bilgileri nasıl kullandıklarını açıklamayı;
- otomasyonun kullanım alanlarını tanımayı;
- dijital hizmetlere örnekler vermeyi;
- bilişim sistemlerinin verileri nasıl işlediğini açıklamayı bileceksiniz.

DÜŞÜNME SORULARI

1. Mobil telefon kullanıcısının yüzünü nasıl tanımaktadır?
2. İnternet mağazaları ilginizi çekebilecek ürünleri nasıl önermektedir?
3. Navigasyon uygulamaları en uygun rotayı nasıl seçmektedir?
4. Bilgisayar sistemleri işledikleri verilerden öğrenebilir mi?
5. Günlük yaşamınızda yapay zekânın hangi kullanım alanlarıyla karşılaştınız?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- yapay zekânın ne olduğunu açıklamayı;
- makine öğrenmesinin ne olduğunu açıklamayı;
- yapay zekânın kullanım alanlarına örnekler vermeyi;
- yapay zekânın nasıl çalıştığını analiz etmeyi;
- verilen bir örneğin avantajlarını ve risklerini belirlemeyi.

Yapay zekâ, bilgi işleme ve normalde insan zekâsı gerektiren görevleri çözme kapasitesine sahip bilgisayar sistemlerinin oluşturulmasıyla ilgilenen bir bilişim alanıdır. Bu sistemler, çalışmalarında büyük miktarda veri ve bilgilerin analiz edilmesi ve işlenmesi için çeşitli algoritmalar kullanmaktadır. Bu sayede örüntüleri tanıyabilir, olası sonuçları tahmin edebilir ve çeşitli faaliyetlerin gerçekleştirilmesinde insanlara yardımcı olabilirler.

1.6.1 YAPAY ZEKÂ NEDIR?

Yapay zekâ, genellikle insan düşüncesi gerektiren görevleri yerine getirebilen bilgisayar sistemlerinin geliştirilmesiyle ilgilenen bir bilişim alanıdır. Bu tür sistemler görüntüleri tanıyabilir, konuşmaları işleyebilir, verileri analiz edebilir ve belirli kararların alınmasına yardımcı olabilir. Çalışmaları sırasında farklı kaynaklardan elde ettikleri bilgileri kullanır ve bu bilgileri işlemek için özel algoritmalardan yararlanırlar. Yapay zekânın temel amacı, çeşitli pratik görevleri daha verimli bir şekilde çözebilen sistemler oluşturmaktır.

Yapay zekânın kullanımı günümüzde günlük yaşamın birçok alanında görülmektedir. İnternet arama motorları, sosyal ağlar, müzik ve film platformları veya mobil telefonlar kullanılırken, kullanıcılar her gün yapay zekâ algoritmalarından yararlanan hizmetleri kullanmaktadır. Tıp alanında bu sistemler tıbbi görüntülerin analizine yardımcı olmakta, ulaşımda otonom araçların geliştirilmesine katkıda bulunmakta, eğitimde ise etkileşimli öğrenme sistemlerinin oluşturulmasını sağlamaktadır. Bu tür uygulamalar sayesinde yapay zekâ modern toplumun önemli bir parçası hâline gelmektedir.

1.6.2 MAKİNE ÖĞRENMESİ NEDİR?

Makine öğrenmesi, bilgisayar sistemlerinin işledikleri verilerden öğrenmesini sağlayan bir yapay zekâ alanıdır. Her durum için özel kurallar yazılması yerine sistem, çok sayıda örneği analiz ederek belirli düzenlilikleri ve örüntüleri zamanla ortaya çıkarır. Sistem ne kadar fazla ve uygun veri işlerse, tasarlandığı görevleri o kadar başarılı bir şekilde gerçekleştirebilir.

Makine öğrenmesi, modern bilişim sistemlerinde geniş bir kullanım alanına sahiptir. Konuşma tanıma, metinlerin otomatik olarak çevrilmesi, fotoğraflardaki yüzlerin ve nesnelerin tanınması, istenmeyen elektronik postaların tespit edilmesi ve internette çeşitli içeriklerin önerilmesi gibi işlemlerde kullanılmaktadır. Örneğin, bir kullanıcı internette video izlediğinde sistem, kullanıcının en sık hangi içerikleri seçtiğini analiz eder ve benzer konulara sahip yeni videolar önerir. Bu şekilde makine öğrenmesi, bilgisayar sistemlerinin kullanıcıların ihtiyaçlarını daha iyi anlamasına ve daha yararlı hizmetler sunmasına yardımcı olur.

1.6.3 YAPAY ZEKÂNIN KULLANIM ALANLARI

Yapay zekâ günümüzde modern toplumun çok sayıda alanında kullanılmaktadır. Sağlık alanında tıbbi verilerin ve görüntülerin analizine yardımcı olmakta, ulaşımda trafik yönetim sistemlerinin ve otonom araçların geliştirilmesine katkıda bulunmakta, eğitimde ise etkileşimli eğitim içeriklerinin ve öğrenme sistemlerinin oluşturulmasında kullanılmaktadır. Tarımda ekinlerin durumunun izlenmesinde, sanayide ise üretim süreçlerinin otomatikleştirilmesi ve ürünlerin kalite kontrolünde uygulanmaktadır.

İnsanların günlük olarak kullandığı çok sayıda dijital hizmet, yapay zekânın farklı yöntemlerinden yararlanmaktadır. İnternet arama motorları bilgi bulmaya yardımcı olmakta, navigasyon sistemleri uygun rotalar önermekte, çeşitli platformlar ise kullanıcıların ilgi alanlarına göre müzik, film veya diğer içerikleri tavsiye etmektedir. Fotoğraf ve video işleme sırasında modern sistemler yüzleri ve nesneleri tanıyabilirken, mobil telefonların kullanımında konuşmanın tanınmasına ve konuşmanın metne dönüştürülmesine yardımcı olabilmektedir. Bu örnekler, yapay zekânın hâlihazırda günlük yaşamın bir parçası olduğunu göstermektedir.

1.6.4 YAPAY ZEKÂ NASIL ÇALIŞIR?

Telefonunuzun sesinizi tanıyarak kilidini açtığı veya YouTube'un size bir sonraki favori videonuzu önerdiği her durumda yapay zekânın (YZ) gücünden yararlanıyorsunuz. İlk bakışta sihir gibi görünse de bu süreçlerin arkasında hassas matematiksel işlemler, veri analizi ve makine öğrenmesi algoritmaları bulunmaktadır. YZ'nin gerçekte nasıl çalıştığını anlamak için her gün kullandığımız iki gerçek örneği inceleyeceğiz.

ÖRNEK 1: Öneri Sistemleri (TikTok, Netflix veya Spotify neyi sevdiğinizi nasıl biliyor?)

Bir uygulamanın dikkatinizi saatlerce nasıl çekebildiğini hiç merak ettiniz mi? Bunun için bir YZ öneri sistemi kullanmaktadır. Bu sistem rastgele tahminde bulunmaz, dört temel aşama üzerinden çalışır:

1. **Veri toplama (dijital iziniz):** Uygulamayı kullandığınız sırada algoritma yaptığınız her hareketi dikkatle takip eder. Hangi gönderilerde daha uzun süre kaldığınızı, hangi videoları ilk saniyeden sonra geçtiğinizi, neleri beğendiğinizi, neleri paylaştığınızı veya yorumladığınızı içeren verileri toplar.
2. **Örüntüleri tanıma (şemaları belirleme):** YZ bu verileri analiz eder ve sizi benzer alışkanlıklara sahip binlerce başka kullanıcıyla gruplandırır. Örneğin Marko ile siz aynı üç yemek pişirme videosunu izliyorsanız, YZ sizi benzer ilgi alanlarına sahip "dijital ikizler" olarak sınıflandırır.
3. **Filtreleme ve tahmin:** Marko dördüncü bir videoyu (örneğin yeni bir pizza tarifi) izler ve videoyu sonuna kadar seyredirse, algoritma matematiksel olarak bu videoyu sizin de beğenme olasılığınızın %95 olduğunu hesaplar.
4. **Sonuç:** Pizza videosu ana sayfanızda görünür. Videoyu izlerseniz sistem, yaptığı hesaplamanın doğru olduğuna ilişkin doğrulama elde eder ve bir sonraki öneri için daha isabetli hâle gelir.

ÖRNEK 2: Konuşma Tanıma (Siri, Google Assistant veya mesaj dikte etme nasıl çalışır?)

"Hey Siri" dediğinizde veya konuşmayı metne dönüştürme (Voice-to-Text) seçeneğini kullandığınızda yapay zekâ, insanın biyolojik dünyası ile bilgisayarın dijital dünyası arasındaki farkı ortadan kaldırmak zorundadır. Bu karmaşık süreç şu şekilde gerçekleşir:

1. **Akustik analiz:** Sesiniz bir ses dalgasıdır. Cihazın mikrofonu bu dalgayı kaydeder ve dijital bir sinyale (sayılar dizisine) dönüştürür. YZ bu sinyali çevredeki gürültülerden (örneğin caddedeki araçların gürültüsünden) arındırır.
2. **Fonemlere ayırma:** YZ kaydedilen konuşmayı fonem adı verilen mümkün olan en küçük ses birimlerine ayırır. Örneğin "bilgisayar" kelimesini söylediğinizde algoritma bunu b-i-l-g-i-s-a-y-a-r seslerinin birleşimi olarak tanır.
3. **Dil modelleme:** Bu, YZ'nin en önemli aşamasıdır. Algoritma bu sesleri büyük bir veri tabanıyla (milyonlarca saatlik kaydedilmiş insan konuşması) karşılaştırarak hangi kelimenin söz konusu sese en uygun olduğunu belirler. Aynı zamanda cümlenin bağlamını da analiz eder. "Elma istiyorum" dediğinizde YZ, önceki "istiyorum" kelimesi sayesinde teknoloji şirketi Apple'ı değil, meyveyi kastettiğinizi bilir.
4. **Komutun gerçekleştirilmesi:** YZ metni anladıktan sonra telefondaki uygun işlevi etkinleştirir, kelimeyi ekrana yazar, istediğiniz şarkıyı çalar veya size oluşturulmuş bir sesle yanıt verir.

Bu iki örnekten, YZ'nin insan gibi duygulara ve bilince sahip bir şekilde "düşünmediğini" görüyoruz. YZ, güçlü bir istatistiksel işlemci olarak çalışır. İlke her zaman aynıdır: Girdi verileri (beğenileriniz veya sesiniz) → Matematiksel işleme ve önceki bilgilerle karşılaştırma → Çıktı sonucu (yeni bir öneri veya yazılı metin).

YZ'nin elinde ne kadar fazla veri bulunursa, günlük yaşamda kullanılan bu tür örneklerde o kadar doğru çalışır.

1.6.5 YAPAY ZEKÂNIN AVANTAJLARI VE RİSKLERİ

Yapay zekâ (YZ), hiç şüphesiz insanlık tarihindeki en önemli teknolojik yeniliklerden biridir. Artık yalnızca bilim kurgunun bir konusu değil, tıp, eğitim, iş dünyası ve günlük yaşamımızı aktif olarak yeniden şekillendiren bir araçtır. Bununla birlikte, her güçlü teknoloji gibi YZ de büyük bir ilerleme potansiyelinin yanı sıra ciddi zorlukları da beraberinde getirmektedir. Geleceğe hazırlıklı olabilmek için "madalyonun" her iki yüzünü, yani kullanımının avantajlarını ve olası risklerini ayrıntılı olarak analiz etmemiz gerekmektedir.

YZ KULLANIMININ AVANTAJLARI

YZ'nin topluma uygulanması, yaşamı daha kolay, daha güvenli ve daha verimli hâle getiren önemli faydalar sağlamaktadır.

- **Otomasyon ve yüksek verimlilik:** YZ milyarlarca veriyi birkaç saniye içinde işleyebilir ve tekrarlayan (yinelenen/sıkıcı) görevleri dinlenmeden gerçekleştirebilir. Bu, insanların idari işlerden kurtularak yaratıcı düşünmeye ve karmaşık sorunları çözmeye odaklanmalarını sağlar.
- **Tıpta ilerleme ve hayat kurtarma:** YZ algoritmaları tıbbi görüntüleri (örneğin röntgen veya manyetik rezonans görüntülerini) insan gözünden daha yüksek bir hassasiyetle analiz ederek hastalıkları en erken aşamada tespit edebilir. Ayrıca YZ, yeni ilaçların keşfedilmesi ve test edilmesi sürecini önemli ölçüde hızlandırmaktadır.
- **Kişiselleştirilmiş öğrenme**: Eğitimde akıllı sistemler, öğrenme materyallerini her öğrencinin öğrenme hızına ve tarzına göre ayrı ayrı uyarlayabilir. Bir öğrenci belirli bir konuda "takılırsa", YZ asistanı alternatif açıklamalar ve alıştırmalar sunabilir.
- **Tehlikeli ortamlarda artırılmış güvenlik:** YZ tarafından kontrol edilen robotlar, insanlar için ölümcül olabilecek görevleri, örneğin mayın temizleme, derin okyanusları araştırma, büyük yangınları söndürme veya nükleer santrallerde çalışma gibi görevleri yerine getirebilir.

YZ KULLANIMININ OLASI RİSKLERİ

Tüm avantajlarına rağmen yapay zekânın kontrolsüz gelişimi ciddi etik, sosyal ve güvenlik sorunlarını gündeme getirmektedir.

- **İş gücü piyasası üzerindeki etkisi (iş kayıpları):** YZ'nin hızlı gelişimi, kitlesel işsizliğe ilişkin endişelere yol açmaktadır. YZ tabanlı sistemler, müşteri hizmetleri, çeviri, araç kullanımı (otonom araçlar) gibi alanlardaki mesleklerin, hatta yeni mezun programcıların veya yazarların yaptığı işlerin yerini alabilir. Bu durum, insanların acilen yeniden eğitim almalarını ve yeni beceriler kazanmalarını gerektirmektedir.

- **Önyargı ve ayrımcılık (algoritmik önyargı): YZ, in**sanlar tarafından oluşturulan verilerden öğrenmektedir. Bu veriler geçmişten gelen önyargılar veya ayrımcılık (ırk, cinsiyet veya milliyet temelinde) içeriyorsa YZ de bu önyargıları öğrenerek adaletsiz kararlar vermeye devam edecektir. Örneğin iş başvurularında adayların seçilmesi veya banka kredilerinin onaylanması sırasında bu durum ortaya çıkabilir.
- **Gizliliğin kaybedilme**si: YZ'nin doğru şekilde çalışabilmesi için büyük miktarda kişisel veriye ihtiyaç duyulmaktadır. Sürekli dijital gözetim riski ciddi bir sorun oluşturmaktadır. Bu durumda attığımız her adım, yaptığımız konuşmalar, aramalarımız ve duygularımız izlenebilir, analiz edilebilir ve ticari veya siyasi amaçlarla kötüye kullanılabilir.
- **Dezenformasyon ve "Deepfake"lerin yayılması**: Gelişmiş YZ araçları, sahte haberlerin, ikna edici metinlerin ve hatta kamuya mal olmuş kişilerin veya sıradan insanların hiç söylemedikleri şeyleri söylüyormuş gibi göründüğü sahte video ve ses kayıtlarının kolayca oluşturulmasına olanak sağlamaktadır. Bu durum bilgiye duyulan güveni tehdit etmekte ve siyasi istikrarsızlığa yol açabilmektedir.
- **İnsanlarda bilişsel tembellik:** Ödev hazırlama, problem çözme ve karar verme süreçlerinde YZ'ye aşırı derecede güvenmek, uzun vadede genç nesillerin eleştirel düşünme, mantık yürütme ve yaratıcı düşünme becerilerini azaltabilir.

Sunduğu çok sayıdaki olanağa rağmen yapay zekâ ile çalışırken bazı riskler de bulunmaktadır. Yapay zekânın çalışması, işlediği verilerin kalitesine ve miktarına ve oluşturulduğu kurallara bağlıdır. Modern sistemler hata yapabilir veya yanlış sonuçlar verebilir. Bu nedenle insanlar tarafından kontrol edilmesi ve denetlenmesi gerekmektedir. Yapay zekânın sorumlu bir şekilde uygulanması, çeşitli faaliyetlerin ve hizmetlerin iyileştirilmesine katkıda bulunan yardımcı bir teknoloji olarak kullanılmasını gerektirir.

Yapay zekâ yaratıcı, etik ve sorumlu bir şekilde kullanılmalıdır. Başarılı bir şekilde uygulanmasının temelinde, insan haklarını ve gizliliği korurken te**knolojik gelişmeyi de teşvik edecek sıkı** etik yasaların ve düzenlemelerin oluşturulması bulunmaktadır. İnsanlığın amacı insanın YZ ile değiştirilmesi değil, YZ'nin insanların daha fazlasını başarmasına yardımcı olacağı bir iş birliği ve uyum ortamının oluşturulması olmalıdır.

1.6.6 YAPAY ZEKÂ VE MODERN TOPLUM

Yapay zekâ, modern toplumun gelişiminde giderek daha büyük bir önem kazanmaktadır. Kullanımı, çeşitli hizmetlerin iyileştirilmesine ve insanların günlük faaliyetlerinin kolaylaştırılmasına katkıda bulunmaktadır. Sağlık alanında tıbbi verilerin analizine yardımcı olmakta, eğitimde modern dijital öğrenme araçlarının oluşturulmasını sağlamakta, ulaşımda ise trafik sistemlerinin daha iyi yönetilmesine katkıda bulunmaktadır. Sanayide üretimin otomatikleştirilmesi ve kalite kontrolü için kullanılmakta, tarımda ise ekinlerin durumunun izlenmesine ve kaynakların daha verimli kullanılmasına yardımcı olmaktadır.

Yapay zekânın gelişimi, bilimsel araştırmalar, ekonomik kalkınma ve yaşam kalitesinin iyileştirilmesi için yeni olanaklar sunmaktadır. Bununla birlikte modern teknolojilerin sorumlu ve dikkatli bir şekilde kullanılması gerekmektedir. İnsanlar, bu sistemlerin geliştirilmesi, denetlenmesi ve uygulanmasında önemli bir role sahiptir, çünkü sistemlerin çalışma kurallarını insanlar oluşturmaktadır. Bu nedenle teknik bilginin yanı sıra eleştirel düşünme, dijital okuryazarlık ve bilişim teknolojilerinin sorumlu kullanımı da geliştirilmelidir.

Yapay Zekâ ve Gelecek

Bilişimin gelişmesiyle birlikte yapay zekânın giderek daha fazla alanda kullanılması beklenmektedir. Modern sistemler bilimsel araştırmalara, yeni ilaçların geliştirilmesine, eğitimin iyileştirilmesine, çevrenin korunmasına ve karmaşık teknik sistemlerin yönetilmesine yardımcı olacaktır. Birçok meslekte yapay zekâ, insanların bilgileri daha hızlı işlemesine ve çeşitli görevleri daha kolay gerçekleştirmesine yardımcı olan bir araç olacaktır.

Teknolojinin hızlı gelişimine rağmen insan, karar alma ve modern sistemlerin uygulanmasında en önemli unsur olmaya devam etmektedir. Yapay zekâ, çeşitli faaliyetlerin iyileştirilmesine katkıda bulunabilecek bir araçtır, ancak insan bilgisinin, yaratıcılığının, deneyiminin ve sorumluluğunun yerini tamamen alamaz. Bu nedenle bu alanın gelecekteki gelişimi, insanlar ile modern bilişim teknolojileri arasındaki başarılı iş birliğine bağlı olacaktır.

Eğitimde Yapay Zekâ

Modern eğitim platformları, öğrencilere ve öğretmenlere yardımcı olmak amacıyla çeşitli yapay zekâ teknolojilerinden yararlanmaktadır. Bu tür sistemler ek öğrenme materyalleri önerebilir, bilgi bulmaya yardımcı olabilir ve ders içeriklerinin öğrenilmesi için etkileşimli etkinlikler sunabilir. Bu şekilde öğrenciler çeşitli bilgi kaynaklarına daha kolay erişebilir ve öğrenme süreçlerini daha iyi organize edebilirler.

Yapay zekâ öğretmenin yerini almaz, aksine eğitim sürecinin iyileştirilmesine katkıda bulunabilecek ek bir araçtır. Öğretmen, öğretim içeriklerinin seçilmesi, öğrencilerin yönlendirilmesi ve bilgilerin doğruluğunun kontrol edilmesinde önemli bir role sahiptir. Modern teknolojilerin sorumlu bir şekilde kullanılması, öğrencilerin dijital yeterliliklerinin ve eleştirel düşünme becerilerinin geliştirilmesine katkıda bulunmaktadır.

BİLİYOR MUSUNUZ?

Günlük yaşamında yapay zekâ kullanan sistemlerden yararlandığını biliyor musun? İnternet arama motorları, navigasyon uygulamaları, konuşma tanıma sistemleri ve birçok dijital hizmet, verileri işlemek için farklı yöntemler kullanmakta ve görevlerin yerine getirilmesine yardımcı olmaktadır.

ARAŞTIR

Günlük yaşamda yapay zekânın kullanımına ilişkin beş örnek araştırınız. Her bir örnek için aşağıdakileri açıklayınız:

- nerede kullanıldığı;
- hangi görevi gerçekleştirdiği;
- insanlara hangi şekilde yardımcı olduğu.

TEKRARLAMA SORULARI

1. Yapay zekâ nedir?
2. Makine öğrenmesi nedir?
3. Yapay zekâ modern toplumun hangi alanlarında kullanılmaktadır?
4. Yapay zekâ günlük yaşamda nasıl yardımcı olmaktadır?
5. Yapay zekâ kullanımının en önemli avantajları nelerdir?
6. Yapay zekâ destekli sistemlerin kullanımında hangi sınırlılıklar bulunmaktadır?
7. Yapay zekânın sorumlu bir şekilde kullanılması neden önemlidir?

İŞ BİRLİĞİNE DAYALI VE UYGULAMALI ETKİNLİKLER

1. Günlük yaşamda yapay zekânın kullanımına ilişkin on örnekten oluşan bir liste hazırlayınız. Her örnek için nerede kullanıldığını ve hangi görevi gerçekleştirdiğini belirtiniz.
2. Yapay zekânın tıp, eğitim veya tarım alanında nasıl kullanıldığını araştırınız. Kısa bir rapor hazırlayınız.
3. Okulunuzun yapay zekâ destekli bir sistem kullanmaya başladığını düşününüz. Sistemin yardımcı olabileceği üç faaliyet öneriniz ve sağlayacağı faydaları açıklayınız.
4. İnsan ile yapay zekâ destekli bir sistemi aşağıdaki özellikler açısından karşılaştıracağınız bir tablo hazırlayınız:
   - veri işleme hızı;
   - öğrenme;
   - yaratıcılık;
   - karar verme;
   - insan denetimine duyulan ihtiyaç.
5. Yapay zekânın tıp ve eğitim alanında kullanılmasının sağlayacağı avantajları, gelecekte iş kaybı yaşanması olasılığına karşı değerlendiriniz. Sizce elde edilen faydalar bu teknolojinin geliştirilmesini haklı çıkarmakta mıdır? Görüşünüzü gerekçelendiriniz.
6. Veri gizliliğiyle ilgili riskleri ve "Deepfake" (sahte) video ve ses kayıtlarının ortaya çıkmasını göz önünde bulundurarak bir değerlendirme yapınız: Yapay zekâ kullanımının yol açtığı hangi etik sorunlar genç nesiller için en tehlikelidir ve bu sorunlar onların eleştirel düşünme becerilerini nasıl etkilemektedir?
7. Düşününüz ve aşağıdaki konu hakkında kısa bir metin yazınız:

"Yapay zekâ insanların günlük yaşamını nasıl iyileştirebilir?"

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Yapay zekâ, akıllı bilgisayar sistemlerinin oluşturulmasıyla ilgilenen bir bilişim alanıdır.

Makine öğrenmesi, yapay zekânın verilerden öğrenmeyi sağlayan bir bölümüdür.

Yapay zekâ, sağlık, eğitim, sanayi, ulaşım, tarım ve daha birçok alanda kullanılmaktadır.

Modern dijital hizmetler günlük olarak yapay zekânın farklı yöntemlerinden yararlanmaktadır.

Yapay zekâ birçok fayda sağlamakla birlikte bazı sınırlılıklara da sahiptir.

Yapay zekânın sorumlu bir şekilde uygulanması, modern toplumun gelişiminde önemli bir role sahiptir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Yapay zekânın gelişimi, modern bilişimin en dinamik alanlarından birini oluşturmaktadır. Günümüzde büyük miktarda veriyi işleyebilen, bilimsel araştırmalara yardımcı olan ve karmaşık problemlerin çözümüne katkıda bulunan sistemler geliştirilmektedir. Yapay zekâ, uzay araştırmalarında, yeni ilaçların geliştirilmesinde, çevrenin korunmasında ve daha birçok faaliyette kullanılmaktadır. Teknolojinin daha da gelişmesiyle birlikte yapay zekânın kullanımının artmaya devam etmesi beklenmektedir. Bu süreçte insan, bu sistemlerin denetlenmesi ve sorumlu bir şekilde kullanılması konusunda önemli bir role sahip olacaktır.

BİLGİNİZİ KONTROL EDİN

1. Doğru cevabı yuvarlak içine alınız.

Yapay zekâ aşağıdakilerden hangisidir:

a) Metin işleme programı;
b) Akıllı bilgisayar sistemleri oluşturan bir bilişim alanı;
c) Bilgisayar oyunu;
ç) İşletim sistemi.

2. Doğru mu, yanlış mı?

Makine öğrenmesi, yapay zekânın bir bölümüdür.

3. Tamamlayınız.

Bilgisayar sistemlerinin verilerden öğrenmesini sağlayan alana _____ adı verilir.

4. Kavramları açıklayınız.

Kavram	Açıklama
Yapay zekâ	_____
Makine öğrenmesi	_____
Algoritma	_____

DURUMSAL ETKİNLİK

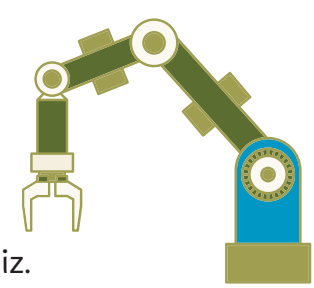
Bir okulun eğitim sürecini iyileştirmek amacıyla yapay zekâ destekli bir sistem kullanmak istediğini düşününüz. Sistemin yardımcı olabileceği iki faaliyet belirtiniz ve kullanım sırasında neden insan denetimine ihtiyaç duyulduğunu açıklayınız.

1.7

BÜYÜK DİL MODELLERİ VE ÜRETKEN YAPAY ZEKÂ

Büyük dil modelleri, İngilizce kısaltması olan LLM (Large Language Models) adıyla daha yaygın olarak bilinen, insan dilini anlamak, işlemek ve yorumlamak amacıyla çok büyük miktarda metinsel veri üzerinde eğitilmiş gelişmiş yapay zekâ sistemleridir. Temel ve en dikkat çekici yetenekleri, devrim niteliğindeki metin üretme kapasitesidir. Model, kendisine verilen bir soru veya bağlam temelinde, insan tarafından yazılmış gibi doğal görünen denemeler, makaleler, şiirler veya ayrıntılı açıklamalar oluşturabilir. Bu teknolojik altyapı, ChatGPT gibi akıllı chatbot sistemlerinin oluşturulmasını mümkün kılmıştır. Bu sistemler gerçek zamanlı etkileşimli iletişim olanağı sunmakta ve çeşitli kullanıcı sorularına anında yanıt vermektedir. Esneklikleri sayesinde bu modeller eğitim alanında yaygın olarak kullanılmakta, öğrencilere zor dersleri öğrenmelerinde veya dilleri çevirmelerinde yardımcı olan kişiselleştirilmiş dijital öğreticiler olarak hizmet vermektedir. Aynı zamanda teknoloji dünyasında da büyük bir dönüşüm gerçekleştirmiş ve programlama alanında güçlü kullanım olanakları sağlamıştır. Geliştiriciler bu modelleri bilgisayar kodlarının otomatik olarak yazılması, incelenmesi, hatalarının ayıklanması (debugging) ve açıklanması amacıyla kullanmaktadır.

BU KONUYU TAMAMLADIĞINIZDA...

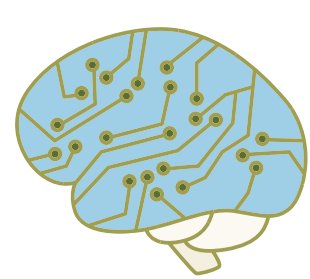


- Yapay zekânın ne olduğunu açıklamayı;
- Makine öğrenmesinin ne olduğunu açıklamayı;
- Yapay zekânın kullanım alanlarına örnekler vermeyi;
- YZ'nin faydalarını ve sınırlılıklarını analiz etmeyi;
- Modern bilişim teknolojilerinin kullanım alanlarını tanımayı bileceksiniz.

DÜŞÜNME SORULARI

1. Bir bilgisayar sistemi doğal dilde sorulan bir soruya nasıl yanıt verebilir?
2. Bir bilgisayarın metin, görsel veya müzik oluşturması nasıl mümkün olabilir?
3. Bilgisayar sistemi her zaman doğru bilgiler verir mi?
4. Üretken yapay zekâ eğitim alanında nasıl yardımcı olabilir?
5. Modern dijital sistemlerden elde edilen bilgilerin eleştirel bir şekilde kontrol edilmesi neden önemlidir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:



- Büyük dil modellerinin ne olduğunu açıklamayı;
- Kullanım alanlarına örnekler vermeyi;
- Metin üretiminde nasıl kullanıldıklarını anlamayı;
- Öğrenme ve çalışma süreçlerinde kullanım olanaklarını belirlemeyi.

Son yıllarda yapay zekâ önemli ölçüde gelişmiş ve farklı türlerde içeriklerin işlenmesi ve oluşturulmasına yönelik yeni olanaklar kazanmıştır. Modern bilgisayar sistemleri doğal dilde sorulan soruları anlayabilir, metinler oluşturabilir, çeviri yapmaya yardımcı olabilir, özetler hazırlayabilir ve çeşitli fikirler üretebilir. Bu yetenekler sayesinde eğitim, bilim, iş dünyası ve daha birçok alanda kullanılmaktadır.

Büyük dil modelleri, büyük miktarda metinsel verinin işlenmesiyle eğitilen modern yapay zekâ sistemleridir. Bu sayede dilsel örüntüleri tanıyabilir ve kullanıcıların taleplerine göre yeni içerikler oluşturabilirler. Üretken yapay zekâ yalnızca bilgileri işlemekle kalmaz, aynı zamanda metinler, görseller, müzik ve diğer dijital içerikleri de oluşturabilir. Bu teknolojilerin geniş kullanım alanları nedeniyle kullanıcıların hem olanaklarını hem de sınırlılıklarını anlamaları önemlidir.

1.7.1 BÜYÜK DİL MODELLERİ NEDİR

Büyük dil modelleri, doğal dilde metinleri işleyebilen ve oluşturabilen modern bilgisayar sistemleridir. Bu sistemler çalışmalarında büyük miktarda veri ve çeşitli yapay zekâ algoritmaları kullanarak kelimeler ve cümleler arasındaki ilişkileri tanır. Bu sayede soruları yanıtlayabilir, kavramları açıklayabilir, metinleri çevirebilir ve çeşitli içeriklerin oluşturulmasına yardımcı olabilirler.

Büyük dil modelleri (LLM), insan dilini anlamak ve üretmek üzere tasarlanmış son derece gelişmiş yapay zekâ sistemleridir. "Büyük" kelimesi rastgele kullanılmamıştır. İki şeyi ifade etmektedir:

1. **Büyük miktarda veri.** Bu modeller internetten kitaplar, makaleler, Wikipedia, bilimsel çalışmalar ve bilgisayar kodları gibi milyarlarca sayfalık metin üzerinde eğitim almıştır.
2. **Milyarlarca parametre**. Bu modeller, kelimeleri ilişkilendirmelerini ve cümlelerin bağlamını neredeyse insanlar kadar iyi anlamalarını sağlayan karmaşık bir iç yapıya (dijital sinir ağına) sahiptir.

En basit ifadeyle LLM, dünyadaki neredeyse tüm kitapları okumuş ve artık sizinle herhangi bir konu hakkında konuşabilen süper akıllı bir dijital okuyucu gibidir.

Büyük dil modelleri eğitimde, bilimde, iş dünyasında ve çeşitli dijital hizmetlerde kullanılmaktadır. Bilgi aramaya, metin hazırlamaya, yabancı dil öğrenmeye ve çeşitli faaliyetleri organize etmeye yardımcı olabilirler. Bununla birlikte, bu sistemlerin insanlar gibi düşünmediğinin, işledikleri verilere ve çalıştıkları algoritmalara dayanarak içerik oluşturduklarının bilinmesi önemlidir.

Günümüzde dünyanın en güçlü LLM araçlarını geliştiren birkaç önde gelen teknoloji şirketi bulunmaktadır. Bunlardan bazılarını mutlaka zaten kullanıyorsunuz:

- **ChatGPT (OpenAI):** Kamuoyunda devrim yaratan ilk araçtır. Günlük konuşmalar, yaratıcı yazma ve karmaşık kavramların açıklanması konusunda oldukça başarılıdır.
- **Gemini (Google):** Google'ın arama motoruyla doğrudan bağlantılı olan ve internetteki en güncel ve doğru bilgilere gerçek zamanlı olarak hızlı erişim sağlayan bir modeldir.
- **Claude (Anthropic):** Son derece doğal yazım tarzı, mantıksal düşünme ve çok uzun belgeleri işleyebilme kapasitesiyle tanınan bir araçtır.
- **Copilot (Microsoft):** Okulda kullandığınız programlara (Word, PowerPoint) entegre edilmiş ve bu programlarda yazma ve tasarım konusunda yardımcı olan bir asistandır.

1.7.2 METİN ÜRETİMİ NASIL ÇALIŞIR?

Birçok kişi LLM modellerinin "düşündüğünü" veya gerçeği bildiğini düşünmektedir. Gerçek ise farklıdır, bunlar güçlü istatistiksel tahmin makineleridir.

Modele bir soru sorduğunuzda (bu soru bir talimat (prompt) olarak adlandırılır), model hazır bir yanıtı bulmak için veri tabanında arama yapıp kopyalayarak "yapıştırmaz". Bunun yerine şu işlemleri gerçekleştirir:

- Sorunuzu kelime kelime analiz eder.
- Geçmişte okuduğu tüm bilgilere dayanarak mantıksal olarak hangi kelimenin sırada gelmesi gerektiğini hesaplar.
- Örneğin cümle "Makedonya'nın başkenti..." şeklinde başlıyorsa model, matematiksel olarak bir sonraki kelimenin %99,9 olasılıkla "Üsküp" olduğunu tahmin eder.

Bu süreç olağanüstü bir hızla, kelime kelime gerçekleşir ve gerçek bir insan tarafından gerçek zamanlı olarak yazılmış gibi görünen bir metin oluşturulur.

LLM modellerini 7/24 erişilebilir kişisel asistanınız olarak kullanabilirsiniz.

Öğrenme sürecinde kişisel süper öğreticiniz olarak:

- Karmaşık dersleri açıklama. Fizik veya kimya dersini anlamıyorsanız modele "İkinci Newton Yasası'nı 10 yaşındaymışım gibi açıkla ve günlük yaşamdan bir örnek ver." diyebilirsiniz.
- Yabancı dil öğrenme. ChatGPT ile İngilizce veya Almanca konuşabilir ve yanıt sırasında dilbilgisi hatalarınızı düzeltmesini sağlayabilirsiniz.
- Sınavlara hazırlık. Notlarınızı modele yükleyebilir ve "Kendimi test etmek için 5 çoktan seçmeli sorudan oluşan bir quiz hazırla." diyebilirsiniz.

Çalışma hayatında ve programlamada:

- Kod yazma ve hata ayıklama. Okulda kodlama öğreniyorsanız (örneğin Python veya C++), LLM size noktalı virgülün (;) nerede eksik olduğunu veya kodun neden çalışmadığını hemen söyleyebilir.
- Fikir üretme. Bir proje veya deneme hazırlamanız gerekiyorsa ve nasıl başlayacağınızı bilmiyorsanız araç, yazı için beş farklı yapı ve konu önerebilir.
- Sıkıcı görevlerin otomatikleştirilmesi. İş dünyasında bu modeller e-postalara hızlı yanıt vermek, raporlar hazırlamak ve uzun sözleşmeleri çevirmek için kullanılmaktadır.

Önemli bir kural: LLM araçları mükemmel iş birliği araçlarıdır, ancak bilişsel yeteneklerinizin yerine geçmek için uygun değildir. Bazen "halüsinasyon" yapabilir (büyük bir özgüvenle yanlış bilgiler uydurabilirler). Bu nedenle önemli bilgileri her zaman kontrol edin ve bu araçları düşünme sürecini atlamak için değil, bilginizi geliştirmek için kullanın!

Büyük dil modelleri ve üretken yapay zekâ, modern toplumun çeşitli alanlarında kullanılmaktadır. İş dünyasında belgelerin hazırlanması ve bilgilerin analiz edilmesi için kullanılırken, bilim alanında büyük miktarda verinin işlenmesine ve araştırmaların düzenlenmesine yardımcı olabilirler. Kullanımları sayesinde birçok faaliyet daha hızlı ve verimli bir şekilde gerçekleştirilebilir.

Üretken yapay zekâ, farklı türlerde dijital içerikler oluşturabilir. Metinlerin yanı sıra modern sistemler görsellerin, müziklerin, sunumların ve diğer multimedya materyallerinin oluşturulmasına da yardımcı olabilir. Bu teknolojiler eğitim, tasarım, sanat, medya ve daha birçok alanda kullanılmaktadır. Bununla birlikte oluşturulan içerikler dikkatli bir şekilde analiz edilmeli, doğruluğu ve belirli kullanım amacına uygunluğu kontrol edilmelidir.

1.7.3 ÜRETKEN YAPAY ZEKÂNIN SORUMLU KULLANIMI

Modern üretken yapay zekâ sistemleri, çeşitli faaliyetlerde insanlara yardımcı olabilecek yararlı araçlardır. Bununla birlikte, bu sistemlerin kullanımı sorumluluk ve eleştirel düşünme gerektirir. Bu sistemlerin oluşturduğu bilgiler otomatik olarak tamamen doğru kabul edilmemeli, güvenilir kaynaklarla karşılaştırılmalı ve dikkatli bir şekilde kontrol edilmelidir. Özellikle eğitim, sağlık ve diğer önemli alanlara ilişkin bilgilerin kullanımında bu kurala uyulması büyük önem taşımaktadır.

Üretken yapay zekâ kullanılırken etik ilkelere ve dijital teknolojilerin güvenli kullanımına ilişkin kurallara uyulmalıdır. Kişisel verilerin ve gizliliğin korunmasına, telif haklarına ve fikrî mülkiyete saygı gösterilmesine dikkat edilmelidir. Üretken yapay zekâ, öğrenmeye, araştırmaya ve yeni fikirler oluşturmaya katkıda bulunan yardımcı bir araç olarak kullanılmalı, insan bilgisinin, yaratıcılığının ve sorumluluğunun yerine geçmemelidir.

1.7.4 MODERN TEKNOLOJİ UYGULAMADA

Eğitimde Üretken Yapay Zekâ

Bir öğrenci, iklim değişikliği hakkında bir sunum hazırlamaktadır. Üretken yapay zekâ yardımıyla sunumun yapısı hakkında fikirler, kavramların açıklamaları ve araştırma için ek konu önerileri elde edebilir. Daha sonra bilgileri ders kitapları, ansiklopediler ve diğer güvenilir kaynaklardan kontrol eder ve kendi sonuçlarıyla tamamlar.

Bu şekilde üretken yapay zekâ, öğrenme sürecine yardımcı olan bir araçtır, ancak bağımsız düşünme ve araştırmanın yerini almaz. Bu tür sistemlerin sorumlu bir şekilde kullanılması, dijital yeterliliklerin, eleştirel düşünmenin ve farklı bilgi kaynaklarını analiz etme becerisinin geliştirilmesine katkıda bulunur.

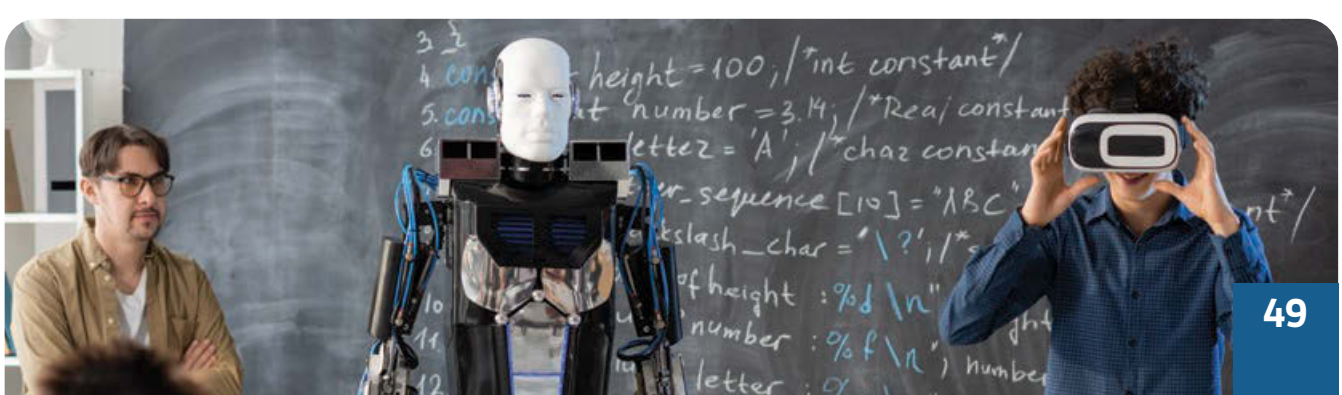
BİLİYOR MUSUNUZ?

Üretken yapay zekânın metin, görsel, müzik ve video gibi farklı türlerde içerikler oluşturabildiğini, ancak bu içeriklerin kontrol edilmesi ve sorumlu bir şekilde kullanılmasında insanın önemli bir role sahip olduğunu biliyor musun?

ARAŞTIR

Üretken yapay zekânın uygulandığı üç alanı araştırınız. Her bir alan için aşağıdakileri belirtiniz:

- hangi tür içerikler oluşturabileceği;
- insanlara nasıl yardımcı olduğu;
- neden sorumlu bir şekilde kullanılması gerektiği.



TEKRARLAMA SORULARI

1. Büyük dil modelleri nedir?
2. Üretken yapay zekâ nedir?
3. Üretken yapay zekâ hangi tür içerikleri oluşturabilir?
4. Büyük dil modelleri hangi alanlarda kullanılmaktadır?
5. Üretken yapay zekâdan elde edilen bilgilerin eleştirel bir şekilde kontrol edilmesi neden gereklidir?
6. Bu teknolojiler kullanılırken telif haklarına ve gizliliğe saygı gösterilmesi neden önemlidir?
7. Üretken yapay zekâ eğitim alanında nasıl yardımcı olabilir?

İŞ BİRLİĞİNE DAYALI VE UYGULAMALI ETKİNLİKLER

1. Üretken yapay zekânın günlük yaşamdaki üç farklı kullanım alanını araştırınız ve kısa bir rapor hazırlayınız.
2. Çevrenin korunması hakkında bir sunum hazırlamanız gerektiğini düşününüz. Üretken yapay zekânın hazırlık sürecinde size nasıl yardımcı olabileceğini ve hangi bilgileri mutlaka kendinizin kontrol etmesi gerektiğini açıklayınız.
3. Klasik bir internet arama motoru ile üretken yapay zekâyı aşağıdaki özellikler açısından karşılaştıracağınız bir tablo hazırlayınız:

- çalışma biçimi;
- bilgi türü;
- kullanım alanı;
- bilgilerin doğrulanmasına duyulan ihtiyaç.

4. Üretken yapay zekânın güvenli ve sorumlu bir şekilde kullanılması için beş kural düşününüz ve belirtiniz.
5. Aşağıdaki konuda kısa bir deneme yazınız:

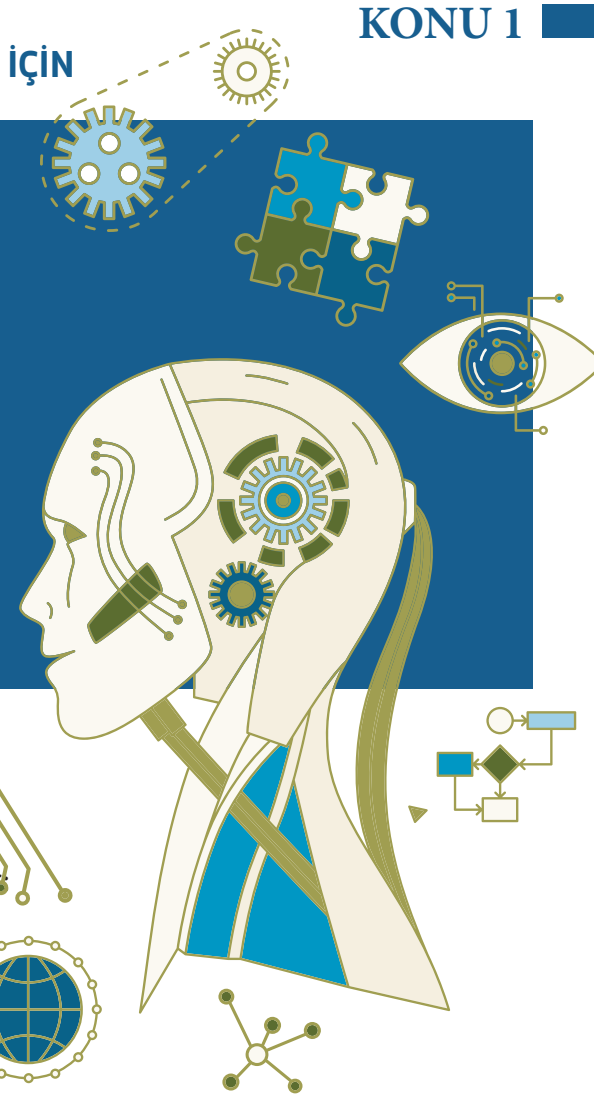
"Üretken yapay zekâ öğrenme ve eğitim alanında nasıl yardımcı olabilir?"

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Büyük dil modelleri, metinleri işleyen ve oluşturan modern yapay zekâ sistemleridir. Üretken yapay zekâ, farklı türlerde dijital içerikler oluşturabilir. Bu teknolojiler eğitim, bilim, sanat, iş dünyası ve daha birçok alanda kullanılmaktadır. Oluşturulan içerikler eleştirel bir şekilde analiz edilmeli ve doğrulukları kontrol edilmelidir. Üretken yapay zekâ kullanılırken telif haklarına ve gizliliğe saygı gösterilmelidir. Üretken yapay zekâ, insan bilgisini ve sorumluluğunu tamamlayan, ancak bunların yerini almayan yardımcı bir araçtır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Üretken yapay zekânın gelişimi büyük bir hızla devam etmekte ve gitgide daha karmaşık dijital içeriklerin oluşturulmasını mümkün kılmaktadır. Metinlerin yanı sıra modern sistemler görseller, müzikler, videolar ve bilgisayar programları oluşturabilmektedir. Gelecekte bu sistemlerin eğitim, bilim, tıp ve çeşitli ekonomik faaliyetlerde daha geniş bir kullanım alanına sahip olması beklenmektedir. Aynı zamanda teknolojinin toplumsal gelişime ve yaşam kalitesinin iyileştirilmesine katkıda bulunması amacıyla güvenli, etik ve sorumlu kullanımına yönelik kurallar ve öneriler geliştirilmektedir.



BİLGİNİZİ KONTROL EDİN

1. Doğru cevabı yuvarlak içine alınız.

Büyük dil modelleri aşağıdakilerden hangisi için kullanılır?

a) Bilgisayar üretimi;

b) Metinsel içeriklerin işlenmesi ve oluşturulması;

c) Elektrik enerjisi üretimi;

ç) Cep telefonlarının onarılması.

2. Doğru mu, yanlış mı?

Üretken yapay zekâ metinler, görseller ve diğer dijital içerikleri oluşturabilir.

3. Cümleyi tamamlayınız.

Yeni dijital içerikler oluşturabilen modern sistemlere _____ adı verilir.

4. Kavramları açıklayınız.

Kavram	Açıklama
Büyük dil modeli	_____
Üretken YZ	_____
Sorumlu kullanım	_____

DURUMSAL ETKİNLİK

Bir öğrenci, bir okul projesi hazırlamak için üretken yapay zekâ kullanmaktadır. Bilgilerin doğruluğunu ve teknolojinin sorumlu bir şekilde kullanılmasını sağlamak için hangi adımları atması gerektiğini açıklayınız.

C++ İLE PROGRAMLAMA

Programlama, çeşitli problemlerin çözülmesini ve görevlerin otomatikleştirilmesini sağlayan bilgisayar programlarının oluşturulması sürecidir. Programlama dilleri aracılığıyla bilgisayara, verileri işleyebilmesi, hesaplamalar yapabilmesi, kararlar alabilmesi ve çeşitli faaliyetleri gerçekleştirebilmesi için talimatlar verilir. Günümüzde programlama, eğitim, bilim, sanayi, tıp, iletişim ve eğlence gibi modern toplumun birçok alanında geniş bir kullanım alanına sahiptir ve dijital çağın temel becerilerinden biri olarak kabul edilmektedir.

Bu konu kapsamında programlamanın temel ilkeleri C++ programlama dili aracılığıyla ele alınmaktadır. Bilgisayarda verilerin temsil edilme biçimleri, algoritmaların ve programların oluşturulması, fonksiyonların uygulanması ve programlama problemlerinin çözümünde tek boyutlu dizilerin kullanılması konularına özel önem verilmektedir. Uygulamalı örnekler ve etkinlikler aracılığıyla algoritmik düşünme geliştirilmekte ve bilgisayar programlarının analiz edilmesi ve oluşturulmasına ilişkin temel bilgi ve beceriler kazandırılmaktadır.

YENİ KAVRAMLAR

- İkili sayı sistemi;
- Ondalık sayı sistemi;
- Sekizlik sayı sistemi;
- Onaltılık sayı sistemi;
- Fonksiyon;
- Parametre;
- Kütüphane;
- Tek boyutlu dizi;
- Eleman indeksi.

BU KONUYU TAMAMLADIĞINIZDA...

- Bilgisayar ve çeşitli uygulamaları kullanmayı;
- Veri girişi yapmayı ve verileri işlemeyi;
- Problem çözmede mantıksal düşünmeyi uygulamayı;
- Farklı bilgi türlerini ayırt etmeyi;
- Verilen talimatları takip etmeyi ve uygulamayı;
- Basit günlük yaşam durumlarını analiz etmeyi;
- Temel bilişim teknolojilerini kullanmayı bileceksiniz..

KONU 2

2.1

BİLGİSAYARDA VERİLERİN DİJİTAL OLARAK TEMSİL EDİLMESİ

BU KONUYU TAMAMLADIĞINIZDA...

- Bilgisayar sisteminin ne olduğunu açıklamayı;
- Dijital cihazlara örnekler vermeyi;
- Donanım ve yazılımı ayırt etmeyi;
- C++'ta temel veri türlerini kullanmayı;
- Değişkenler ve operatörlerle basit programlar oluşturmayı bileceksiniz.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Bilgisayarın verileri nasıl temsil ettiğini açıklamayı;
- Bit ve baytı ayırt etmeyi;
- Sayıların, metinlerin, görsellerin ve seslerin nasıl temsil edildiğini açıklamayı;
- Farklı veri türlerini tanımayı;
- Bilgilerin saklanması için belleğin nasıl kullanıldığını analiz etmeyi.

Modern dijital dünyada çok sayıda bilgi, çeşitli bilgisayar sistemleri aracılığıyla sürekli olarak işlenmekte, depolanmakta ve aktarılmaktadır. Yazılan metinler, çekilen fotoğraflar, dinlenen müzikler ve görüntülenen videolar bilgisayarda doğal biçimleriyle saklanmaz, öncelikle dijital verilere dönüştürülür. Bu nedenle bilgisayarın bilgileri nasıl temsil ettiğini ve bellekte nasıl sakladığını anlamak gerekmektedir.

Bilgisayardaki her veri, ikili değerlerin birleşimleriyle temsil edilir. Kullanıcılar harfleri, sayıları, görselleri ve sesleri görse de arka planda tüm bilgiler sıfır ve birlerle temsil edilmektedir. Bu temsil biçimi, verilerin bilgisayarın elektronik bileşenleri tarafından kolaylıkla işlenmesini sağlar. Verilerin dijital olarak temsil edilmesini anlamak, programlama ve bilişimin daha ileri düzeyde öğrenilmesi için temel oluşturmaktadır.

Bilgisayar sistemleri iki farklı duruma sahip olabilen elektronik sinyallerle çalışır. Bu nedenle bilgisayarlarda verilerin ikili sistemle temsil edilmesi kullanılır. Temel bilgi birimine bit adı verilir. Bit yalnızca iki değere sahip olabilir: 0 veya 1. Tek bir bit kendi başına çok küçük miktarda bilgi taşısa da birden fazla bitin birleştirilmesiyle karmaşık veriler temsil edilebilir. Sekiz bit bir bayt oluşturur ve bayt, veri büyüklüğünün temel ölçü birimlerinden biridir. Günlük kullanımda kilobayt, megabayt, gigabayt ve terabayt gibi daha büyük ölçü birimleri de sıklıkla kullanılmaktadır. Teknolojinin gelişmesiyle işlenen veri miktarı sürekli olarak artmakta ve modern cihazlar çok büyük miktarda bilginin depolanabileceği belleklere sahip olmaktadır. Bellekte saklanan verilerin hızlı bir şekilde okunmasını ve işlenmesini sağlayacak biçimde düzenlenmesi gerekir. Bu nedenle dijital temsil, tüm modern bilişim teknolojisinin işleyişini sağlayan temel ilkelerden biridir.

Örnek 1: Veri ölçü birimlerinin dönüştürülmesi

main.cpp

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

int main() {
    int megabayt;

    cout << "Megabayt sayısını girin: " ;
    cin >> megabayt ;

    int kilobayt = megabayt * 1024;

    cout << "Kilobayt cinsinden: " << kilobayt << endl;

    return 0;
}
```

Sayıların yanı sıra bilgisayarlarda metinsel veriler de temsil edilmektedir. Her harf, rakam veya sembole özel kodlama standartları aracılığıyla belirli bir ikili kod atanır. En bilinen standartlardan biri ASCII kodudur. Günümüzde ise çok sayıda dünya dilinin ve sembolün temsil edilmesine olanak sağlayan Unicode standardı da yaygın olarak kullanılmaktadır. Bu standartlar sayesinde metin, farklı cihazlarda ve işletim sistemlerinde doğru şekilde görüntülenebilmektedir. Metinlerin yanı sıra görseller de dijital olarak temsil edilmektedir. Her dijital görsel, piksel adı verilen çok sayıda küçük noktadan oluşur. Her piksel renk bilgisi içerir ve binlerce veya milyonlarca pikselin bir araya getirilmesiyle bütün bir görsel elde edilir. Piksel sayısı ne kadar fazla olursa görselin kalitesi de o kadar yüksek olur. Ses de benzer şekilde temsil edilir, ses dalgaları, bilgisayarın işleyebileceği dijital değerlere dönüştürülür. Böylece farklı türlerdeki bilgiler dijital biçimde saklanabilir, kopyalanabilir ve işlenebilir.

Örnek 2: Bir karakterin ASCII değerinin gösterimi

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

int main() {
    char isaret;

    cout << "İşaret girin: ";
    cin >> isaret;

    cout << "ASCII değeri: " << (int)isaret << endl;

    return 0;
}
```

Programlama dillerinde veriler farklı veri türleri aracılığıyla düzenlenir. Her veri türü bellekte belirli bir alan kaplar ve farklı türde bilgilerin saklanmasına hizmet eder. Örneğin int veri türü tam sayılar, double gerçek sayılar, char karakterler ve bool mantıksal değerler için kullanılır. Uygun veri türünün seçilmesi büyük önem taşır, çünkü bu seçim verilerin doğruluğunu ve kullanılacak bellek miktarını etkiler. Basit bir değer için çok fazla bellek

kaplayan bir veri türü seçilirse program gereksiz yere kaynak tüketebilir. Buna karşılık, aralığı dar olan bir veri türü seçilirse hesaplamalar sırasında hatalar meydana gelebilir. Bu nedenle programcılar, programın ihtiyaçlarına göre veri türlerini dikkatli bir şekilde seçer. Verilerin bellekte nasıl düzenlendiğini anlamak, daha verimli programlar oluşturma yolunda önemli bir adımdır.

Örnek 3: Farklı veri türlerinin kullanılması

*main.cpp

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

int main() {
    int    yas    = 16;
    double ortalama = 4.75;
    char   sube    = 'A';
    bool   duzenli   = true;

    cout << "Yas: "    <<    yas    << endl;
    cout << "Ortalama: " << ortalama << endl;
    cout << "Sube: "    <<    sube    << endl;
    cout << "Düzenli öğrenci: " <<    duzenli    << endl;

    return 0;
}
```

Bilgisayar belleği, programların ve verilerin saklandığı alanı ifade eder. Bir programda kullanılan her değişkene bellekte bir yer ayrılır. Bilgisayar her veri için otomatik olarak bir adres belirler ve program bu verilere değişkenlerin adları aracılığıyla erişir. Belleğin düzenlenme biçimi, programların hızı ve verimliliği açısından büyük önem taşır. Modern bilgisayarlar RAM belleği ve veri depolamaya yönelik kalıcı bellek gibi farklı bellek türlerine sahiptir. RAM belleği, program çalışırken bilgilerin geçici olarak saklanmasını sağlarken kalıcı bellek, bilgisayar kapatıldıktan sonra da verileri saklar. Programın kullandığı bellek miktarı, işlenen verilerin sayısına ve türüne bağlıdır. Bu nedenle dijital temsilin ve verilerin düzenlenmesinin anlaşılması, algoritmaların, fonksiyonların ve veri yapılarının daha ileri düzeyde öğrenilmesi için önemli bir temel oluşturmaktadır.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Bit nedir?
2. Bir bayt kaç bit içerir?
3. Metinsel veriler bilgisayarda nasıl temsil edilir?
4. C++'ta double veri türünün rolü nedir?

ETKİNLİKLER

1. Verilen kilobaytların kaç bayt olduğunu hesaplayan bir program yazınız.
2. Girilen bir karakterin ASCII değerini görüntüleyen bir program yazınız.
3. Farklı veri türlerini kullanarak bir öğrencinin bilgilerini saklayan ve görüntüleyen bir program yazınız.
4. Girilen gigabaytların kaç megabayt olduğunu hesaplayan bir program yazınız.
5. Girilen karakterin büyük harf mi yoksa küçük harf mi olduğunu görüntüleyen bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Bilgisayarlar verileri ikili biçimde temsil eder.
- Temel bilgi birimi bittir.
- Sekiz bit bir bayt oluşturur.
- Metinler, görseller ve sesler ikili değerlerle temsil edilir.
- Farklı veri türleri bellekte farklı miktarda yer kaplar.
- Bellek, verilerin saklanmasını ve işlenmesini sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilgisayar sistemlerinde verilerin sıkıştırılması için özel teknikler kullanılmaktadır. Sıkıştırma sayesinde dosyaların boyutu, kalitede önemli bir kayıp yaşanmadan azaltılmaktadır. Bu sayede fotoğraflar, müzikler ve videolar internet üzerinden daha kolay aktarılabilir ve bellekte daha az yer kaplar. JPG, MP3 ve MP4 gibi formatlar, depolanması gereken veri miktarını azaltmak amacıyla farklı sıkıştırma algoritmaları kullanmaktadır.

2.2

SAYI SİSTEMLERİ VE DÖNÜŞÜMLER

BU KONUYU TAMAMLADIĞINIZDA...

- Bilgisayarın verileri nasıl temsil ettiğini açıklamayı;
- Bit ve baytı ayırt etmeyi;
- Veri türlerine örnekler vermeyi;
- Metin, görsel ve seslerin nasıl temsil edildiğini açıklamayı;
- C++'ta temel veri türlerini kullanmayı bileceksiniz.

TEKRARLAMA ÇALIŞMALARI

1. 4 bayt kaç bit içerir?
2. C++'ta karakter saklamak için hangi veri türü kullanılır?
3. Piksel nedir?
4. Verilen megabaytların kaç kilobayt olduğunu hesaplayan bir program yazınız.
5. Bilgisayardaki verilerin neden dijital olarak temsil edildiğini açıklayınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- İkili, ondalık, sekizlik ve onaltılık sayı sistemlerini ayırt etmeyi;
- Bir sayı sisteminin tabanının ne olduğunu açıklamayı;
- Sayı sistemleri arasında elle dönüşümler yapmayı;
- Rakamların basamak değerlerini analiz etmeyi;
- Programlama problemlerinde dönüşümleri uygulamayı.

Günlük yaşamda en yaygın olarak, 0'dan 9'a kadar on rakamdan oluşan ondalık sayı sistemi kullanılır. Ancak bilgisayarlar aynı şekilde çalışmaz. Elektronik devreler güvenilir biçimde yalnızca iki durumu algılayabildiğinden, bilgisayar sistemlerinde ikili sayı sistemi kullanılır. Bu nedenle bilgisayarda işlenen tüm bilgiler, sıfır ve birlerin birleşimleriyle temsil edilir.

İkili sistemin yanı sıra bilişimde sekizlik ve onaltılık sayı sistemleri de sıklıkla kullanılmaktadır. Bu sistemler, uzun ikili sayıların daha kolay temsil edilmesini ve verilerin daha düzenli biçimde ifade edilmesini sağlar. Sayı sistemlerini anlamak, bilgisayarın bilgileri nasıl işlediğini ve sakladığını anlamak açısından bilişimin önemli bir bölümünü oluşturmaktadır.

Sayı sistemi, belirli sayıda sembol kullanılarak sayıların temsil edilme biçimidir. Kullanılan sembollerin sayısına sayı sisteminin tabanı adı verilir. Ondalık sayı sisteminin tabanı 10'dur, çünkü on rakam kullanılır. İkili sistemin tabanı 2'dir ve yalnızca 0 ve 1 rakamları kullanılır. Bir sayıdaki her rakamın basamak değeri, bulunduğu konuma ve sayı sisteminin tabanına bağlıdır. Ondalık sistemde basamaklar 10'un kuvvetlerini, ikili sistemde ise 2'nin kuvvetlerini ifade eder. İkili sayı 4 basamaklıysa 2'nin 0'dan 3'e kadar olan kuvvetleri kullanılır. İkili sayı 15 basamaklıysa 2'nin 0'dan 14'e kadar olan kuvvetleri kullanılır. Yalnızca ikili sayıda 1 bulunan basamakların değerleri toplanır. Örneğin, 1011_2 ikili sayısı aşağıdaki şekilde hesaplanır:

İkili rakam	**1**	**0**	**1**	**1**
2'nin kuvveti	2^3	2^2	2^1	2^0
Değer	8	0	2	1

Değerler toplandığında:
$8 + 0 + 2 + 1 = 11_{10}$

Bu şekilde ikili sayı sisteminden ondalık sayı sistemine dönüşüm yapılır. İkili sistem, bilgisayarların çalışma temelini oluşturur, çünkü elektronik bileşenler iki durumu, yani açık ve kapalı durumlarını kolaylıkla temsil edebilir. Bu nedenle bilgisayardaki tüm veriler ve komutlar ikili biçimde temsil edilir.

Örnek 1: İkili sayının ondalık sayıya dönüştürülmesi

main.cpp

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

int main() {
    int ikiliSayi ;
    int onlukSayi  = 0;
    int taban = 1;
    int kalan ;

    cout << "İkili sayı girin: ";
    cin >> ikiliSayi ;

   while (ikiliSayi > 0) {
        kalan = ikiliSayi % 10;
        onlukSayi = onlukSayi + kalan * taban ;
        taban = taban * 2;
       ikiliSayi = ikiliSayi / 10;
    }

    cout << "Onluk sayı: " << onlukSayi << endl;

    return 0;
}
```

```
"D:\TD\Cpp\ 2 sınıf bin\ Debug \ 2 .exe"
İkili sayı girin: 1011
Onluk sayı: 11

Process returned 0 (0x0)   execution time : 2.515 s
Press any key to continue.
```

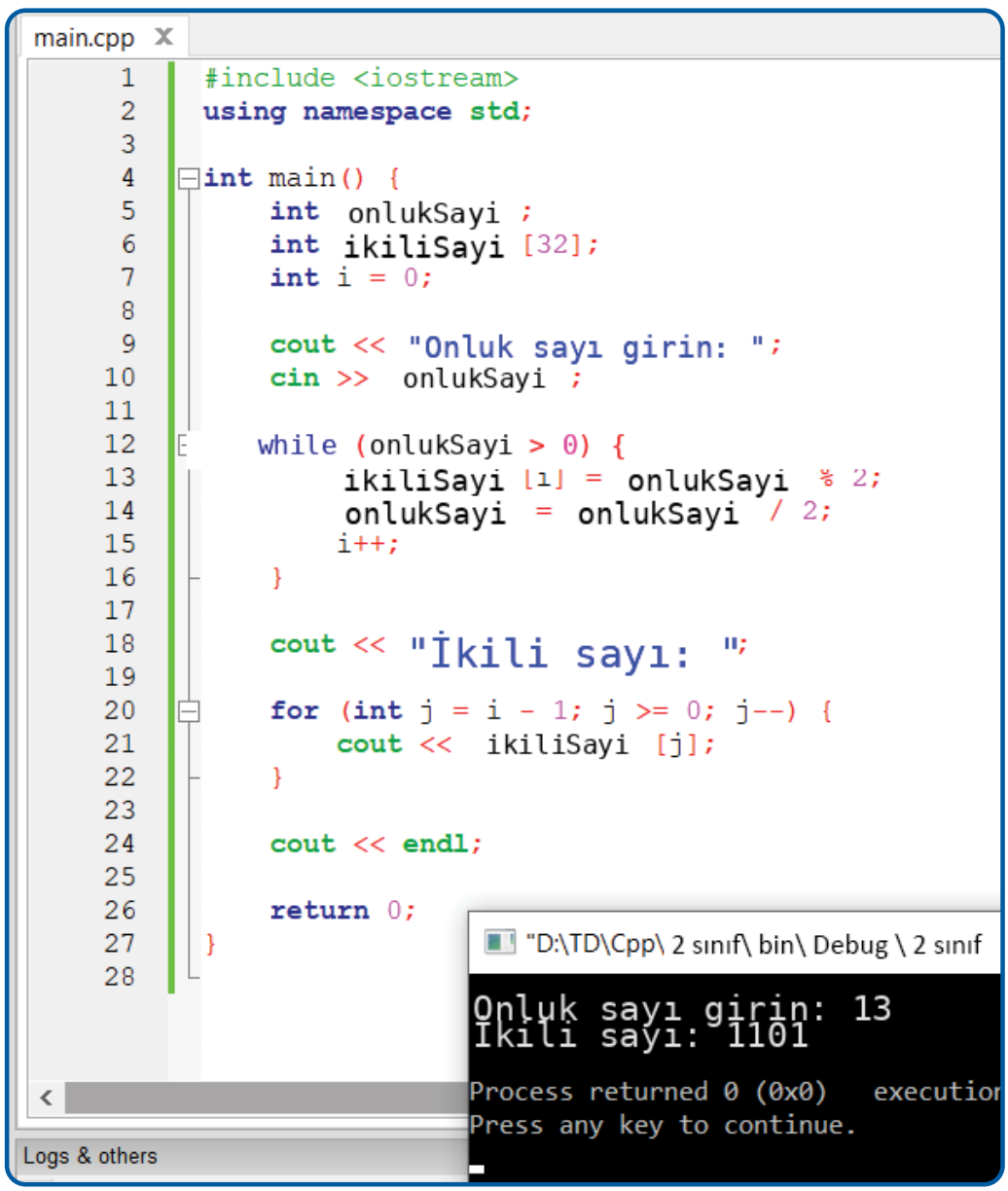
Ondalık sayı sisteminden ikili sayı sistemine dönüşüm, sayının art arda 2'ye bölünmesiyle gerçekleştirilir. Her bölme işleminde kalan yazılır ve kalanlar ters sırayla okunduğunda ikili sayı elde edilir. Örneğin 13_{10} sayısının ikili sisteme dönüştürülmesi aşağıdaki şekilde gerçekleştirilir:

Bölme	Bölüm	Kalan
13 : 2	6	1
6 : 2	3	0
3 : 2	1	1
1 : 2	0	1

Kalanlar aşağıdan yukarıya doğru okunduğunda:
$13_{10} = 1101_2$

Bu işlem, bilişimdeki en önemli dönüşümlerden biridir. Bu tür dönüşümler sayesinde programcılar bilgisayarın bellekte sayıları nasıl işlediğini daha iyi anlayabilir. Programlama dillerinde bu işlemleri otomatik olarak gerçekleştiren algoritmalar sıklıkla kullanılmaktadır.

Örnek 2: Ondalık sayının ikili sayıya dönüştürülmesi



İkili sistemin yanı sıra bilişimde sekizlik ve onaltılık sayı sistemleri de kullanılmaktadır. Sekizlik sistemin tabanı 8'dir ve 0'dan 7'ye kadar olan rakamları kullanır. Onaltılık sistemin tabanı ise 16'dır ve 0'dan 9'a kadar olan rakamların yanı sıra A, B, C, D, E ve F harflerini kullanır. Bu sistemler, uzun ikili sayıların daha kısa ve daha anlaşılır biçimde gösterilmesini sağlar. İkili sistemden sekizlik sisteme dönüşüm yapılırken ikili rakamlar üçlü gruplara, onaltılık sisteme dönüşüm yapılırken ise dörtlü gruplara ayrılır. Gruplama sağdan sola doğru yapılır. En solda bir veya iki rakam kalırsa, bunların sol tarafına sıfır eklenir.

Sekizlik sisteme dönüşüm örneği:

10110110_2

3'erli gruplama:

010 110 110

$010_2 = 2_8$

$110_2 = 6_8$

$110_2 = 6_8$

Dolayısıyla:

$10110110_2 = 266_8$

Onaltılık sisteme dönüşüm örneği:

10110110_2

4'erli gruplama:

1011 0110

$1011_2 = B_{16}$

$0110_2 = 6_{16}$

Demek:

$10110110_2 = B6_{16}$

Onaltılık sistem, programlamada, bellek adreslerinde ve web tasarımında renklerin temsil edilmesinde yaygın olarak kullanılmaktadır. Örneğin #FF0000 rengi, RGB sisteminde kırmızı rengi temsil eder.

Örnek 3: Sekizlik ve onaltılık gösterimin sunulması

main.cpp

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

int main() {
    int sayi;

    cout << "Onluk sayı girin: ";
    cin >> sayi;

    cout << "Sekizlik gösterim: " << oct << sayi << end

    cout << "Onaltılık gösterim: " << hex << sayi

    return 0;
}
```

Farklı sayı sistemleri, bilişim ve bilgisayar mimarisinin önemli bir bölümünü oluşturmaktadır. Modern bilgisayarlar dönüşümleri otomatik olarak gerçekleştirse de programcıların bu sistemlerin nasıl çalıştığını anlaması gerekir. Bu bilgi, belleğin düzenlenmesinin, işlemcinin çalışma biçiminin ve verilerin işlenmesinin daha kolay anlaşılmasını sağlar. Ayrıca sayı sistemleri bilgisayar ağlarında, kriptografide, dijital elektronikte ve yazılım geliştirmede de kullanılmaktadır. Bu nedenle sayı sistemlerinin öğrenilmesi, programlama ve modern bilişim teknolojilerinin daha ileri düzeyde öğrenilmesi için temel oluşturmaktadır.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Bir sayı sisteminin tabanı nedir?
2. İkili sistemde hangi rakamlar kullanılır?
3. İkili sayı sisteminden ondalık sayı sistemine dönüşüm nasıl yapılır?
4. Onaltılık sayı sisteminin tabanı kaçtır?
5. Bilişimde onaltılık gösterim neden kullanılır?

ETKİNLİKLER

1. 10101_2 sayısını elle ondalık sayı sistemine dönüştürünüz.
2. 25_{10} sayısını elle ikili sayı sistemine dönüştürünüz.
3. 11101010_2 sayısını elle sekizlik ve onaltılık sayı sistemine dönüştürünüz.
4. İkili bir sayıyı ondalık sayıya dönüştüren bir program yazınız.
5. Girilen bir sayının sekizlik ve onaltılık gösterimini görüntüleyen bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Sayı sistemi, sayıların yazılma biçimidir.
- İkili sistemin tabanı 2'dir.
- Sekizlik sistemin tabanı 8'dir.
- Onaltılık sistemin tabanı 16'dır.
- İkili sayı sisteminden ondalık sayı sistemine dönüşüm basamak değerleri kullanılarak yapılır.
- Ondalık sayı sisteminden ikili sayı sistemine dönüşüm, sayının art arda 2'ye bölünmesiyle gerçekleştirilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Bilgisayar işlemcilerinde ve bellek sistemlerinde, ikili bilgilerin daha kısa ve daha anlaşılır biçimde gösterilmesini sağladığı için onaltılık değerler sıklıkla kullanılmaktadır. Programlama dillerinde bellek adresleri çoğunlukla onaltılık biçimde gösterilir. Ayrıca onaltılık sayı sistemi, web teknolojilerinde renklerin temsil edilmesinde de kullanılmaktadır, burada her renk, altı onaltılık karakterden oluşan bir kombinasyonla ifade edilir.

2.3

TAM SAYILARIN TEMSİLİ VE VERİ ARALIKLARI

BU KONUYU TAMAMLADIĞINIZDA...

- İkili, ondalık, sekizlik ve onaltılık sayı sistemlerini ayırt etmeyi;
- Sayı sistemleri arasında dönüşümler yapmayı;
- Bir sayı sisteminin tabanının ne olduğunu açıklamayı;
- Basamak değerlerini analiz etmeyi;
- C++'ta temel operatörleri kullanmayı bileceksiniz.

TEKRARLAMA ÇALIŞMALARI

1. Sekizlik sayı sisteminin tabanı kaçtır?
2. 1111_2 sayısını elle ondalık sayı sistemine dönüştürünüz.
3. 45_{10} sayısını elle ikili sayı sistemine dönüştürünüz.
4. 10101111_2 ikili sayısının onaltılık gösterimi nedir?
5. Girilen bir sayının sekizlik gösterimini görüntüleyen bir program yazınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Tam sayıların bilgisayarda nasıl temsil edildiğini açıklamayı;
- Bellekte pozitif ve negatif sayıları ayırt etmeyi;
- Veri aralığının ne olduğunu açıklamayı;
- Taşma (overflow) durumunu analiz etmeyi;
- C++'ta farklı tam sayı veri türlerini kullanmayı.

Günlük iletişimde sayılar çoğunlukla bilgisayarda nasıl saklandıkları düşünülmeden kullanılır. Bununla birlikte bilgisayar sistemlerinin belleği sınırlıdır. Bu nedenle her sayı belirli bir biçimde ve sınırlı sayıda bit kullanılarak temsil edilmelidir. Sayıların bellekte temsil edilme biçimi, hesaplamaların doğruluğu, işlem hızı ve bilgi depolama kapasitesi üzerinde büyük etkiye sahiptir.

Programlama dillerinde tam sayıların temsil edilmesi için farklı veri türleri bulunmaktadır. Her veri türü bellekte farklı miktarda yer kaplar ve farklı bir değer aralığının temsil edilmesine olanak sağlar. Program, izin verilen aralıktan daha büyük bir sayıyı saklamaya çalışırsa overflow (taşma) adı verilen bir hata meydana gelebilir. Bu nedenle programcıların programlarında kullanacakları veri türlerini dikkatli bir şekilde seçmeleri gerekir.

Bilgisayarlar tüm sayıları ikili biçimde temsil eder. Bellek sınırlı olduğundan her sayı için belirli sayıda bit ayrılır. Örneğin bir sayı için 8 bit kullanılıyorsa, sınırlı sayıda farklı değer temsil edilebilir. Pozitif sayılarda her bit, 2'nin kuvvetleri aracılığıyla sayının oluşturulmasına katkıda bulunur. Örneğin 13 sayısı ikili biçimde 00001101_2 olarak temsil edilir. Bellekte her bitin belirli bir yeri ve anlamı vardır. Kullanılan bit sayısı arttıkça temsil edilebilecek sayı aralığı da genişler. Programlama dillerinde çoğunlukla short, int, long long ve diğer veri türleri kullanılır ve her veri türü farklı miktarda bellek kullanır. Bu nedenle veri türünün seçimi, programın kullandığı bellek miktarını doğrudan etkiler. Modern bilgisayar sistemlerinde veri türlerinin doğru seçilmesi, programların optimizasyonunun önemli bir bölümünü oluşturur.

Örnek 1: Veri türlerinin boyutlarının gösterimi

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      cout << "int: " << sizeof(int) << " bayt" << endl;
7      cout << "short: " << sizeof(short) << " bayt" << endl;
8      cout << "long long: " << sizeof(long long) << " bayt" << endl;
9
10     return 0;
11 }
12
```

```
"D:\TD\Cpp\ 2 sınıf\ bin\ Debug \ 2 sınıf .exe"
int: 4 bayt
short: 2 bayt
long long: 8 bayt

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

Pozitif sayıların yanı sıra bilgisayarlarda negatif sayıların da temsil edilmesi gerekir. Bunun için two's complement (ikiye tümleyen) adı verilen özel bir gösterim yöntemi kullanılır. Bu gösterim yönteminde en soldaki bit genellikle sayının işaretini belirtir. Bu sistem sayesinde işlemci, hem pozitif hem de negatif sayılarla toplama ve çıkarma işlemlerini kolaylıkla gerçekleştirebilir. Örneğin sayıların temsil edilmesi için 8 bit kullanılıyorsa –128 ile 127 arasındaki değerler temsil edilebilir. Bu, bellekte saklanabilecek en büyük ve en küçük sayıların sınırlı olduğu anlamına gelir. Program izin verilen aralığın dışında bir sayıyı saklamaya çalışırsa sonuç hatalı olabilir. Bu tür sınırlamalar küçük programlarda çoğu zaman fark edilmese de büyük sistemlerde ciddi sorunlara yol açabilir. Bu nedenle veri aralığını anlamak, programlamanın önemli bir bölümünü oluşturmaktadır.

Örnek 2: Minimum ve maksimum değerin gösterimi

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

int main() {

    cout << "int için minimum değer: " << INT_MIN << endl;
    cout << "int için maksimum değer: " << INT_MAX << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
int için minimum değer: -2147483648
int için maksimum değer: 2147483647
```

Veri aralığı, belirli bir veri türüyle temsil edilebilen değerler aralığını ifade eder. Kullanılan bit sayısı arttıkça veri aralığı da genişler. Örneğin short veri türü long long veri türünden daha az bellek kaplar, ancak daha küçük sayıları saklayabilir. Programlarda çoğu zaman gerekli olacak aralığın dikkatli bir şekilde belirlenmesi gerekir. Büyük hesaplamalarda küçük aralığa sahip bir veri türü kullanılırsa overflow (taşma) meydana gelebilir. Overflow, sonucun saklanabilecek maksimum değeri aşması durumudur. Böyle durumlarda program beklenmeyen veya hatalı bir sonuç gösterebilir. Bu tür hatalar özellikle finansal, bilimsel ve güvenlik sistemlerinde tehlikelidir. Bu nedenle programcıların veri türlerinin sınırlamalarını anlamaları ve bunları programın ihtiyaçlarına uygun şekilde seçmeleri gerekir.

Örnek 3: Tam sayıda Overflow

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

int main() {

    int sayi = INT_MAX;

    cout << "Artırmadan önce: " << sayi << endl;

    sayi++;

    cout << "Artırdıktan sonra: " << sayi << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
Artırmadan önce: 2147483647
Artırdıktan sonra: -2147483648
```

Modern programlama dilleri, programların daha verimli ve daha doğru olmasını sağlamak amacıyla farklı veri türlerinin kullanılmasına olanak tanır. Bazı durumlarda çok büyük sayıların saklanması gerekirken, bazı durumlarda bellekte yer tasarrufu yapmak önemlidir. Bu nedenle programcılar, belirli probleme göre farklı veri türleri

seçerler. Ayrıca sayıların aralığının doğru anlaşılması, hataların daha kolay tespit edilmesini ve programların daha iyi analiz edilmesini sağlar. Sayıların bellekte nasıl temsil edildiğinin bilinmesi, algoritmaların, veri yapılarının ve program optimizasyonunun daha ileri düzeyde öğrenilmesi için temel oluşturmaktadır.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Bilgisayarlar sayıları neden ikili biçimde temsil eder?
2. Veri aralığı nedir?
3. Overflow (taşma) nedir?
4. Hangi veri türü daha büyük değerleri saklayabilir: int mi yoksa long long mı?
5. Veri türünün doğru seçilmesi neden önemlidir?

ETKİNLİKLER

1. char, int ve double veri tiplerinin kapladığı alanı gösterecek bir program yazınız.
2. int veri tipinin minimum ve maksimum değerlerini gösterecek bir program yazınız.
3. short veri tipinde taşma (overflow) durumunu gösterecek bir program yazınız.
4. İki büyük sayı girecek ve bunları long long veri tipini kullanarak toplayacak bir program yazınız.
5. unsigned int veri tipinin hangi değer aralığına sahip olduğunu araştırınız ve int veri tipi ile karşılaştırınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Bilgisayarlar sayıları ikili biçimde temsil eder.
- Her veri türü farklı sayıda bayt kullanır.
- Veri aralığı, saklanabilecek en küçük ve en büyük değeri ifade eder.
- Overflow, bir değerin izin verilen aralığı aşması durumunda meydana gelir.
- Veri türünün doğru seçilmesi, programın doğruluğu ve verimliliği açısından önemlidir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern programlama dillerinde, çok geniş değer aralığına sahip veri türleri de bulunmaktadır. Bu veri türleri bilimsel simülasyonlarda, kriptografide ve büyük veri tabanlarının işlenmesinde kullanılmaktadır. Bazı programlama kütüphaneleri yüzlerce veya binlerce basamak içeren sayılarla çalışmaya olanak sağlamaktadır. Bu tür sayılar standart veri türleriyle doğrudan temsil edilemez, bu nedenle bunların işlenmesi için özel algoritmalar ve veri yapıları kullanılmaktadır.

2.4

ALGORİTMALARIN OPTİMIZASYONU VE VERİMLİLİĞİ

BU KONUYU TAMAMLADIĞINIZDA...

- Algoritmanın ne olduğunu açıklamayı;
- If ve switch yapılarını kullanmayı;
- For ve while döngülerini kullanmayı;
- Zaman karmaşıklığını analiz etmeyi;
- C++'ta metinsel problemleri çözmeyi bileceksiniz.

TEKRARLAMA ÇALIŞMALARI

1. Algoritma nedir?
2. 10 kez tekrarlanan bir for döngüsü örneği yazınız.
3. for ve while döngüleri arasındaki fark nedir?
4. Zaman karmaşıklığı nedir?
5. Girilen bir sayının karesini hesaplayan bir program yazınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Algoritmaların verimliliğini analiz etmeyi;
- Aynı problem için farklı çözümleri karşılaştırmayı;
- Gereksiz işlemleri fark etmeyi;
- Algoritmaları optimize etmeyi;
- Veri büyüklüğünün programın çalışma hızı üzerindeki etkisini analiz etmeyi.

Programlamada aynı problemi çözmek için çoğu zaman birden fazla farklı yöntem bulunmaktadır. Farklı çözümler aynı sonucu verebilse de her zaman aynı hızda çalışmaz ve aynı miktarda bellek kullanmaz. Küçük programlarda bu tür farklılıklar çoğu zaman fark edilmez, ancak büyük miktarda verinin işlenmesi sırasında bu farklılıklar sistemin çalışmasını önemli ölçüde etkileyebilir.

Bu nedenle modern programlama yalnızca programın doğruluğuna değil, aynı zamanda verimliliğine de odaklanmaktadır. Programcılar sürekli olarak algoritmanın kaç işlem gerçekleştirdiğini, ne kadar bellek kullandığını

ve problemi çözmenin daha iyi bir yolunun bulunup bulunmadığını analiz eder. Bu süreç algoritma optimizasyonu olarak adlandırılır ve modern yazılım geliştirmenin önemli bir bölümünü oluşturur.

Bir algoritmanın verimliliği çoğunlukla problemi çözerken gerçekleştirilen işlem sayısı üzerinden analiz edilir. Gereken işlem sayısı arttıkça programın çalışması için gereken süre de artar. Örneğin, 1'den n'e kadar olan sayıların toplamının hesaplanması gerekiyorsa, tüm sayıları tek tek toplayan bir döngü kullanılabilir. Bu yaklaşım doğrudur, ancak n'nin çok büyük değerlerinde çok fazla işlem gerçekleştirilir. Öte yandan aynı problem, sonucu doğrudan veren matematiksel bir formülle çözülebilir. Bu durumda algoritma çok daha az işlem gerçekleştirir ve daha hızlı çalışır. Bu nedenle programcılar gereksiz hesaplamaları azaltmanın yollarını sürekli olarak ararlar. Bu düşünme biçimi, algoritma optimizasyonunun temelini oluşturur ve modern programlamanın en önemli bölümlerinden biridir.

Örnek 1: İki algoritmanın karşılaştırılması

main.cpp

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

int main() {

    int n;
    cin >> n;

    int toplam1 = 0;

    for (int i = 1; i <= n; i++) {
        toplam1 += i;
    }

    int toplam2 = n * (n + 1) / 2;

    cout << "Döngü ile: " << toplam1 << endl;
    cout << "Formül ile: " << toplam2 << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godin...
500
Döngü ile: 125250
Formül ile: 125250
Process returned 0 (0x0)   execution time : 3.100 s
Press any key to continue.
```

Algoritmaların analizinde, veri miktarı arttığında programın hızının nasıl değiştiğinin incelenmesi büyük önem taşır. 10 eleman için hızlı çalışan bir algoritma, bir milyon elemanı işlerken çok yavaş hâle gelebilir. Bu nedenle programcılar yalnızca mevcut çalışma hızını değil, algoritmanın büyük miktarda giriş verisiyle karşılaştığında nasıl davrandığını da analiz eder. Bilişimde karmaşıklığı tanımlamak için sıklıkla O(n), $O(n^2)$ ve O(log n) gibi gösterimler kullanılır. $O(n^2)$ karmaşıklığına sahip bir algoritma, büyük veri kümelerinde genellikle O(n) karmaşıklığına sahip bir algoritmadan çok daha yavaş hâle gelir. Örneğin iç içe döngüler, işlem sayısının önemli ölçüde artmasına neden olabilir. Bu nedenle programcılar gereksiz tekrarların önlenip önlenemeyeceğini veya daha iyi bir çözüm bulunup bulunmadığını dikkatle analiz eder. Büyük bilgisayar sistemlerinde doğru optimizasyon, programın çalışma performansını önemli ölçüde iyileştirebilir.

Örnek 2: Verimsiz ve daha verimli arama

*main.cpp

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

int main()
{
    int dizi[5] = {2, 4, 6, 8, 10};
    int sayi;
    bool  bulundu  = false;
    cin >> sayi;
    for (int i = 0; i < 5; i++) {

        if (dizi[i] == sayi) {
            bulundu  = true;
            break;
        }
    }
    if (bulundu) {
        cout << "Sayı bulundu. " << endl;
    }
    else {
        cout << "Sayı bulunamadı. " << endl;
    }
    return 0;
}
```

Önceki örnekte, sayı bulunduktan hemen sonra döngüyü sonlandırmak için break komutu kullanılmıştır. break kullanılmadığında döngü, gereksiz elemanları da kontrol etmeye devam eder. Beş elemanlı bir dizide fark küçük olsa da büyük dizilerde bu durum programın çalışma hızını önemli ölçüde etkileyebilir.

Bu tür küçük iyileştirmeler, profesyonel programlamada çoğu zaman büyük önem taşır. Programcılar işlem sayısını nasıl azaltabileceklerini ve belleği nasıl daha verimli kullanabileceklerini sürekli olarak analiz eder. Bu süreç yalnızca "daha hızlı bir program" anlamına gelmez, aynı zamanda kodun daha iyi düzenlenmesini ve sistemin daha kolay bakımının yapılmasını sağlar.

Örnek 3: İç içe döngülerin etkisi

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8      for (int i = 1; i <= n; i++) {
9          for (int j = 1; j <= n; j++) {
10             cout << "* ";
11         }
12         cout << endl;
13     }
14     return 0;
15 }
16
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.ex...
5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

Process returned 0 (0x0)   execution time : 1.186 s
Press any key to continue.
```

Önceki örnekte iç içe döngüler kullanılmıştır. n değeri 5 olduğunda 25 tekrar gerçekleştirilir. n değeri 1000 olduğunda ise bir milyon tekrar gerçekleştirilir. Bu durum, giriş verilerinin büyüklüğü arttığında algoritmanın çalışma hızının nasıl önemli ölçüde değişebileceğini göstermektedir. Bu nedenle programcılar iç içe döngüler kullanırken dikkatli düşünmeli ve daha verimli bir çözüm olup olmadığını analiz etmelidir. Çok büyük miktarda veriyi işleyen modern sistemlerde optimizasyon, programların başarılı bir şekilde çalışmasını sağlayan en önemli faktörlerden biridir.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Algoritma optimizasyonu nedir?
2. İki algoritmanın çalışma hızları neden farklı olabilir?
3. İç içe döngüler karmaşıklığı nasıl etkiler?
4. Gereksiz işlemlerden kaçınmak neden önemlidir?
5. Giriş verilerinin büyüklüğü programın çalışma hızını nasıl etkiler?

ETKİNLİKLER

1. 1 ile n arasındaki 7'ye bölünebilen sayıların kaç tane olduğunu hesaplayan bir program yazınız.
2. n adet sayı girişi alan ve en büyük sayıyı bulan bir program yazınız.
3. İç içe döngüler kullanarak # karakterlerinden bir dikdörtgen yazdıran bir program yazınız.
4. Programın 1000 elemanlı bir dizinin her elemanını kontrol etmesi durumunda kaç kontrol gerçekleştirileceğini analiz ediniz.
5. 1'den n'e kadar olan sayıların toplamını hesaplamak için iki farklı çözüm yazınız ve hangi çözümün daha verimli olduğunu karşılaştırınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Farklı algoritmalar aynı problemi farklı verimlilik düzeyleriyle çözebilir.
- Optimizasyon, işlem sayısını ve bellek tüketimini azaltmayı amaçlar.
- İç içe döngüler karmaşıklığı önemli ölçüde artırır.
- break, gereksiz kontrollerin azaltılmasına yardımcı olabilir.
- Algoritmaların analizi, büyük miktarda veriyle çalışırken önemlidir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern internet hizmetlerinde ve yapay zekâ sistemlerinde her saniye milyonlarca veri işlenmektedir. Algoritmalar yeterince optimize edilmezse sunucular çok daha yavaş çalışır ve çok büyük miktarda kaynak tüketir. Bu nedenle profesyonel programcıların çalışmalarının önemli bir bölümü algoritmaları analiz etmek ve geliştirmekten oluşur. Bazı durumlarda bir algoritmanın optimize edilmesi, programı yüzlerce hatta binlerce kat daha hızlı çalıştırabilir.

2.5

KOŞULLARIN VE DÖNGÜLERİN BİRLEŞTİRİLMESİ

BU KONUYU TAMAMLADIĞINIZDA...

- If ve switch yapılarını kullanmayı;
- For ve while döngülerini kullanmayı;
- Algoritmaların verimliliğini analiz etmeyi;
- İç içe döngüleri kullanmayı;
- Farklı programlama çözümlerini karşılaştırmayı bileceksiniz.

TEKRARLAMA ÇALIŞMALARI

1. Algoritma optimizasyonu nedir?
2. break verimliliği neden artırabilir?
3. İki iç içe döngü kullanıldığında ne olur?
4. 1'den n'e kadar olan sayıların toplamını hesaplayan bir program yazınız.
5. Büyük verilerin neden daha verimli algoritmalar gerektirdiğini açıklayınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Koşulları ve döngüleri aynı program içinde birleştirmeyi;
- Daha karmaşık metinsel problemleri çözmeyi;
- Döngüler içinde iç içe koşulları kullanmayı;
- Tekrar ve seçim mantığını analiz etmeyi;
- Daha düzenli programlama çözümleri oluşturmayı.

Programlamada birçok problem yalnızca koşullar veya yalnızca döngüler kullanılarak çözülemez. Gerçek programlarda eksiksiz bir çözüm elde etmek için çoğunlukla farklı programlama yapılarının bir arada kullanılması gerekir. Örneğin bir program, verileri birden fazla kez alabilir ve her veri girişi sırasında belirli bir koşulun sağlanıp sağlanmadığını kontrol edebilir. Bu tür durumlar, döngülerin ve koşulların eş zamanlı olarak kullanılmasını gerektirir.

Programlama yapılarının birleştirilmesi, daha karmaşık algoritmaların geliştirilmesi yolunda önemli bir adımdır. Bu yaklaşımla programlar daha esnek hâle gelir ve farklı durumları işleyebilir. Bu nedenle programcıların koşulun

nereye yerleştirilmesi gerektiğini, işlemin ne zaman tekrarlanacağını ve programın mantığının nasıl doğru şekilde düzenleneceğini dikkatle analiz etmeleri gerekir.

Döngüler belirli komutların tekrarlanmasını, koşullar ise karar verilmesini sağlar. Bu yapılar birleştirildiğinde program aynı anda işlemleri tekrarlayabilir ve farklı koşulları kontrol edebilir. Örneğin, 1 ile n arasındaki yalnızca çift sayıların sayılması gerekiyorsa döngü tüm sayılar üzerinde ilerler ve koşul her sayının çift olup olmadığını kontrol eder. Böylece koşul, döngünün içerisinde gerçekleştirilir. Bu çalışma biçimi programlamada en sık kullanılan tekniklerden biridir. Modern programlarda çoğunlukla çok miktarda veri işlenir ve her veri için farklı kontroller gerçekleştirilir. Bu nedenle koşulların ve döngülerin doğru şekilde birleştirilmesi, daha karmaşık problemlerin çözülmesi ve verimli algoritmaların oluşturulması için temel oluşturmaktadır.

Örnek 1: Çift sayıların sayılması

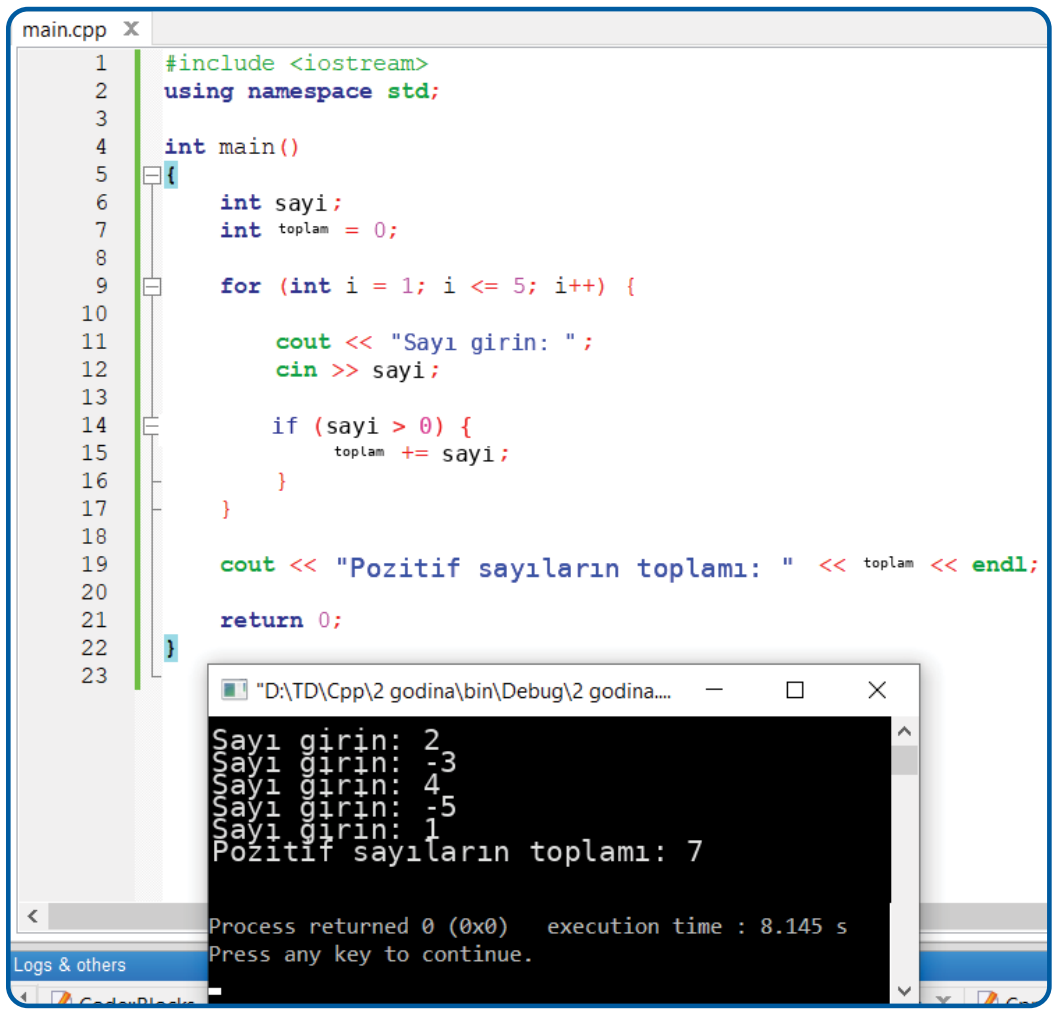
main.cpp

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

int main()
{
    int n;
    int sayac = 0;

    cout << "Sayı girin: ";
    cin >> n;

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

        if (i % 2 == 0) {
            sayac +;
        }
    }

    cout << "Çift sayıların sayısı: " << sayac << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina....
Sayı girin: 10
Çift sayıların sayısı: 5

Process returned 0 (0x0)   execution time : 1.201 s
Press any key to continue.
```

Birçok problemde koşulların döngüler içerisinde veya döngülerin koşullar içerisinde kullanılması gerekir. Bu tür yapılar daha karmaşık problemlerin çözülmesini sağlar. Örneğin bir sınıftaki öğrenciler analiz edilirken, belirli bir değer girilene kadar notların alınması ve ardından her notun başarılı veya başarısız olup olmadığının kontrol edilmesi gerekebilir. Bu tür algoritmalar mantığın dikkatli bir şekilde düzenlenmesini gerektirir. Yanlış yerleştirilen bir koşul sonsuz döngüye veya hatalı bir sonuca neden olabilir. Bu nedenle programcılar genellikle adım adım ilerler: önce tekrarlama işlemini analiz eder, ardından gerekli koşulları eklerler. Bu yaklaşım programın daha kolay test edilmesini ve kodun daha düzenli olmasını sağlar.

Örnek 2: Pozitif sayıların toplamı



Koşulların ve döngülerin birleştirilmesi, metinsel problemlerin çözümünde de sıklıkla kullanılır. Bu tür problemlerde farklı durumların analiz edilmesi ve uygun işlemin seçilmesi gerekir. Örneğin program bir sayının asal olup olmadığını, başka bir sayıya bölünüp bölünmediğini veya aynı anda birden fazla koşulu sağlayıp sağlamadığını kontrol edebilir. Bu durumlarda koşullar oldukça karmaşık olabilir ve birden fazla mantıksal operatör içerebilir. Ayrıca döngüler, kontrollerin birden fazla veri üzerinde tekrarlanması için kullanılır. Bu nedenle bu tür programlar algoritmik düşünme ve mantıksal analiz becerilerinin geliştirilmesi için iyi bir temel oluşturur. Farklı yapılar ne kadar çok birleştirilirse programlar o kadar güçlü hâle gelir ve gerçek problemlerin çözülmesinde o kadar yetkin olur.

Örnek 3: Asal sayı kontrolü

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int sayi;
7      bool asal = true;
8      cout << "Sayı girin: ";
9      cin >> sayi;
10     if (sayi <= 1)
11     {
12         asal = false;
13     }
14     for (int i = 2; i < sayi; i++)
15     {
16         if (sayi % i == 0)
17         {
18             asal = false;
19         }
20     }
21     if (asal)
22     {
23         cout << "Sayı asal. " << endl;
24     }
25     else
26     {
27         cout << "Sayı asal değil. " << endl;
28     }
29     return 0;
30 }
31
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.
Sayı girin: 9
Sayı asal değil.

Process returned 0 (0x0)   execution
Press any key to continue.
```

Önceki örnekte döngü, sayının olası bölenlerini sırayla kontrol ederken koşul, sayının kalansız bölünüp bölünmediğini kontrol eder. Bu yapıların birleştirilmesi sayesinde sayı doğru şekilde analiz edilir ve doğru sonuç elde edilir. Bu yaklaşım, veri analizi, koşul kontrolü ve işlemlerin tekrarlanmasını gerektiren birçok algoritmada kullanılır. Bu nedenle koşulların ve döngülerin birleştirilmesi, daha karmaşık programlama çözümlerinin geliştirilmesindeki en önemli adımlardan biridir.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Programlarda koşullar ve döngüler neden birlikte kullanılır?
2. İç içe koşul neyi ifade eder?
3. Bir döngü içerisinde koşulun görevi nedir?
4. Program mantığının doğru şekilde düzenlenmesi neden önemlidir?
5. Birden fazla verinin işlenmesinde döngüler nasıl yardımcı olur?

ETKİNLİKLER

1. 1'den n'e kadar olan tek sayıların sayısını bulan bir program yazınız.
2. 10 sayı alıp yalnızca negatif sayıların toplamını hesaplayan bir program yazınız.
3. Girilen sayının mükemmel sayı olup olmadığını kontrol eden bir program yazınız.
4. Belirli bir aralıkta 3 ve 5'e bölünebilen tüm sayıları yazdıran bir program yazınız.
5. Girilen sayıların kaç tanesinin pozitif, kaç tanesinin negatif olduğunu bulan bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Döngüler ve koşullar çoğunlukla birlikte kullanılır.
- Koşullar farklı durumların kontrol edilmesini sağlar.
- Döngüler işlemlerin tekrarlanmasını sağlar.
- Programlama yapılarını birleştirmek daha karmaşık problemlerin çözülmesini mümkün kılar.
- Program mantığının doğru şekilde düzenlenmesi programlamada çok önemlidir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilgisayar sistemlerinde çok sayıda algoritma, koşulların ve tekrarların birleştirilmesine dayanır. İnternet arama motorları, öneri sistemleri ve yapay zekâ sistemleri, farklı koşullar ve döngüler kullanarak sürekli olarak büyük miktarda veriyi işler. Bazı programlarda saniyede milyarlarca kontrol gerçekleştirilebilir. Bu nedenle koşulların ve döngülerin doğru şekilde düzenlenmesi, modern sistemlerin hızı ve verimliliği üzerinde büyük bir etkiye sahiptir.

2.6

KARMAŞIK METİNSEL PROBLEMLERİN ÇÖZÜMÜ

BU KONUYU TAMAMLADIĞINIZDA...

- Koşulları ve döngüleri birleştirmeyi;
- İç içe döngüler kullanmayı;
- Algoritmaların verimliliğini analiz etmeyi;
- Basit programlama çözümlerini optimize etmeyi;
- Program mantığını düzenlemeyi bileceksiniz.

TEKRARLAMA SORULARI

1. Bir algoritmanın optimizasyonu neyi ifade eder?
2. İç içe döngüler programı neden önemli ölçüde yavaşlatabilir?
3. 1'den n'e kadar 5'e bölünebilen sayıların kaç tane olduğunu bulan bir program yazınız.
4. Bir algoritmadaki gereksiz işlem neyi ifade eder?
5. Farklı çözümlerin neden farklı verimliliklere sahip olabileceğini açıklayınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Karmaşık metinsel problemleri analiz etmeyi;
- Gerekli verileri ve koşulları belirlemeyi;
- Birden fazla adımdan oluşan algoritmalar oluşturmayı;
- Birden fazla programlama tekniğini birleştirmeyi;
- Programlama çözümlerini test etmeyi ve geliştirmeyi.

Programlamada birçok problem, tek bir basit hesaplamayla veya yalnızca tek bir programlama yapısı kullanılarak çözülemez. Çoğu zaman aynı anda birden fazla koşulun analiz edilmesi, çok sayıda verinin işlenmesi ve tek bir programda birden fazla adımın düzenlenmesi gerekir. Bu tür problemler, daha ayrıntılı bir analiz ve algoritmanın daha dikkatli düzenlenmesini gerektirdiği için karmaşık metinsel problemler olarak adlandırılır.

Karmaşık problemler çözülürken problemin önce dikkatlice okunması ve anahtar bilgilerin belirlenmesi çok önemlidir. Giriş verilerinin, kontrol edilmesi gereken koşulların ve elde edilmesi gereken sonucun belirlenmesi gerekir. Daha sonra problem, bir programda daha kolay gerçekleştirilebilecek küçük adımlara kademeli olarak bölünür. Bu yaklaşım, hataların daha kolay tespit edilmesini ve daha düzenli çözümler oluşturulmasını sağlar.

Karmaşık bir metinsel problem analiz edilirken ilk adım, problemin doğru şekilde anlaşılmasıdır. Birçok durumda problem, aynı anda sağlanması gereken birden fazla koşul içerir. Programcı tüm gereksinimleri dikkatlice analiz etmezse program yanlış sonuçlar verebilir. Bu nedenle, problemin birden fazla alt göreve bölündüğü kademeli bir yaklaşım sıklıkla kullanılır. Örneğin, öğrencilerin sınav sonuçlarının analiz edilmesi gerekiyorsa önce başarılı olan öğrenciler sayılabilir, ardından ortalama hesaplanabilir ve son olarak istatistikler görüntülenebilir. Bu tür bir bölümlendirmeyle algoritma daha anlaşılır ve gerçekleştirilmesi daha kolay hâle gelir. Modern programlamada bu düşünme biçimi, karmaşık problemlerin çözülmesi ve daha büyük yazılım sistemlerinin geliştirilmesi için temel oluşturur.

Örnek 1: Notların analizi

main.cpp

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

int main()
{
    int n;
    int  not  ;
    int basarili = 0;

    cout << "Kaç tane not girilecek? " ;
    cin >> n;

    for (int i = 1; i <= n; i++)
        {
        cin >>  not  ;
        if (not >= 2)
        {
            basarili++;
        }
    }
    cout << "Başarılı öğrencilerin sayısı: "  << basarili << endl;
    return 0;
}
```

```
"D:\TD\Cpp\ 2 sınıf\bin\Debug\2 sınıf.exe
Kaç tane not girilecek? 5
2
1
3
5
4
Başarılı öğrencilerin sayısı: 4
```

Karmaşık problemler çoğu zaman aynı program içinde birden fazla programlama tekniğinin birleştirilmesini gerektirir. Tek bir problemde aynı anda döngüler, koşullar, sayaçlar ve farklı hesaplamalar kullanılabilir. Bu nedenle programın iyi organize edilmesi ve mantıksal olarak bölümlere ayrılması çok önemlidir. Programcılar, programdaki farklı durumları izlemek için sıklıkla ek değişkenler kullanır. Örneğin sayılar analiz edilirken aynı anda en büyük değer, en küçük değer ve pozitif sayıların adedi izlenebilir. Bu tür problemler, programlamaya başlamadan önce algoritmanın dikkatli bir şekilde planlanmasını gerektirir. Mantık doğru şekilde düzenlenmezse programın anlaşılması ve bakımı zorlaşabilir. Bu nedenle çözümün kademeli olarak oluşturulması, programlamanın çok önemli bir parçasıdır.

Örnek 2: En büyük ve en küçük değerin bulunması

main.cpp

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

int main()
{
    int n;
    int sayi;
    cout << "Kaç sayı girilecek? "  ;
    cin >> n;
    cin >> sayi;
    int maksimum = sayi;
    int minimum = sayi;
    for (int i = 2; i <= n; i++)
        {
        cin >> sayi;
        if (sayi > maksimum)
        {
            maksimum = sayi;
        }
        if (sayi < minimum)
        {
            minimum = sayi;
        }
    }
    cout << "Maksimum: " << maksimum << endl;
    cout << "Minimum: " << minimum << endl;
    return 0;
}
```

```
"D:\TD\Cpp\ 2 sınıf\bin\Debug\2 sınıf.exe
Kaç sayı girilecek? 5
24 -2 8 102
Maksimum: 102
Minimum: -2
```

Daha karmaşık problemler çözülürken algoritmanın farklı giriş verileriyle test edilmesi çok önemlidir. Bazı programlar küçük değerler için doğru çalışır, ancak daha büyük veya beklenmeyen verilerde hata verir. Bu nedenle programcılar, programın kararlı ve doğru olup olmadığını kontrol etmek için sürekli olarak farklı durumları test eder. Sıfır girilmesi, negatif sayılar veya çok büyük değerler gibi sınır durumlarının analiz edilmesi özellikle önemlidir. Bu tür testlerle mantıksal hatalar ve algoritmadaki sorunlar daha kolay tespit edilir. Profesyonel programlamada test etme, her programın geliştirilmesinin zorunlu bir parçasıdır.

Örnek 3: Pozitif ve negatif sayıların analizi

main.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int n;
7       int sayi;
8       int pozitifler = 0;
9       int negatifler = 0;
10      cout << "Kaç sayı girilecek? "  ;
11      cin >> n;
12      for (int i = 1; i <= n; i++)
13          {
14          cin >> sayi;
15          if (sayi >= 0)
16          {
17              pozitifler ++;
18          }
19          else
20          {
21              negatifler ++;
22          }
23      }
24      cout << "Pozitif: " << pozitifler << endl;
25      cout << "Negatif: " << negatifler << endl;
26      return 0;
27  }
28
```

```
"D:\TD\Cpp\. 2 sınıf\bin\Debug\2 sınıf.exe
-12 -34-5
Pozitif: 2
Negatif: 3
```

Karmaşık metinsel problemler, algoritmik düşüncenin geliştirilmesinin önemli bir parçasıdır. Bu problemler çözülürken yalnızca programlama dilinin sözdiziminin bilinmesi yeterli değildir; problemin dikkatlice analiz edilmesi ve adımların doğru şekilde düzenlenmesi de gerekir. Problemler karmaşıklaştıkça mantıksal düşünme, analiz etme ve çözümleri planlama becerisi o kadar gelişir. Bu nedenle bu tür problemler, daha ileri düzey programlama becerilerinin geliştirilmesi yolunda önemli bir adımdır.

TEKRARLAMA SORULARI

SORULAR

1. Karmaşık metinsel problem nedir?
2. Problemi daha küçük adımlara bölmek neden önemlidir?
3. Programların farklı verilerle test edilmesi neden gereklidir?
4. Sayaçların programlarda hangi görevi vardır?
5. Algoritmanın düzenli bir şekilde oluşturulması neden önemlidir?

ETKİNLİKLER

1. n adet sayı girecek ve bu sayıların ortalamasını hesaplayacak bir program yazınız.
2. Girilen sayıların kaç tanesinin 3'e bölündüğünü sayacak bir program yazınız.
3. Girilen sayılar arasındaki ikinci en büyük sayıyı bulacak bir program yazınız.
4. Girilen notlardan kaç tanesinin pekiyi olduğunu kontrol edecek bir program yazınız.
5. Girilen sayıların kaç tanesinin çift, kaç tanesinin tek olduğunu analiz edecek bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Karmaşık problemler dikkatli bir analiz gerektirir.
- Problem çoğunlukla birden fazla küçük adıma ayrılır.
- Karmaşık problemlerde birden fazla programlama tekniği bir arada kullanılır.
- Test etme, program geliştirme sürecinin önemli bir parçasıdır.
- Algoritmanın iyi organize edilmesi, daha anlaşılır ve daha verimli çözümler oluşturulmasını sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern yazılım sistemlerinde programcılar nadiren yalnızca basit problemlerle uğraşırlar. Günümüzde birçok uygulama çok büyük miktarda veriyi işler ve aynı anda çok sayıda koşulu kontrol eder. Bankacılık sistemleri, çevrim içi mağazalar ve sosyal medya platformları, bilgileri sürekli analiz eden ve kararlar veren karmaşık algoritmalar kullanır. Bu nedenle karmaşık metinsel problemleri çözebilme becerisi, modern programlamadaki en önemli becerilerden biri olarak kabul edilir.

2.7

MODÜLER PROGRAMLAMA VE "BÖL VE YÖNET" KAVRAMI

BU KONUYU TAMAMLADIĞINIZDA...

- C++ dilinde koşullu yapıları kullanmayı;
- İşlemleri tekrarlamak için döngüleri kullanmayı;
- Koşulları ve döngüleri aynı program içerisinde birleştirmeyi;
- Algoritmaların verimliliğini analiz etmeyi;
- Daha karmaşık programlama problemlerini çözmeyi bileceksiniz.

TEKRARLAMA SORULARI

1. Algoritma optimizasyonu nedir?
2. Programdaki gereksiz işlem sayısının azaltılması neden önemlidir?
3. 1'den n'e kadar 4'e bölünebilen sayıların kaç tane olduğunu bulan bir program yazınız.
4. İç içe döngü nedir?
5. Büyük programların küçük programlara göre bakımının neden daha zor olduğunu açıklayınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Modüler programlamanın ne olduğunu açıklamayı;
- "Böl ve yönet" ilkesini uygulamayı;
- Karmaşık problemleri analiz ederek daha küçük görevlere ayırmayı;
- Programlama çözümlerini mantıksal bölümlere ayırarak düzenlemeyi;
- Fonksiyonların kullanılmasına yönelik programlar hazırlamayı.

Programların karmaşıklığı arttıkça kodun daha iyi organize edilmesine ve problemlerin çözümünde daha verimli yaklaşımların kullanılmasına ihtiyaç duyulur. Küçük programlar nispeten kolay bir şekilde tek bir bütün olarak yazılabilir, ancak daha büyük programlama çözümlerinde bu yaklaşım birçok güçlük oluşturur. Tüm komutların tek bir yerde toplanması, programın okunmasını, anlaşılmasını, test edilmesini ve bakımını zorlaştırır. Bu nedenle

modern programlamada karmaşık problemlerin daha düzenli bir şekilde organize edilmesini sağlayan teknikler kullanılmaktadır.

Karmaşık problemlerin çözümünde kullanılan en önemli ilkelerden biri "böl ve yönet" ilkesidir. Tüm problemi tek ve büyük bir görev olarak ele almak yerine, problem daha kolay analiz edilip çözülebilecek daha küçük görevlere ayrılır. Her küçük görev başarıyla çözüldükten sonra elde edilen çözümler birleştirilerek bütünsel bir çözüm oluşturulur. Bu yaklaşım, modüler programlamanın temelini oluşturur ve hemen hemen tüm modern programlama dillerinde ve yazılım sistemlerinde kullanılır.

Günlük yaşamda da birçok problem doğal olarak daha küçük faaliyetlere ayrılarak çözülür. Örneğin bir okul gezisi düzenlenirken tüm görevler aynı anda gerçekleştirilmez. Öncelikle güzergâh planlanır, ardından ulaşım organize edilir, konaklama için rezervasyon yapılır ve son olarak gerekli belgeler hazırlanır. Bu faaliyetlerin her biri bağımsız olarak analiz edilip gerçekleştirilebilecek ayrı bir görevdir. Aynı düşünce programlamada da uygulanır. Programcı tüm problemi bir anda çözmeye çalışmak yerine problemi birden fazla mantıksal bölüme ayırır. Böylece çözüm daha anlaşılır ve geliştirilmesi daha kolay hâle gelir. Ayrıca hataların tespit edilmesi de kolaylaşır, çünkü programın her bölümü ayrı ayrı analiz edilebilir. Bu çalışma biçimi modüler programlamanın temelini oluşturur. Modüller genellikle belirli görevleri yerine getirir ve nihai sonucu oluşturmak için birbirleriyle iş birliği yaparlar. Bu derste henüz fonksiyonlar kullanılmayacak olsa da, ilerleyen konularda fonksiyonları ve kullanım alanlarını daha kolay anlamayı sağlayacak düşünme biçimi geliştirilmeye başlanacaktır.

Örnek 1: Problem analizi

Görev: Bir öğrencinin beş notuna göre başarı durumunu hesaplayınız.
Problem aşağıdaki küçük görevlere ayrılabilir:
Notların girilmesi.
Notların toplamının hesaplanması.
Ortalamanın hesaplanması.
Başarı durumunun belirlenmesi.
Sonucun ekranda gösterilmesi.
Bu şekilde problem bir bütün olarak ele alınmak yerine birkaç basit adıma ayrılır.

Problem daha küçük görevlere ayrıldıktan sonraki adım, bu görevlerin düzenlenmesidir. Her görevin açıkça belirlenmiş bir amacı olması büyük önem taşır. Örneğin verilerin girilmesinden sorumlu bölüm aynı zamanda karmaşık hesaplamalar yapmamalıdır. Benzer şekilde sonuçları hesaplayan bölüm, verilerin ekranda gösterilmesinden sorumlu olmamalıdır. Bu tür bir ayrım, programın daha düzenli ve anlaşılır olmasını sağlar. Büyük yazılım sistemlerinde bir program, birbiriyle iletişim kuran onlarca veya yüzlerce mantıksal bölüm içerebilir.

Organizasyon iyi yapılmazsa kod karmaşık hâle gelir ve geliştirilmesi zorlaşır. Bu nedenle programlamayı öğrenmenin ilk aşamalarından itibaren problemi bölme ve çözümü açık mantıksal bölümler hâlinde organize etme alışkanlığı geliştirilmelidir. Bu düşünme biçimi daha sonra fonksiyonların, kütüphanelerin ve diğer programlama tekniklerinin kullanımını önemli ölçüde kolaylaştırır.

Örnek 2: "Böl ve yönet" ilkesine göre sözde kod

Görev: Olası bir indirimle birlikte bir ürünün nihai fiyatını hesaplayınız.
Problem aşağıdaki küçük görevlere ayrılabilir:
Ürün fiyatının girilmesi.
Miktarın girilmesi.
Toplam tutarın hesaplanması.
İndirim uygulanıp uygulanmayacağının kontrol edilmesi.
Nihai fiyatın hesaplanması.
Sonucun gösterilmesi.

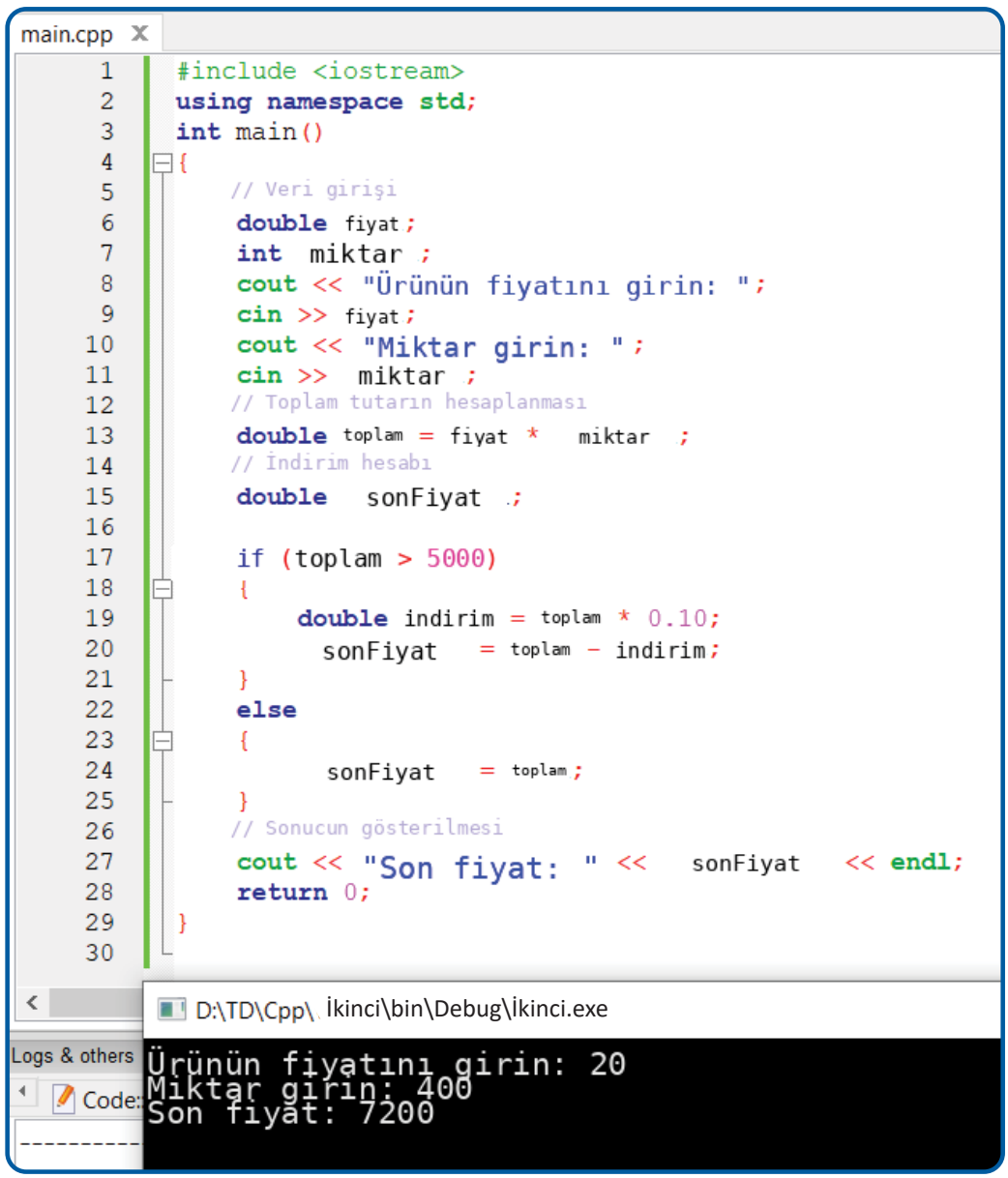
Sözde kod:

```
Başla
Fiyatı gir
Miktarı gir Toplam = fiyat × miktar
Eğer toplam > 5000 ise
  İndirim = toplam × 0,10
NihaiFiyat = toplam - indirim
Değilse
 NihaiFiyat = toplam
Eğer sonu
NihaiFiyatı göster
Bitir
```

Bu örnekte problemin öncelikle birkaç küçük göreve ayrıldığı görülmektedir. Her görevin çözüm içerisinde açıkça belirlenmiş bir amacı vardır. Bu organizasyon sayesinde algoritmanın anlaşılması kolaylaşır ve programlama dilinde uygulanmaya hazır hâle gelir. Tam olarak bu düşünme biçimi "böl ve yönet" ilkesinin özünü oluşturur ve modüler programlamanın temelini meydana getirir.

Örnek 3: Mantıksal bölümler hâlinde organize edilmiş program

Aşağıdaki örnekte, henüz fonksiyonlar kullanılmadan bir programın birden fazla mantıksal bölüme nasıl ayrılabileceği gösterilmektedir. Program, olası bir indirimle birlikte bir ürünün nihai fiyatını hesaplamaktadır.



Programda üç mantıksal bölüm açıkça görülebilir: Verilerin girilmesi, Verilerin işlenmesi ve toplam tutar ile indirimin hesaplanması, Sonucun gösterilmesi. Program main() fonksiyonu içerisinde yazılmış olsa da mantıksal bölümler kolayca ayırt edilebilir. İlerleyen derslerde bu bölümler ayrı fonksiyonlara ayrılmaya başlanacaktır. Böylece programlar daha düzenli, anlaşılır ve bakımı daha kolay hâle gelecektir. Bu, gerçek anlamda modüler programlamaya geçişteki ilk uygulamalı adımdır.

TEKRARLAMA SORULARI

SORULAR

1. Modüler programlama nedir?
2. "Böl ve yönet" ilkesi nedir?
3. Karmaşık problemler neden daha küçük görevlere ayrılır?
4. Modüler yaklaşım hata tespitine nasıl yardımcı olur?
5. Programın mantıksal bölümler hâlinde düzenlenmesi neden önemlidir?

ETKİNLİKLER

1. Maaş hesaplama programını analiz ediniz ve programı mantıksal birimlere ayırınız.
2. Yakıt tüketimi hesaplama programını analiz ediniz ve gerçekleştirilmesi gereken görevleri sırasına göre belirleyiniz.
3. Dört sınavın ortalamasını hesaplayan bir program için sözde kod yazınız.
4. Bir kütüphane kayıt programını en az dört mantıksal birime ayırınız.
5. "Böl ve yönet" ilkesinin bir okul bilgi sistemi geliştirilirken nasıl yardımcı olabileceğini açıklayınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Modüler programlama, programın daha küçük mantıksal birimlere ayrılarak düzenlenmesini ifade eder.
- "Böl ve yönet" ilkesi, karmaşık problemlerin daha küçük görevlere ayrılarak çözülmesini sağlar.
- Programın iyi organize edilmesi, kodun daha kolay anlaşılmasını ve bakımının daha kolay yapılmasını sağlar.
- Mantıksal birimler, fonksiyonların kullanılmasının temelini oluşturur.
- Modüler düşünme, her programcı için önemli bir beceridir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Büyük yazılım şirketlerinde bir programın tamamı üzerinde tek bir programcının bağımsız olarak çalışması nadirdir. Bunun yerine program çok sayıda modüle ayrılır ve her ekip sistemin belirli bir bölümünü geliştirir. Örneğin, bir çevrim içi mağazada bir ekip kullanıcı kayıt işlemleri, başka bir ekip ödeme işlemleri, üçüncü bir ekip siparişlerin işlenmesi ve dördüncü bir ekip ise istatistiksel raporların hazırlanması üzerinde çalışabilir. Her ekip farklı bir modül üzerinde çalışsa da tüm modüller, sonunda tek bir sistem olarak birlikte çalışır. Bu nedenle modüler programlama, modern yazılım geliştirmedeki en önemli yaklaşımlardan biri olup büyük ve karmaşık yazılım çözümlerinin oluşturulmasının temelini oluşturur.

2.8

C++'TA FONKSİYON TANIMLAMA

BU KONUYU TAMAMLADIĞINIZDA...

- Modüler programlamanın ne olduğunu açıklamayı;
- "Böl ve yönet" ilkesini uygulamayı;
- Bir programlama problemini analiz ederek daha küçük görevlere ayırmayı;
- Bir programı mantıksal birimler hâlinde düzenlemeyi;
- C++'ta koşul ve döngü yapılarını kullanmayı bileceksiniz.

TEKRARLAMA SORULARI

1. Modüler programlama nedir?
2. "Böl ve yönet" ilkesi nedir?
3. Büyük problemler neden daha küçük görevlere ayrılır?
4. Bir programda bulunabilecek üç mantıksal birim belirtiniz.
5. İyi bir program organizasyonu hata tespitine nasıl yardımcı olur?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Fonksiyonun ne olduğunu açıklamayı;
- C++'ta basit fonksiyonlar tanımlamayı;
- Ana programdan fonksiyonları çağırmayı;
- Fonksiyon tanımı ile fonksiyon çağrısı arasındaki farkı ayırt etmeyi;
- Birden fazla fonksiyon içeren programlar düzenlemeyi.

Programlar büyüdükçe kodun daha iyi organize edilmesine ihtiyaç duyulur. Önceki derste, karmaşık problemlerin birden fazla küçük göreve ayrılabileceği açıklanmıştı. Bu düşünce biçimi, modüler programlamanın temelini oluşturur. Ancak bu bölümlendirmenin programlama dilinde uygulanabilmesi için her görevin ayrı bir program birimi içerisinde tanımlanmasına olanak sağlayan bir mekanizmaya ihtiyaç vardır.

C++ programlama dilinde bu görevi fonksiyonlar üstlenir. Fonksiyon, programın bir parçası (alt program) olup belirli bir görevi yerine getiren, adlandırılmış komutlar bütünüdür. Aynı kodun program içerisinde birden fazla kez yazılması yerine, bu kod bir fonksiyon içerisinde tanımlanabilir ve gerektiğinde çağrılabilir. Bu sayede programlar daha düzenli, daha kısa ve bakımı daha kolay hâle gelir. Bu nedenle fonksiyonlar, modern programlamadaki en önemli araçlardan biridir.

Fonksiyon, daha büyük bir programın içerisindeki küçük bir program olarak düşünülebilir. Her fonksiyonun, çağrılmasını sağlayan bir adı vardır ve etkinleştirildiğinde çalıştırılacak komutları içerir. Her C++ programında en az bir fonksiyon bulunur: main(). Bu fonksiyon, programın çalıştırılmaya başladığı noktadır. Programcı, main() fonksiyonunun yanı sıra farklı görevleri yerine getirmek amacıyla kendi fonksiyonlarını da oluşturabilir. Bu yaklaşım, tüm kodun tek bir fonksiyon içerisinde bulunması yerine programın birden fazla bölüme ayrılarak düzenlenmesini sağlar. Her fonksiyonun açıkça belirlenmiş bir görevi olduğunda programın okunması ve anlaşılması çok daha kolay olur. Ayrıca bir hata meydana geldiğinde, tüm program yerine belirli bir fonksiyona odaklanılarak hata daha kolay tespit edilebilir. Bu nedenle fonksiyonların kullanılması, iyi programlama alışkanlıklarının geliştirilmesinde önemli bir adımdır.

Bir fonksiyon tanımlanırken adından önce fonksiyonun geri dönüş türü belirtilir. Bu tür, fonksiyonun çalışması sonucunda bir değer döndürüp döndürmeyeceğini ve döndüreceği değerin türünü belirler.

Bu derste kullanılan tür:

void

Void anahtar sözcüğü, fonksiyonun herhangi bir değer döndürmediğini ifade eder. Bu tür fonksiyonlar genellikle ekranda bir mesaj görüntülemek, başlık yazdırmak veya belirli komutları çalıştırmak gibi herhangi bir sonuç döndürmeyi gerektirmeyen görevler için kullanılır.

Bu tür bir fonksiyonun genel biçimi şöyledir:

```
void fonksiyonAdi()
 {
komutlar
}
```

Programlamada değer döndüren fonksiyonlar da kullanılır. Örneğin:

```
int toplam()
 {
return 10;
}
```

Bu durumda fonksiyon bir tam sayı (int) döndürür. Buna karşılık void türündeki fonksiyon herhangi bir değer döndürmez.

Örnek 1: Basit fonksiyon

main.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   void selamla()
6   {
7       cout << "Fonksiyonlar dünyasına hoş geldiniz! "   << endl;
8   }
9
10  int main()
11  {
12      selamla();
13
14      return 0;
15  }
16
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Fonksiyonlar dünyasına hoş geldiniz!
Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

Örnekte selam() adlı bir fonksiyon tanımlanmıştır. Fonksiyon çağrıldığında ekrana bir mesaj yazdırılır. Fonksiyonun tanımlanmasının, fonksiyonun otomatik olarak çalıştırılacağı anlamına gelmediğinin bilinmesi önemlidir. Fonksiyon içerisindeki komutların çalıştırılabilmesi için fonksiyonun program tarafından çağrılması gerekir. Fonksiyon çağrısı, fonksiyonun adının parantezlerle birlikte yazılmasıyla gerçekleştirilir. Bu mekanizma, aynı fonksiyonun aynı kod tekrar yazılmadan birden fazla kez kullanılmasına olanak sağlar. Gerçek programlarda bu yaklaşım kodun uzunluğunu önemli ölçüde azaltır ve kodun okunabilirliğini artırır. Programcılar, sıklıkla tekrarlanan görevler için fonksiyonlar oluşturur. Böylece gereksiz komut tekrarlarının önüne geçilir ve programın organizasyonu iyileştirilir.

Örnek 2: Bir Fonksiyonun Birden Fazla Kez Çağrılması

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void cizgi ()
6  {
7      cout << "====================" << endl;
8  }
9
10 int main()
11 {
12     cizgi();
13     cout << "Programın ilk kısmı " << endl;
14
15     cizgi();
16     cout << "Programın ikinci kısmı " << endl;
17
18     cizgi();
19
20     return 0;
21 }
22
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe

Programın ilk kısmı
Programın ikinci kısmı
```

Önceki örnekte aynı fonksiyon birden fazla kez çağrılmıştır. Karakterlerden oluşan satırı her seferinde yeniden programa yazmak yerine, bu satır bir fonksiyon içerisinde tanımlanmıştır. Bu sayede kod daha kısa ve düzenli hâle gelmiştir. Bu durum, fonksiyonların en önemli avantajlarından biridir: daha önce yazılmış kodun yeniden kullanılabilmesi. Büyük programlarda aynı fonksiyon onlarca veya yüzlerce kez çağrılabilir. Fonksiyonun çalışma biçiminde bir değişiklik yapılması gerektiğinde, programın tamamında değişiklik yapmak yerine yalnızca fonksiyon içerisindeki kodun değiştirilmesi yeterlidir. Bu nedenle fonksiyonlar, yazılımın geliştirilmesini ve bakımını önemli ölçüde kolaylaştırır.

Örnek 3: Birden fazla fonksiyon içeren program

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void baslik()
6  {
7      cout << "Öğrenci raporu " << endl;
8  }
9
10 void son ()
11 {
12     cout << "Raporun sonu " << endl;
13 }
14
15 int main()
16 {
17     baslik();
18
19     cout << "Öğrenci mükemmel başarı gösterdi. " << endl;
20
21     son();
22
23     return 0;
24 }
25
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Öğrenci raporu
Öğrenci mükemmel başarı gösterdi.
Raporun sonu
```

Bu örnekte iki farklı fonksiyon kullanılmaktadır. Fonksiyonlardan biri başlık görüntülemekten, diğeri ise kapanış mesajını görüntülemekten sorumludur. Böylece programdaki görevler birden fazla mantıksal birime ayrılmıştır. Program küçük olmasına rağmen fonksiyonların kodun daha iyi organize edilmesine nasıl katkı sağladığı açıkça görülebilir. Sonraki derslerde fonksiyonlara parametrelerin ve geri dönüş değerlerinin eklenmesiyle daha gelişmiş özellikler kazandırılacak ve daha karmaşık görevleri yerine getirmeleri sağlanacaktır.

TEKRARLAMA SORULARI VE ETKİNLİKLER

SORULAR

1. Fonksiyon nedir?
2. main() fonksiyonunun görevi nedir?
3. Fonksiyon çağrısı nedir?
4. Fonksiyon tanımı ile fonksiyon çağrısı arasındaki fark nedir?
5. Fonksiyonlar programların organizasyonunu neden iyileştirir?

ETKİNLİKLER

1. "İyi günler!" mesajını gösterecek bir fonksiyon yazınız.
2. 30 adet * karakterinden oluşan bir satırı gösterecek bir fonksiyon yazınız.
3. Başlık göstermek ve kapanış mesajı göstermek için iki fonksiyon kullanacak bir program yazınız.
4. Aynı fonksiyonu üç kez çağıracak bir program yazınız.
5. Fonksiyonların büyük programların geliştirilmesine nasıl yardımcı olduğunu açıklayınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Fonksiyon, adlandırılmış bir komutlar bloğudur.
- Fonksiyonlar, programın daha küçük birimlere ayrılmasını sağlar.
- Fonksiyon tanımı, fonksiyonu açıklar, fonksiyon çağrısı ise onu çalıştırır.
- Aynı fonksiyon birden fazla kez çağrılabilir.
- Fonksiyonlar, programların daha iyi organize edilmesine ve bakımının kolaylaştırılmasına katkı sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern programlama dillerinde, binlerce hazır fonksiyon içeren kütüphaneler bulunmaktadır. Programcılar her zaman kodu sıfırdan yazmak yerine, matematiksel hesaplamalar, metin işlemleri, dosya işleme ve daha birçok görev için bu fonksiyonları kullanabilirler. Bu yaklaşım sayesinde yazılım geliştirme süreci daha hızlı ve verimli hâle gelir. Bu nedenle fonksiyonlar, programlamanın en önemli kavramlarından biri ve karmaşık yazılım sistemlerinin geliştirilmesinin temelini oluşturur.

2.9

PARAMETRELER VE DEĞER DÖNDÜRME

BU KONUYU TAMAMLADIĞINIZDA...

- Fonksiyonun ne olduğunu açıklayabilirsin;
- C++ dilinde basit fonksiyonlar tanımlayabilirsin;
- Ana programdan fonksiyonları çağırabilirsin;
- Fonksiyon tanımı ile fonksiyon çağrısı arasındaki farkı ayırt edebilirsin;
- Programları birden fazla fonksiyon kullanarak düzenleyebileceksiniz.

TEKRARLAMA SORULARI

1. Fonksiyon nedir?
2. main() fonksiyonunun görevi nedir?
3. Fonksiyon çağrısı nedir?
4. "Hoş geldiniz!" mesajını gösterecek bir fonksiyon yazınız.
5. Fonksiyonlar programların daha iyi organize edilmesine neden katkı sağlar?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Fonksiyonlarda parametre kullanmayı;
- Parametreler ile argümanlar arasındaki farkı ayırt etmeyi;
- Veri alan fonksiyonlar tanımlamayı;
- Değer döndüren fonksiyonları kullanmayı;
- return komutunu kullanmayı.

Fonksiyonlar, programların birden fazla mantıksal birim hâlinde düzenlenmesini sağlar ve kodun daha anlaşılır ve bakımı daha kolay olmasına katkıda bulunur. Fonksiyonların kullanımında, her çağrılışta farklı verilerin işlenmesine ihtiyaç duyulabilir. Bu amaçla, C++ programlama dilinde, işlenmesi gereken değerlerin fonksiyona aktarılmasını sağlayan parametreler kullanılır. Böylece daha esnek programlama çözümleri oluşturulabilir ve aynı kod farklı durumlarda tekrar kullanılabilir.

Veri almanın yanı sıra, çoğu zaman fonksiyonun programın başka bir bölümünde kullanılabilecek bir sonuç döndürmesi de gerekir. Örneğin bir fonksiyon bir alanı, bir sayının karesini veya bir ortalamayı hesaplayabilir ve elde edilen sonucu programa geri döndürebilir. Bu özellik sayesinde fonksiyonlar daha güçlü ve esnek hâle gelir. Bu derste parametrelerin kullanımı ele alınacak ve değer döndüren fonksiyonların öğrenilmesine başlanacaktır.

Bir fonksiyonun farklı verileri işlemesi gerektiğinde, bu veriler parametreler aracılığıyla gönderilir. Parametreler, fonksiyonun parantezleri içinde belirtilen değişkenlerdir. Fonksiyon her çağrıldığında parametrelere yeni değerler atanır. Böylece aynı fonksiyon, kod yeniden yazılmadan farklı verileri işlemek için kullanılabilir. Bu yaklaşım, programların yeniden kullanılabilirliğini önemli ölçüde artırır ve daha esnek çözümler oluşturulmasını sağlar. Her bir değer için ayrı bir fonksiyon oluşturmak yerine, farklı verilerle çalışabilecek tek bir fonksiyon oluşturulabilir. Bu nedenle parametreler, fonksiyonların en önemli özelliklerinden biridir. Parametreler, ana program ile fonksiyon arasındaki iletişimi sağlar ve daha karmaşık programlama çözümlerinin geliştirilmesinin temelini oluşturur.

PARAMETRELER VE ARGÜMANLAR

Fonksiyonlarla çalışırken parametre ve argüman ka**vramları** sıklı**kla kull**anılır.

- Parametre, fonksiyonun tanımında belirtilen değişkendir.
- Argüman, fonksiyon çağrılırken gönderilen değerdir.

Örnek 1: Tek parametreli fonksiyon

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void sayiyiGoster(int sayi)
6  {
7      cout << "Girilen sayı: " << sayi << endl;
8  }
9
10 int main()
11 {
12     sayiyiGoster(15);
13
14     return 0;
15 }
16
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Girilen sayı: 15

Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
```

Örnekte fonksiyon, broj parametresi aracılığıyla bir değer almaktadır. Fonksiyon çağrıldığında, 15 argümanı parametreye aktarılır ve ardından ekranda gösterilir. Parametreler sayesinde aynı fonksiyon, kodunda herhangi bir değişiklik yapılmadan farklı değerlerle kullanılabilir. Bu yaklaşım, programların esnekliğini önemli ölçüde artırır ve daha kısa ve düzenli kodların oluşturulmasını sağlar.

Fonksiyonlar aynı anda birden fazla parametre alabilir. Bu gibi durumlarda, fonksiyon çağrılırken her parametreye kendi değeri atanır. Bu durum, fonksiyonun birden fazla veriyle çalışmasına ve daha karmaşık görevleri yerine getirmesine olanak sağlar. Uygulamada iki, üç veya daha fazla değer alan fonksiyonlar sıklıkla kullanılır. Bununla birlikte, argümanların sırası oldukça önemlidir, çünkü her değer, konumuna göre ilgili parametreye atanır. Bu mekanizma, programın farklı bölümleri arasındaki iletişimin temelini oluşturur. Bu sayede fonksiyonlar, her bir durum için yeni fonksiyonlar oluşturulmasına gerek kalmadan çok sayıda farklı durumda kullanılabilir.

Örnek 2: İki parametreli fonksiyon

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void dikdortgenYazdir(int satirlar, int sutunlar)
6  {
7      for (int i = 1; i <= satirlar; i++)
8      {
9          for (int j = 1; j <= sutunlar; j++)
10         {
11             cout << "* ";
12         }
13
14         cout << endl;
15     }
16 }
17
18 int main()
19 {
20     dikdortgenYazdir(3, 5);
21
22     return 0;
23 }
24
```

```
D:\TD\Cpp\İkinci\bin\De
* * * * *
* * * * *
* * * * *

Process returned 0 (0x0
Press any key to contir
```

Örnekte fonksiyon iki parametre almaktadır. İlk parametre satır sayısını, ikinci parametre ise sütun sayısını belirlemektedir. Bu sayede aynı fonksiyon, fonksiyonun kodunda herhangi bir değişiklik yapılmasına gerek kalmadan farklı boyutlarda dikdörtgenler yazdırabilir.

Bunlar, parametreler aracılığıyla veri alan fonksiyonlardır. Ancak birçok durumda fonksiyonun yalnızca belirli bir görevi yerine getirmesi yeterli değildir. Çoğu zaman, işlemin sonucunun daha sonra kullanılabilmesi için programa geri döndürülmesi gerekir. Bu amaçla değer döndüren fonksiyonlar kullanılır. Sonuç döndürmeyen void fonksiyonların aksine, bu fonksiyonların çalışmaları tamamlandıktan sonra döndürecekleri belirli bir veri türü vardır. Değerin döndürülmesi return komutu yardımıyla gerçekleştirilir. return komutu çalıştırıldığında fonksiyonun çalışması sona

erer ve elde edilen değer, fonksiyonun çağrıldığı yere geri gönderilir. Bu özellik sayesinde fonksiyonlar ifadelerin, hesaplamaların ve diğer programlama yapılarının bir parçası olarak kullanılabilir. Bu çalışma biçimi, fonksiyonların kullanım alanını önemli ölçüde genişletir ve daha karmaşık programlama çözümlerinin oluşturulmasını sağlar. Bu nedenle değer döndüren fonksiyonlar, programlamadaki en önemli araçlardan biridir.

Değer Döndüren Fonksiyonlar

Değer döndüren bir fonksiyonun genel biçimi şöyledir:

```
tip fonksiyonAdı(parametreler)
{
 komutlar
    return deger;
}
```

void yerine, fonksiyonun döndüreceği veri türü fonksiyon adının önünde belirtilir. Örneğin, fonksiyon bir tam sayı döndürüyorsa int veri türü kullanılır.

Örnek 3: Değer döndüren fonksiyon

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int  kare  (int sayi)
6  {
7      return sayi * sayi;
8  }
9
10 int main()
11 {
12     int  sonuc  ;
13
14      sonuc  =  kare  (7);
15
16     cout << "Karesi: "   <<   sonuc   << endl;
17
18     return 0;
19 }
20
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Karesi: 49

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

Örnekte kare() fonksiyonu bir parametre almakta ve return komutu yardımıyla bir sonuç döndürmektedir. Döndürülen değer daha sonra sonuc değişkenine kaydedilmektedir. Bu özellik sayesinde fonksiyonlar hesaplamalar gerçekleştirebilir ve elde edilen sonuçları programın diğer bölümlerine aktarabilir. Bu, günümüz programlamasında fonksiyonların en yaygın kullanım alanlarından biridir.

Değer döndüren fonksiyonlar, matematiksel hesaplamalarda, veri analizinde ve çeşitli programlama problemlerinin çözümünde sıklıkla kullanılır. Aynı hesaplamayı programın farklı bölümlerinde tekrar tekrar yapmak yerine, bu hesaplama bir fonksiyon içinde tanımlanabilir ve gerektiğinde kullanılabilir. Böylece kod daha kısa ve anlaşılır hâle gelir. Ayrıca hesaplama yönteminde bir değişiklik yapılması gerektiğinde yalnızca fonksiyonun değiştirilmesi yeterlidir. Programın geri kalan bölümü herhangi bir değişiklik yapılmadan çalışmaya devam eder. Bu nedenle fonksiyonlar, programların daha iyi organize edilmesine, kodun yeniden kullanılabilirliğinin artırılmasına ve programların daha kolay bakımının yapılmasına katkı sağlar. Sonraki derslerde bu kavramlar, fonksiyonlara veri aktarmanın farklı yöntemleri ve fonksiyonların kapsamındaki değişkenlerle çalışma konuları ele alınarak daha ayrıntılı biçimde incelenecektir.

TEKRARLAMA SORULARI VE GÖREVLER

SORULAR

1. Parametre nedir?
2. Argüman nedir?
3. void türündeki bir fonksiyon ile değer döndüren bir fonksiyon arasındaki fark nedir?
4. return komutunun görevi nedir?
5. Parametreli fonksiyonlar, parametresiz fonksiyonlara göre neden daha esnektir?

ÖDEVLER

1. Parametre olarak bir isim alacak ve bu ismi ekranda gösterecek İsimyazdır() fonksiyonunu yazınız.
2. İki tam sayı alacak ve büyük olan sayıyı döndürecek dahabüyük() fonksiyonunu yazınız.
3. Uzunluk ve genişlik değerlerini alacak ve dikdörtgenin alanını döndürecek dikdörtgeninalanı() fonksiyonunu yazınız.
4. Girilen sayının küpünü döndürecek küp() fonksiyonunu yazınız.
5. Karenin çevresini hesaplamak için bir fonksiyon kullanacak ve sonucu gösterecek bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Parametreler, fonksiyonun tanımında belirtilen değişkenlerdir.
- Argümanlar, fonksiyon çağrılırken gönderilen değerlerdir.
- Fonksiyonlar bir veya birden fazla parametre alabilir.
- return komutu, fonksiyondan bir değer döndürmek için kullanılır.
- Değer döndüren fonksiyonlar, hesaplamalarda ve ifadelerde kullanılabilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Büyük yazılım sistemlerinde onlarca parametre alan ve karmaşık sonuçlar döndüren fonksiyonlar bulunmaktadır. Bununla birlikte profesyonel programlamada fonksiyonların açık bir amacının olması ve tek bir belirli görevi yerine getirmesi önerilir. Bu yaklaşım, kodun okunmasını, test edilmesini ve bakımını kolaylaştırır. Birçok modern programlama dili, programcıların metinlerle, dosyalarla, ağlarla, grafiklerle ve matematiksel hesaplamalarla çalışmak için her gün kullandıkları binlerce hazır fonksiyon içerir. Fonksiyonlar, modern programlamanın en önemli temellerinden biridir ve iyi bilinmesi, karmaşık programlama çözümlerinin geliştirilmesini önemli ölçüde kolaylaştırır.

2.10

FONKSİYONLARDA YEREL VE GLOBAL DEĞİŞKENLER

BU KONUYU TAMAMLADIĞINIZDA...

- Fonksiyonları tanımlamayı ve çağırmayı;
- Fonksiyonlarda parametre kullanmayı,
- Parametreler ile argümanlar arasındaki farkı ayırt etmeyi,
- Değer döndüren fonksiyonları kullanmayı,
- return komutunu kullanmayı bileceksiniz.

TEKRARLAMA SORULARI

1. Parametre nedir?
2. Argüman nedir?
3. return komutunun görevi nedir?
4. Bir sayı alacak ve bu sayıyı ekranda gösterecek bir fonksiyon yazınız.
5. Parametreli fonksiyonlar, parametresiz fonksiyonlara göre neden daha esnektir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Bir değişkenin kapsamının ne olduğunu açıklamayı;
- Yerel ve global değişkenleri ayırt etmeyi;
- Belirli bir değişkenin nerede kullanılabileceğini analiz etmeyi;
- Birden fazla fonksiyonda global değişkenleri kullanmayı;
- Farklı değişken türlerinin avantajlarını ve dezavantajlarını tanımayı.

Programlama çözümlerinin geliştirilmesi sırasında verileri saklamak için sürekli olarak değişkenler kullanılır. Ancak her değişken programın her yerinde kullanılamaz. Belirli bir değişkene erişim olanağı, değişkenin tanımlandığı yere bağlıdır. Bu nedenle, C++ programlama dilinde değişkenin kapsamı kavramı kullanılır.

Kapsam, belirli bir değişkenin programın hangi bölümünde kullanılabileceğini belirler. Değişkenler, tanımlandıkları yere bağlı olarak yerel veya global olabilir. Bu kavramların bilinmesi oldukça önemlidir, çünkü doğru değişken türünün seçilmesi, kodun daha iyi organize edilmesine ve programlardaki hataların daha kolay bulunmasına katkı sağlar.

Programda tanımlanan her değişkenin belirli bir geçerlilik kapsamı vardır. Bu, değişkenin nerede kullanılabileceğini ve nereden erişilemeyeceğini belirleyen kuralların bulunduğu anlamına gelir. Bir değişken bir fonksiyonun içinde tanımlandığında, kullanımı yalnızca o fonksiyonla sınırlıdır. Bu tür değişkenlere yerel değişkenler denir.

Yerel değişkenler, programlarda en sık kullanılan değişken türüdür, çünkü veriler üzerinde daha iyi kontrol sağlar ve istenmeyen hataların ortaya çıkma olasılığını azaltır. Her fonksiyon, diğer fonksiyonlar tarafından görülemeyen kendi yerel değişkenlerine sahip olabilir. Bu sayede farklı fonksiyonlar, birbirlerini etkilemeden kendi verileriyle bağımsız olarak çalışabilir. Bu ilke, program kodunun daha güvenli ve daha düzenli olmasına katkı sağlar.

Örnek 1: Yerel değişken

main.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   void mesaj()
6   {
7       int sayi = 10;
8
9       cout << sayi << endl;
10  }
11
12  int main()
13  {
14      mesaj();
15
16      return 0;
17  }
18
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
10

Process returned 0 (0x0)   executi
Press any key to continue.
```

Örnekte sayı değişkeni mesaj() fonksiyonu içinde tanımlanmıştır. Bu nedenle yalnızca bu fonksiyonun içinde kullanılabilir. main() fonksiyonunda veya başka bir fonksiyonda kullanılmaya çalışılırsa program hata verecektir. Bu durum, yerel değişkenlerin kapsamının sınırlı olmasından kaynaklanır.

Yerel değişkenlerin yanı sıra programlarda global değişkenler de kullanılabilir. Yerel değişkenlerin aksine, global değişkenler tüm fonksiyonların dışında tanımlanır. Bu nedenle aynı programdaki birden fazla fonksiyon tarafından erişilebilirler. Bu durum, belirli verilerin programın farklı bölümleri arasında paylaşılmasına olanak sağlar. Ancak bu yaklaşım dikkatli kullanılmalıdır. Çok sayıda fonksiyon aynı global değişkeni kullanıp değiştirdiğinde, verilerin akışını takip etmek ve olası hataları tespit etmek daha zor hâle gelebilir. Bu nedenle, modern programlamada genellikle yerel değişkenler tercih edilir, global değişkenler ise yalnızca verilerin paylaşılmasına gerçekten ihtiyaç duyulduğunda kullanılır.

Örnek 2: Global değişken

main.cpp

```cpp
#include <iostream>

using namespace std;

int sayi = 25;

void sayiyiGoster()
{
    cout << sayi << endl;
}

int main()
{
    sayiyiGoster();

    cout << sayi << endl;

    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
25
25

Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.
```

Bu örnekte sayi değişkeni tüm fonksiyonların dışında tanımlanmıştır. Bu nedenle hem sayiyiGoster() fonksiyonunda hem de main() fonksiyonunda kullanılabilir.

Birden fazla fonksiyonla çalışırken yYerel ve global değişkenlerin aynı ada sahip olduğu bir durum ortaya çıkabilir. Böyle bir durumda, değişkenin tanımlandığı fonksiyonun içinde yerel değişkene öncelik verilir. Bu, söz konusu fonksiyon içinde aynı ada sahip global değişkenin kullanılmayacağı anlamına gelir. Bu durum, özellikle çok sayıda değişkenin bulunduğu büyük programlarda programcıların kafasını karıştırabilir. Bu nedenle değişkenleri adlandırırken açık ve anlaşılır isimler seçmeye dikkat edilmelidir. Değişkenlerin iyi organize edilmesi, kodun daha okunabilir olmasına ve mantıksal hataların daha kolay tespit edilmesine katkı sağlar. Bunun yanı sıra, yerel değişkenlerin kullanılması her fonksiyonun programın diğer bölümlerinden bağımsız olmasını sağlar. Böylece başka bir fonksiyon tarafından kullanılan bir değerin yanlışlıkla değiştirilme olasılığı azalır. Bu nedenle yerel değişkenler, programlama çözümlerinde verilerle çalışmak için genellikle önerilen yöntemdir.

Örnek 3: Aynı ada sahip yerel ve global değişken

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int sayi = 100;
6
7  void goster()
8  {
9      int sayi = 20;
10
11     cout << "Yerel değişken: " << sayi << endl;
12 }
13
14 int main()
15 {
16     goster();
17
18     cout << "Global değişken: " << sayi << endl;
19
20     return 0;
21 }
22
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Yerel değişken: 20
Global değişken: 100

Process returned 0 (0x0)   execution time : 0.323 s
Press any key to continue.
```

Örnekte aynı ada sahip iki değişken bulunmaktadır. goster() fonksiyonunda 20 değerine sahip yerel değişken kullanılırken, main() fonksiyonunda 100 değerine sahip global değişken kullanılmaktadır. Bu örnek, değişkenin tanımlandığı yerin programda hangi değişkenin kullanılacağı üzerinde büyük bir etkisi olduğunu göstermektedir.

Yerel ve global değişkenler arasındaki seçim, çözülen probleme bağlıdır. Yerel değişkenler, veriler üzerinde daha iyi kontrol sağlar ve programda istenmeyen değişikliklerin meydana gelme olasılığını azaltır. Buna karşılık, global değişkenler aynı verilerin birden fazla fonksiyonda, bu değerlerin sürekli olarak parametre olarak aktarılmasına gerek kalmadan kullanılmasına olanak sağlar. Ancak global değişkenlerin aşırı kullanılması, programın anlaşılmasını ve bakımını zorlaştırabilir. Bu nedenle uygulamada genellikle öncelikle yerel değişkenlerin kullanılması, global değişkenlerin ise yalnızca programın birden fazla bölümünde kullanılmalarına açık bir ihtiyaç olduğunda tercih edilmesi önerilir. Bu yaklaşım, daha düzenli, güvenilir ve bakımı daha kolay programlama çözümlerinin oluşturulmasına katkı sağlar.

TEKRARLAMA SORULARI VE GÖREVLER

SORULAR

1. Bir değişkenin kapsamı nedir?
2. Bir değişken ne zaman yerel olarak kabul edilir?
3. Bir değişken ne zaman global olarak kabul edilir?
4. Yerel ve global değişkenler arasındaki temel fark nedir?
5. Global değişkenlerin aşırı kullanılması neden sorunlara yol açabilir?

ÖDEVLER

1. Bir fonksiyon içinde yerel bir değişken içeren ve bu değişkenin değerini ekranda gösteren bir program yazınız.
2. Aynı global değişkeni iki farklı fonksiyonda kullanacak bir program yazınız.
3. Farklı isimlere sahip bir yerel ve bir global değişken içeren ve bu değişkenlerin değerlerini ekranda gösteren bir program yazınız.
4. Tüm fonksiyonların dışında tanımlanan global bir değişkenin hangi fonksiyonlarda kullanılabileceğini analiz ediniz.
5. Global değişken yerine yerel değişken kullanmanın daha uygun olduğu durumları açıklayınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Değişkenin kapsamı, nerede kullanılabileceğini belirler.
- Yerel değişkenler bir fonksiyon içinde tanımlanır ve yalnızca o fonksiyon içinde kullanılabilir.
- Global değişkenler tüm fonksiyonların dışında tanımlanır.
- Global değişkenler birden fazla fonksiyonda kullanılabilir.
- Yerel değişkenler genellikle programın daha iyi organize edilmesini ve daha güvenli olmasını sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Büyük programlama sistemlerinde binlerce değişken bulunabilir. Bunların tümü global olsaydı değerlerinin izlenmesi son derece zor olurdu. Bu nedenle modern programlama uygulamaları, global değişkenlerin kullanımının sınırlandırılmasını ve mümkün olduğunca yerel değişkenlerin kullanılmasını önerir. Bu yaklaşım, program kodunun daha iyi organize edilmesini, fonksiyonların daha kolay test edilmesini ve yazılım çözümlerinin geliştirilmesi sırasında mantıksal hataların sayısının azaltılmasını sağlar.

2.11

CMATH KÜTÜPHANESINDEKI TEMEL MATEMATIKSEL FONKSIYONLAR (BÖLÜM 1)

BU KONUYU TAMAMLADIĞINIZDA...

- Fonksiyonları tanımlamayı ve çağırmayı,
- Fonksiyonlarda parametre kullanmayı;
- Değer döndüren fonksiyonları kullanmayı;
- Yerel ve global değişkenleri ayırt etmeyi,
- Programı birden fazla mantıksal birime ayırarak düzenlemeyi bileceksiniz.

TEKRARLAMA SORULARI

1. Yerel değişken nedir?
2. Global değişken nedir?
3. Yerel bir değişken programın hangi bölümünde kullanılabilir?
4. Yerel değişkenlerin temel avantajı nedir?
5. Bir sayı alan ve sayının karesini döndüren bir fonksiyon yazınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- cmath kütüphanesini kullanmayı;
- Hazır matematiksel fonksiyonları uygulamayı;
- Bir sayının karekökünü ve kuvvetini hesaplamayı;
- Yuvarlama fonksiyonlarını kullanmayı;
- Trigonometrik fonksiyonları uygulamayı;
- Programlama çözümlerinde matematiksel fonksiyonları kullanmayı.

Programların geliştirilmesi sırasında sıklıkla matematiksel hesaplamaların yapılması gerekir. Bu hesaplamaların bir bölümü aritmetik operatörlerle kolayca gerçekleştirilebilir, ancak bazı işlemler için daha karmaşık algoritmalara ihtiyaç duyulur. Örneğin karekök hesaplama, üs alma veya trigonometrik fonksiyonların hesaplanması elle gerçekleştirilecek olsaydı, çok daha fazla sayıda komut yazılması gerekirdi.

Programcıların çalışmalarını kolaylaştırmak amacıyla C++ programlama dili, farklı kütüphanelerde bulunan çok sayıda hazır fonksiyon içerir. En sık kullanılan kütüphanelerden biri, matematiksel hesaplamalar için gerekli fonksiyonları içeren cmath kütüphanesidir. Bu fonksiyonlar sayesinde karmaşık işlemler, ek kod yazmaya gerek kalmadan hızlı ve kolay bir şekilde gerçekleştirilebilir. Bunun sonucunda programlar daha kısa, daha anlaşılır ve bakımı daha kolay hâle gelir.

cmath kütüphanesi, programlarda kullanılabilecek hazır matematiksel fonksiyonlar bütünüdür. Bu fonksiyonları kullanabilmek için programın başlangıcında #include <cmath> yönergesi aracılığıyla kütüphanenin programa eklenmesi gerekir. Kütüphane programa eklendikten sonra, program farklı matematiksel işlemleri gerçekleştiren çok sayıda fonksiyona erişim kazanır. Programcı, karekök veya bir sayının kuvvetini hesaplamak için kendi algoritmasını geliştirmek yerine, doğrudan kütüphanede bulunan fonksiyonları kullanabilir. Bu yaklaşım, programların geliştirilmesi için gereken süreyi azaltır ve sonuçların daha güvenilir olmasına katkı sağlar. cmath kütüphanesindeki fonksiyonlar kapsamlı bir şekilde test edilmiş olup çok sayıda profesyonel yazılım çözümünde kullanılmaktadır. Bu nedenle bu fonksiyonların bilinmesi, programlama becerilerinin önemli bir bölümünü oluşturur.

Örnek 1: sqrt() fonksiyonu

sqrt() fonksiyonu, bir sayının karekökünü hesaplamak için kullanılır.

main.cpp

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double sayi = 81;
9
10     double karekok = sqrt(sayi);
11
12     cout << "Karekök: "          << karekok << endl;
13
14     return 0;
15 }
16
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Karekök: 9

Process returned 0 (0x0)   execution time : 0.334 s
Press any key to continue.
```

Örnekte 81 sayısının karekökü hesaplanmaktadır. Elde edilen sonuç 9'dur. sqrt() fonksiyonu bir sayıyı argüman olarak alır ve bu sayının karekökünün değerini döndürür.

Karekökün yanı sıra, sayıların kuvvetinin alınması da sıklıkla kullanılır. Matematikte kuvvet alma, bir sayının belirli bir sayıda kendisiyle çarpılması işlemidir. cmath kütüphanesinde bu işlem pow() fonksiyonu aracılığıyla gerçekleştirilir. Bu fonksiyon iki argüman alır. Birinci argüman tabanı, ikinci argüman ise sayının alınacağı kuvveti ifade eder. Bu fonksiyon sayesinde çok sayıda matematiksel formül ve algoritma kolaylıkla uygulanabilir. Hazır fonksiyonların kullanılması, aynı işlemlerin elle gerçekleştirilmesine kıyasla kodun daha kısa ve anlaşılır olmasına katkı sağlar. Bu nedenle pow() fonksiyonu, cmath kütüphanesinde en sık kullanılan fonksiyonlardan biridir.

Örnek 2: pow() fonksiyonu

main.cpp

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double   sonuc  ;
9
10       sonuc   = pow(2,5);
11
12     cout << "Sonuç: "      <<   sonuc   << endl;
13
14     return 0;
15 }
16
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Sonuç: 32

Process returned 0 (0x0)   execution time : 0.327 s
Press any key to continue.
```

Örnekte 2^5 ifadesinin değeri hesaplanmaktadır. Elde edilen sonuç 32'dir. Birinci argüman tabanı, ikinci argüman ise kuvveti ifade eder.

Karekök ve kuvvet hesaplama fonksiyonlarının yanı sıra cmath kütüphanesinde mutlak değer ve sayıların yuvarlanması için kullanılan fonksiyonlar da bulunmaktadır. Bir sayının mutlak değeri, sayı doğrusunda sıfıra olan uzaklığını ifade eder ve her zaman negatif olmayan bir değere sahiptir. Mutlak değeri hesaplamak için fabs() fonksiyonu kullanılır. Ondalıklı sayılarla çalışırken değerlerin yuvarlanmasına sıklıkla ihtiyaç duyulur. Bu amaçla ceil() ve floor() fonksiyonları kullanılır. ceil() fonksiyonu sayıyı bir sonraki büyük tam sayıya yuvarlarken, floor() fonksiyonu sayıyı bir önceki küçük tam sayıya yuvarlar. Bu fonksiyonlar, çeşitli programlama çözümlerinde geniş bir kullanım alanına sahiptir. Özellikle elde edilen sonuçların tam sayı biçiminde ifade edilmesi gerektiğinde kullanılırlar. Bu fonksiyonlar sayesinde programlar, ondalıklı değerleri kolayca işleyebilir ve bunları çözülen problemin gereksinimlerine uygun hâle getirebilir.

cmath kütüphanesinde sık kullanılan bazı fonksiyonlar

Fonksiyon	İşlev
sqrt(x)	Karekök
pow(a,b)	Kuvvet alma
fabs(x)	Mutlak değer
ceil(x)	Yukarı yuvarlama
floor(x)	Aşağı yuvarlama
sin(x)	Sinüs
cos(x)	Kosinüs

Örnek 3: Birden fazla matematiksel fonksiyonun kullanılması

main.cpp

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double sayi = -15.7;
9
10     cout << "Mutlak değer: "
11          << fabs(sayi) << endl;
12
13     cout << "Yukarı yuvarlama: "
14          << ceil(sayi) << endl;
15
16     cout << "Aşağı yuvarlama: "
17          << floor(sayi) << endl;
18
19     return 0;
20 }
21
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Mutlak değer: 15.7
Yukarı yuvarlama: -15
Aşağı yuvarlama: -16

Process returned 0 (0x0)   execution time : 1.645 s
Press any key to continue.
```

Örnekte cmath kütüphanesinden üç farklı fonksiyon kullanılmaktadır. fabs() fonksiyonu ile sayının mutlak değeri elde edilirken, ceil() ve floor() fonksiyonları farklı yuvarlama işlemleri gerçekleştirir. Bu tür işlemler, programlarda veri işleme ve matematiksel hesaplamaların gerçekleştirilmesi sırasında sıklıkla kullanılır.

cmath kütüphanesinde matematik, fizik, mühendislik ve bilgisayar grafiklerinde önemli kullanım alanlarına sahip trigonometrik fonksiyonlar da bulunmaktadır. En sık kullanılanlar, verilen bir açının sinüsünü ve kosinüsünü hesaplayan sin() ve cos() fonksiyonlarıdır. Bu fonksiyonları kullanırken açıların radyan cinsinden verilmesi gerektiği bilinmelidir. Okul matematiğinde genellikle derece ile çalışılmasına rağmen, bilgisayar programlarında açıların ölçü birimi olarak genellikle radyan kullanılır. Trigonometrik fonksiyonlar, simülasyonların, oyunların, animasyonların, grafik uygulamalarının ve çok sayıda bilimsel programın geliştirilmesinde kullanılır. cmath kütüphanesi sayesinde bu hesaplamalar, özel matematiksel algoritmalar geliştirmeye gerek kalmadan kolayca gerçekleştirilebilir.

SORULAR VE TEKRARLAMA GÖREVLERİ

SORULAR

1. cmath kütüphanesi ne için kullanılır?
2. sqrt() fonksiyonunun kullanım amacı nedir?
3. pow() fonksiyonunun kullanım amacı nedir?
4. ceil() ve floor() fonksiyonları arasındaki fark nedir?
5. sin() ve cos() fonksiyonları ne için kullanılır?

ÖDEVLER

1. Girilen sayının karekökünü hesaplayan bir program yazınız.
2. pow() fonksiyonunu kullanarak 3^4 ifadesinin değerini hesaplayan bir program yazınız.
3. Girilen bir gerçek sayının mutlak değerini gösteren bir program yazınız.
4. Girilen bir sayı için ceil() ve floor() sonuçlarını gösteren bir program yazınız.
5. Radyan cinsinden verilen bir açının sinüsünü ve kosinüsünü hesaplayan bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

1. cmath kütüphanesi, hazır matematiksel fonksiyonları içerir.
2. sqrt() fonksiyonu, karekök hesaplamak için kullanılır.
3. pow() fonksiyonu, kuvvet alma işlemi için kullanılır.
4. fabs() fonksiyonu, bir sayının mutlak değerini hesaplar.
5. ceil() ve floor() fonksiyonları, yuvarlama işlemleri için kullanılır.
6. sin() ve cos() fonksiyonları, trigonometrik hesaplamalar için kullanılır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

cmath kütüphanesi, bu derste ele alınanlardan çok daha fazla fonksiyon içerir. Bu kütüphanede logaritma hesaplama, üstel fonksiyonlar, trigonometrik ve ters trigonometrik fonksiyonların yanı sıra çok sayıda başka matematiksel işlem için kullanılan fonksiyonlar da bulunmaktadır. Profesyonel programlama çözümlerinde bu fonksiyonlar, bilimsel simülasyonların, mühendislik uygulamalarının, finansal analizlerin, bilgisayar oyunlarının ve veri işleme sistemlerinin geliştirilmesinde kullanılır. cmath kütüphanesinin bilinmesi, programcının matematiksel problemleri çok daha hızlı çözmesine ve daha verimli programlama çözümleri oluşturmasına olanak sağlar.

2.12

FONKSİYONLAR KULLANILARAK METİNSEL PROBLEMLERİN ÇÖZÜLMESİ

BU KONUYU TAMAMLADIĞINIZDA...

- Fonksiyonları tanımlamayı ve çağırmayı;
- Parametreleri ve argümanları kullanmayı;
- Değer döndüren fonksiyonları kullanmayı;
- Yerel ve global değişkenleri ayırt etmeyi;
- cmath kütüphanesindeki fonksiyonları uygulamayı bileceksiniz.

TEKRARLAMA SORULARI

1. cmath kütüphanesi ne için kullanılır?
2. sqrt() fonksiyonunun kullanım amacı nedir?
3. ceil() ve floor() fonksiyonları arasındaki fark nedir?
4. Değer döndüren fonksiyonlar hangi durumlarda kullanılır?
5. return komutunun görevi nedir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Metinsel bir problemi analiz etmeyi;
- Çözümün hangi bölümlerinin fonksiyon olarak gerçekleştirilmesi gerektiğini belirlemeyi;
- Bir programda birden fazla fonksiyonu bir arada kullanmayı;
- Problem çözerken parametreleri ve döndürülen değerleri kullanmayı;
- Daha düzenli programlama çözümleri oluşturmayı.

Fonksiyonlar, program kodunun düzenlenmesi için en yararlı araçlardan biridir. Gerçek değerleri, bir programın birden fazla farklı görevi yerine getirmesi gereken gerçek problemlerin çözümünde daha iyi anlaşılır. Tüm komutları main() fonksiyonuna yerleştirmek yerine program, birden fazla küçük birime ayrılabilir ve her fonksiyona açıkça belirlenmiş bir görev verilebilir.

Metinsel problemlerin çözümünde öncelikle problemin analizi yapılmalıdır. Daha sonra hangi işlemlerin tekrarlandığı ve çözümün hangi bölümlerinin ayrı fonksiyonlar hâlinde gerçekleştirilebileceği belirlenmelidir. Bu yaklaşım, programların daha düzenli, anlaşılır ve bakımı daha kolay olmasını sağlar.

Bir programlama problemini çözerken öncelikle hangi verilerin girileceği, hangi hesaplamaların yapılacağı ve hangi sonucun gösterileceği belirlenmelidir. Bu analiz yapıldıktan sonra çözüm birden fazla mantıksal birime ayrılabilir. Her birim ayrı bir fonksiyon olarak gerçekleştirilebilir. Böylece tek ve büyük, karmaşık bir bütün yerine birden fazla küçük bölümden oluşan bir program elde edilir. Bu yaklaşım, «böl ve yönet» ilkesini temel alır ve modüler programlamanın temelini oluşturur.

Örnek 1: Bilet ücretinin hesaplanması

Bir sinemada biletler 250 denar fiyatından satılmaktadır. Girilen satılmış bilet sayısına göre toplam geliri hesaplayınız.

main.cpp

```
#include <iostream>

using namespace std;

int gelir(int biletSayisi)
{
    return biletSayisi * 250;
}

int main()
{
    int biletSayisi;

    cout << "Bilet sayısını girin: " ;
    cin >> biletSayisi;

    cout << "Gelir: "
         << gelir(biletSayisi)
         << "denar " << endl;

    return 0;
}
```

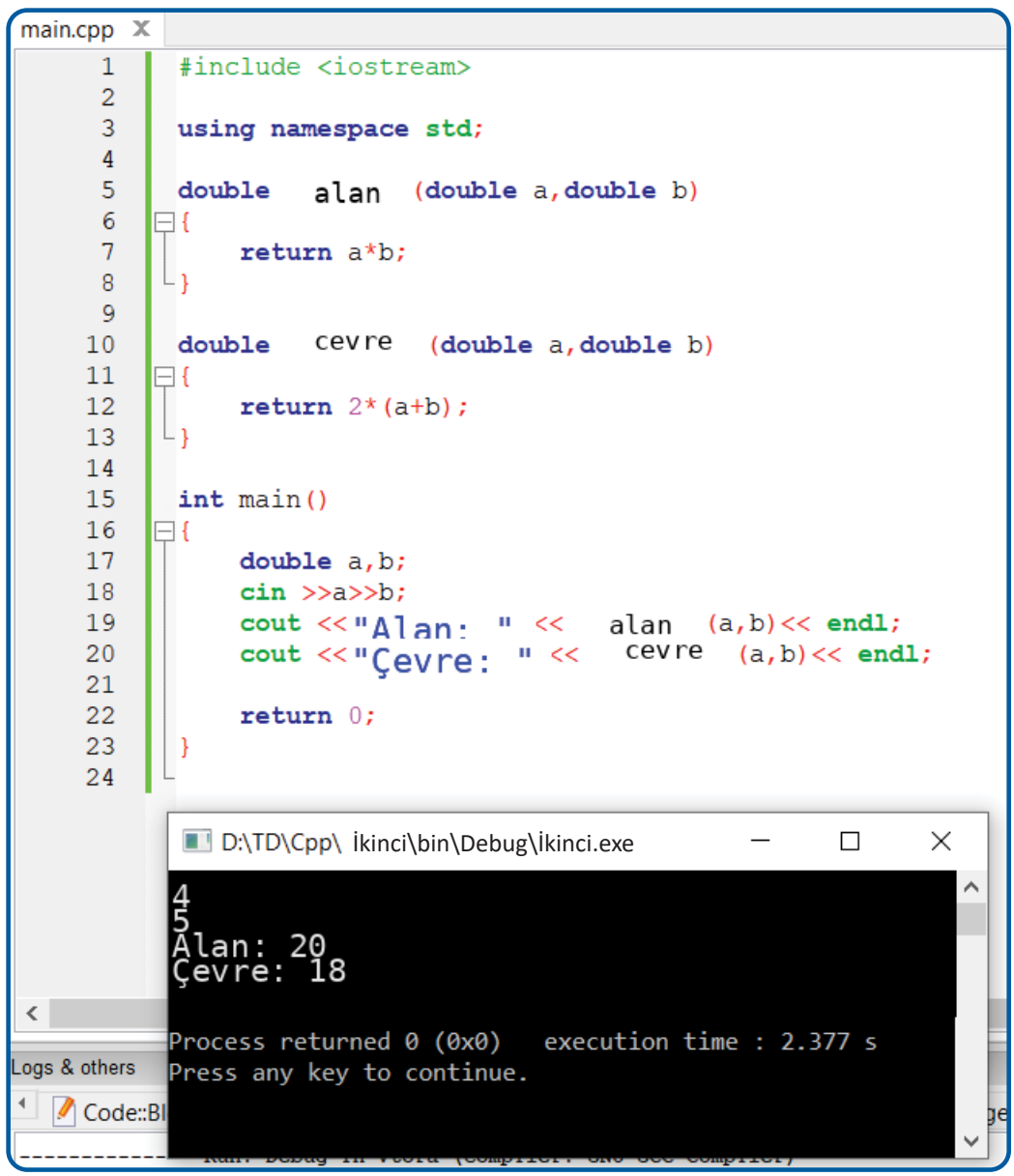
```
D:\TD\Cpp\İkinci\bin\Debug\İkinci.exe
Bilet sayısını girin: 15
Gelir: 3750 denar

Process returned 0 (0x0)   execution time : 1.607 s
Press any key to continue.
```

Örnekte, gelirin hesaplanması ayrı bir fonksiyon hâlinde gerçekleştirilmiştir. Bu sayede çözüm daha düzenli ve bakımı daha kolay hâle gelmiştir.

Birçok durumda bir programın birden fazla fonksiyon kullanması gerekir. Her fonksiyon farklı bir görevi yerine getirebilir ve daha sonra elde edilen sonuçlar ana programda birleştirilebilir. Bu yaklaşım, özellikle problemin birbirinden bağımsız olarak gerçekleştirilebilecek birden fazla hesaplama içermesi durumunda oldukça yararlıdır.

Örnek 2: Alan ve çevrenin hesaplanması

Bir dikdörtgenin uzunluk ve genişlik değerlerini giriniz. Alanını ve çevresini hesaplayınız.



Örnekte her fonksiyonun açıkça belirlenmiş bir görevi vardır. Fonksiyonlardan biri alanı, diğeri ise çevreyi hesaplamaktadır.

Uygulamalı programlama çözümlerinde çoğu zaman yalnızca bir fonksiyonun kullanılması yeterli değildir. Daha büyük problemler genellikle birbirinden bağımsız olarak çözülmesi gereken birden fazla küçük görevden oluşur. Bu nedenle problem analiz edilirken öncelikle hangi işlemlerin gerçekleştirileceği belirlenmeli, ardından bu işlemlerin her biri ayrı bir fonksiyon hâlinde gerçekleştirilmelidir. Bu şekilde program açık bir yapıya kavuşur ve her fonksiyon belirli bir sorumluluk üstlenir. Program kodu bu şekilde düzenlendiğinde hataları tespit etmek ve yeni özellikler eklemek çok daha kolay olur. Ayrıca fonksiyonlar başka programlarda da yeniden kullanılabilir. Tam da bu nedenle, problemin analiz edilmesi ve görevlerin doğru şekilde bölünmesi, her programlama çözümünün geliştirilmesinde önemli bir adımdır.

Örnek 3: Mars'taki araştırma istasyonu

Mars'taki bir araştırma istasyonunda her gün güneş panelleri tarafından üretilen elektrik enerjisi miktarı ile yaşam destek sistemleri tarafından tüketilen enerji miktarı kaydedilmektedir. Enerji fazlası mı yoksa enerji açığı mı oluştuğunu hesaplayınız.

main.cpp

```
#include <iostream>
using namespace std;
double enerjiUret(double panelAdedi, double panelEnerjisi)
{
    return panelAdedi * panelEnerjisi;
}
double bilanco(double uretilen, double tuketilen)
{
    return uretilen - tuketilen;
}
int main()
{
    double panelAdedi;
    double panelEnerjisi;
    double tuketilen;
    cout << "Panel sayisini girin: " ;
    cin >> panelAdedi;
    cout << "Panel başına enerji: " ;
    cin >> panelEnerjisi;
    cout << "Tüketilen enerji: ";
    cin >> tuketilen;
    double uretilen = enerjiUret(panelAdedi, panelEnerjisi);
    double sonuc = bilanco(uretilen, tuketilen);
    cout << "Enerji bilançosu: "
         << sonuc
         << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Panel sayisini girin: 25
Panel başına enerji: 22
Tüketilen enerji: 473
Enerji bilançosu: 77

Press any key to continue.
```

Örnekte problem, birbirinden bağımsız iki göreve ayrılmıştır. İlk fonksiyonda üretilen toplam enerji hesaplanırken, ikinci fonksiyonda enerji dengesi hesaplanmaktadır. Bu tür bir düzenleme sayesinde her fonksiyonun açıkça belirlenmiş bir amacı olur ve programın anlaşılması kolaylaşır.

Fonksiyonlar, kaliteli programlama çözümlerinin geliştirilmesi için önemli bir araçtır. Fonksiyonların kullanılmasıyla karmaşık problemler, daha küçük ve daha kolay yönetilebilir birimlere ayrılabilir. Bu sayede program kodunun daha iyi organize edilmesi, çözümün ayrı bölümlerinin daha kolay test edilmesi ve hataların daha verimli bir şekilde tespit edilmesi sağlanır. Metinsel bir problem analiz edilirken, hangi işlemlerin fonksiyonlara ayrılabileceği ve hangi verilerin parametreler aracılığıyla aktarılması gerektiği her zaman belirlenmelidir. Bu yaklaşım, daha büyük programlama projelerinin ve modern yazılım sistemlerinin geliştirilmesinin temelini oluşturur. Bu nedenle

fonksiyonlar, C++ programlama dilindeki en önemli kavramlar arasında yer almakta ve veri depolama ve işleme yapılarının daha ileri düzeyde öğrenilmesinin temelini oluşturmaktadır.

SORULAR VE TEKRARLAMA GÖREVLERİ

SORULAR

1. Karmaşık problemlerin birden fazla fonksiyona bölünmesi neden yararlıdır?
2. Yeni bir fonksiyon oluşturmak ne zaman gereklidir?
3. Metinsel problemlerin çözümünde parametrelerin rolü nedir?
4. Değer döndüren fonksiyonların avantajı nedir?
5. Fonksiyonlar, program kodunun daha iyi organize edilmesine nasıl katkı sağlar?

ÖDEVLER

1. Bir hayvanat bahçesinde günlük yem tüketimiyle ilgili kayıt tutulmaktadır. Biri aslanlar için gerekli yem miktarını, diğeri ise kaplanlar için gerekli yem miktarını hesaplayan iki fonksiyonun kullanılacağı bir program yazınız.
2. Bir havaalanında uçaklardaki yakıt miktarı takip edilmektedir. Bir fonksiyonun toplam doldurulan yakıt miktarını, diğer fonksiyonun ise uçuş sonrasında kalan yakıt miktarını hesaplayacağı bir program yazınız.
3. Bir botanik bahçesinde dairesel çiçek alanları oluşturulmaktadır. Bir fonksiyonun alanı, diğer fonksiyonun ise metrekare başına fiyat klavyeden girildiğinde düzenleme maliyetini hesaplayacağı bir program yazınız.
4. Bir su altı araştırma istasyonunda oksijen tüketimi izlenmektedir. Belirli bir çalışma süresinden sonra kalan oksijen miktarını hesaplayan bir fonksiyonun kullanılacağı bir program yazınız.
5. Bir astronomi gözlemevinde, bir sinyalin uydudan Dünya'ya ulaşması için gereken süre hesaplanmaktadır. Girilen mesafeye göre bu değeri hesaplayan bir fonksiyonun kullanılacağı bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Metinsel problem öncelikle analiz edilmelidir.
- Karmaşık problemler birden fazla küçük göreve ayrılabilir.
- Her görev ayrı bir fonksiyon olarak gerçekleştirilebilir.
- Fonksiyonlar, kodun daha iyi organize edilmesini ve yeniden kullanılmasını sağlar.
- Parametreler ve döndürülen değerler, fonksiyonlar ile program arasında veri alışverişini sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilgisayar oyunları, mobil uygulamalar ve işletim sistemleri milyonlarca satır program kodu içerir. Tüm kod tek bir fonksiyonun içine yerleştirilseydi bu tür sistemler geliştirilemezdi. Bu nedenle programcılar programları çok sayıda fonksiyon ve modül hâlinde düzenler; her fonksiyon toplam problemin küçük bir bölümünü çözer. Bu şekilde birden fazla programcının aynı proje üzerinde eş zamanlı olarak çalışması mümkün olur ve programların bakımı ile geliştirilmesi kolaylaşır. Böylece fonksiyonlar konusu tamamlanır ve tek boyutlu diziler gibi çok sayıda verinin işlenmesine olanak sağlayan yapıların öğrenilmesi için temel oluşturulur.

2.13

FONKSIYONLAR ÜZERINE ALIŞTIRMA SORULARI

C++'TA FONKSIYONLAR İÇİN ALIŞTIRMA SORULARI

Kolay Sorular

1. Bir meteoroloji istasyonunda hava sıcaklığı günlük olarak takip edilmektedir. Santigrat derece cinsinden sıcaklığı alan ve buna karşılık gelen Fahrenheit sıcaklığını hesaplayan bir fonksiyon içeren program yazınız.
2. Bir inşaat şirketinde kare şeklinde beton plakalar üretilmektedir. Plakanın bir kenar uzunluğunu alan ve çevresini hesaplayan bir fonksiyon içeren program yazınız.
3. Bir turizm acentesi, birkaç gün süren bir gezi düzenlemektedir. Gün sayısını alan ve bu sürenin kaç tam hafta içerdiğini hesaplayan bir fonksiyon içeren program yazınız.
4. Bir mağazada sezon sonu indirimi düzenlenmektedir. Ürünün fiyatını ve indirim yüzdesini alan ve yeni fiyatı hesaplayan bir fonksiyon içeren program yazınız.
5. Bir sağlık araştırmasında katılımcıların yaşları girilmektedir. Yaşı alan ve 100 yaşına ulaşmasına kaç yıl kaldığını hesaplayan bir fonksiyon içeren program yazınız.
6. Bir parkta dairesel bir çeşme yapılması planlanmaktadır. Çeşmenin yarıçapını alan ve çevre uzunluğunu hesaplayan bir fonksiyon içeren program yazınız.
7. Bir spor salonunda antrenmanların süresi kaydedilmektedir. Dakika sayısını alan ve bunun kaç saate karşılık geldiğini hesaplayan bir fonksiyon içeren program yazınız.
8. Bir laboratuvarda iki ölçümün sonuçları karşılaştırılmaktadır. İki değeri alan ve aralarındaki mutlak farkı hesaplayan bir fonksiyon içeren program yazınız. Fonksiyonda fark adlı yerel bir değişken kullanınız.
9. Bir akaryakıt istasyonunda yakıt miktarı analiz edilmektedir. Litre cinsinden yakıt miktarını alan ve aynı miktarı mililitre cinsinden hesaplayan bir fonksiyon içeren program yazınız.
10. Bir bankada hesap bakiyesi analiz edilmektedir. Bir sayı alan ve değerin pozitif olup olmadığını belirleyen bir fonksiyon içeren program yazınız.

Orta Düzey Sorular

1. Bir ortaçağ kalesinde her sakin aylık 12 sikke vergi ödemektedir. Sakin sayısını alan ve toplam aylık vergiyi hesaplayan bir fonksiyon içeren program yazınız.

2. Bir atölyede üçgen bir çerçeve oluşturmak için üç ahşap çıta kontrol edilmektedir. Üç uzunluğu alan ve bunlarla bir üçgen oluşturulup oluşturulamayacağını kontrol eden bir fonksiyon içeren program yazınız.
3. Bir okul yarışmasında hem 3'e hem de 5'e bölünebilen sayılara özel bir işaret verilmektedir. Bir sayı alan ve bu koşulu sağlayıp sağlamadığını kontrol eden bir fonksiyon içeren program yazınız.
4. Bir haritada bir nesnenin konumu x ve y koordinatlarıyla gösterilmektedir. Koordinatları alan ve nesnenin koordinat başlangıç noktasına olan uzaklığını hesaplayan bir fonksiyon içeren program yazınız.
5. Bir evde elektrik faturası hesaplanmaktadır. cenaKwh = 7 adlı global bir değişken ve tüketilen fiyatKwh miktarını alan ve ödenecek tutarı hesaplayan bir fonksiyon içeren program yazınız.
6. Bir dağ evinde bir gecelik konaklama 850 denardır. fiyatKonaklama = 850 adlı global bir değişken ve gece sayısını alan ve toplam tutarı hesaplayan bir fonksiyon içeren program yazınız.
7. Bir bilgi sisteminde girilen kimlik numarasının uzunluğu kontrol edilmektedir. Doğal bir sayı alan ve kaç basamak içerdiğini hesaplayan bir fonksiyon içeren program yazınız.
8. Bir spor dalında üç yarışmacının sonuçları karşılaştırılmaktadır. Üç sonucu alan ve en büyük sonucu döndüren bir fonksiyon içeren program yazınız. Fonksiyonda enBuyuk adlı yerel bir değişken kullanınız.
9. Bir okul laboratuvarında derece cinsinden verilen bir açının sinüsü hesaplanmaktadır. Açıyı derece cinsinden alan, radyana dönüştüren ve cmath kütüphanesini kullanarak sinüsünü hesaplayan bir fonksiyon içeren program yazınız.
10. Bir finans uygulamasında yuvarlanması gereken gerçek sayılar işlenmektedir. Gerçek bir sayı alan ve ceil() ile floor() sonuçlarını gösteren bir fonksiyon içeren program yazınız.

Zor Sorular

1. Bir müzede yetişkinler ve çocuklar için bilet satılmaktadır. fiyatYetiskin = 200 ve fiyatCocuk = 120 adlı global değişkenlerin yanı sıra yetişkin ve çocuk sayılarını alan ve toplam geliri hesaplayan bir fonksiyon içeren program yazınız.
2. Bir takvim uygulamasında belirli bir yılın artık yıl olup olmadığı kontrol edilmektedir. Yılı alan ve o yılın artık yıl olup olmadığını döndüren bir fonksiyon içeren program yazınız.
3. Bir güvenlik sisteminde bir sayı, rakamlarının toplamı üzerinden analiz edilmektedir. Doğal bir sayı alan ve rakamlarının toplamını hesaplayan bir fonksiyon içeren program yazınız.
4. Bir matematik uygulamasında ardışık sayıların çarpımları hesaplanmaktadır. Doğal bir sayı alan ve faktöriyelini hesaplayan bir fonksiyon içeren program yazınız.
5. Bir video oyununda oyuncu, topladığı altın miktarına göre seviyenin kilidini açmaktadır. Altın sayısını alan ve açılan seviyeyi belirleyen bir fonksiyon içeren program yazınız: 100 altına kadar – 1. seviye, 101 ile 300 arası – 2. seviye, 300'den fazla – 3. seviye.
6. Bir fabrikada iki ürün serisi, hiç artan kalmayacak şekilde eşit gruplara ayrılmaktadır. İki sayıyı alan ve bunların en büyük ortak bölenini hesaplayan bir fonksiyon içeren program yazınız.
7. Bir zaman ölçüm sistemine süre saniye cinsinden girilmektedir. Saniye sayısını alan ve saat, dakika ve saniye değerlerini gösteren bir fonksiyon içeren program yazınız. Fonksiyonda saat, dakika ve saniye adlı yerel değişkenleri kullanınız.
8. Bir meteoroloji istasyonunda sıcaklık ve nem ölçülmektedir. İki değeri alan ve sıcaklık 5 °C'nin altında, nem ise %85'in üzerinde olduğunda sis oluşma ihtimali olup olmadığını döndüren bir fonksiyon içeren program yazınız.
9. Bir navigasyon uygulamasında harita üzerindeki iki nokta arasındaki uzaklık hesaplanmaktadır. İki noktanın koordinatlarını alan ve aralarındaki uzaklığı hesaplayan bir fonksiyon içeren program yazınız.

10. Bir astronomi gözlemevinde bir kuyruklu yıldızın hareketi takip edilmektedir. Alınan mesafeyi ve geçen süreyi alan ve ortalama hızı hesaplayan bir fonksiyon içeren program yazınız. Fonksiyonda hiz adlı yerel bir değişken kullanınız.

2.14

VERİ YAPISI OLARAK TEK BOYUTLU DİZİ

BU KONUYU TAMAMLADIĞINIZDA...

- Farklı türlerde değişkenleri kullanmayı;
- Veri girmeyi ve verileri görüntülemeyi;
- Koşul yapılarını kullanmayı;
- Döngüleri kullanmayı;
- Fonksiyonları tanımlamayı ve çağırmayı bileceksiniz.

TEKRARLAMA SORULARI

1. Fonksiyon nedir?
2. Parametrelerin görevi nedir?
3. return komutu ne için kullanılır?
4. Bir programın birden fazla fonksiyona bölünmesi neden yararlıdır?
5. cmath kütüphanesinden bir fonksiyon belirtiniz.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Dizinin ne olduğunu açıklamayı;
- Dizi kullanımının gerekli olduğu durumları belirlemeyi;
- Değişken ile dizi arasındaki farkı ayırt etmeyi;
- Dizi elemanlarının bellekte nasıl saklandığını açıklamayı;
- Basit programlama çözümlerinde tek boyutlu dizileri kullanmayı.

Programlama çözümlerinin geliştirilmesi sırasında sıklıkla aynı türden çok sayıda verinin işlenmesi gerekir. Örneğin, bir sınıftaki öğrencilerin notlarının, bir ay boyunca günlük sıcaklıkların veya bir spor turnuvasındaki yarışmacıların sonuçlarının saklanması gerekebilir. Her veri için ayrı bir değişken kullanılması durumunda program kısa sürede karmaşık ve bakımı zor bir hâle gelir.

C++ programlama dilinde verilerin daha verimli bir şekilde düzenlenmesi için diziler kullanılır. Dizi, aynı türden birden fazla verinin tek bir ortak ad altında saklanmasını sağlar. Bu sayede veriler daha kolay girilebilir, işlenebilir ve analiz edilebilir. Bu nedenle diziler, programlamada veri depolamak için kullanılan en önemli yapılardan biridir.

Bir değişken, bilgisayarın belleğinde tek bir değerin saklanmasını sağlar. Aynı türden on, yirmi veya yüz değerin saklanması gerektiğinde çok sayıda ayrı değişken kullanmak pratik bir çözüm değildir. Bu tür durumlarda dizi kullanılır. Dizi, aynı veri türüne sahip ve bellekte art arda saklanan elemanlardan oluşan bir bütündür. Tüm elemanlar ortak bir ad ile ilişkilendirilir ve her elemanın kendine ait bir konumu vardır. Bu düzenleme sayesinde çok sayıda veri tek bir bütün olarak işlenebilir. Diziler, verilerin verimli bir şekilde işlenmesini sağladıkları ve programlamadaki daha karmaşık birçok yapının temelini oluşturdukları için günümüzdeki hemen hemen tüm programlama çözümlerinde kullanılmaktadır.

C++ programlama dilinde dizi, veri türü, dizi adı ve içereceği eleman sayısı belirtilerek tanımlanır. Genel biçimi şöyledir:

```
tip diziAdi[boyut];
Örneğin: int notlar[5];
```

Bu örnekte int anahtar sözcüğü, dizide tam sayıların saklanacağını gösterir. Dizinin adı notlar, 5 sayısı ise dizinin beş eleman içerdiğini belirtir. Dizi beş elemana sahip olmasına rağmen elemanların konumları 1'den değil, 0'dan başlar. Buna göre birinci eleman 0. konumda, ikinci eleman 1. konumda, üçüncü eleman 2. konumda, dördüncü eleman 3. konumda ve beşinci eleman 4. konumda bulunur. Genel kural olarak, boyutu n olan bir dizinin ilk elemanı 0. konumda, son elemanı ise n - 1 konumunda bulunur.

Dizi:

```
int notlar[5];
Elemanların konumları:
notlar[0]  notlar[1]  notlar[2]  notlar[3]  notlar[4]
```

Bu nedenle elemanlara erişirken izin verilen son konumdan daha büyük bir konum kullanılmamasına her zaman dikkat edilmelidir. Aksi hâlde programın hatalı çalışmasına neden olabilir.

Örnek 1: Dizi kullanmadan problem

Beş öğrencinin notlarını gösteriniz.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int   not1   = 5;
    int   not2   = 4;
    int   not3   = 5;
    int   not4   = 3;
    int   not5   = 5;

    cout <<   not1   << endl;
    cout <<   not2   << endl;
    cout <<   not3   << endl;
    cout <<   not4   << endl;
    cout <<   not5   << endl;

    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
5
4
5
3
5

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

Örnekte her not için ayrı bir değişken kullanılmıştır. Veri sayısı az olduğunda bu çözüm kabul edilebilir, ancak onlarca veya yüzlerce değerin işlenmesi gerektiğinde pratik olmaktan çıkar.

Birden fazla veri aynı türe ve aynı amaca sahipse, bunlar bir dizi hâlinde düzenlenebilir. Çok sayıda farklı değişken adı kullanmak yerine, tüm değerler tek bir ortak ad altında gruplanır. Bu sayede program daha kısa ve daha düzenli hâle gelir. Ayrıca veriler, döngüler kullanılarak kolayca işlenebilir. Bu özellik, dizilerin en büyük avantajlarından birini oluşturur ve programlamada yaygın olarak kullanılmalarının temel nedenlerinden biridir.

Örnek 2: Aynı verilerin bir dizi hâlinde düzenlenmesi

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int notlar[5] = {5, 4, 5, 3, 5};
8
9      cout << notlar [0] << endl;
10     cout << notlar [1] << endl;
11     cout << notlar [2] << endl;
12     cout << notlar [3] << endl;
13     cout << notlar [4] << endl;
14
15     return 0;
16 }
17
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
5
4
5
3
5

Process returned 0 (0x0)   execution time : 1.675 s
Press any key to continue.
```

Örnekte tüm notlar notlar adlı tek bir dizide saklanmıştır. Bu sayede birden fazla değer tek bir ortak ad altında düzenlenmiş olur ve program daha basit ve bakımı daha kolay hâle gelir.

Bir dizinin elemanları ardışık bellek konumlarında saklanır. Bu, bilgisayarın elemanları bellekte yan yana tuttuğu anlamına gelir. Bu düzenleme sayesinde elemanlara erişim çok hızlı ve verimli olur. Tüm elemanlar ortak

bir ad ile ilişkilendirilmiş olsa da her biri bağımsız olarak okunabilen veya değiştirilebilen ayrı bir değerdir. Verilerin bu şekilde düzenlenmesi, çok sayıda değerin tek bir bütün olarak işlenmesini sağlar. Tam da bu nedenle diziler programlamada önemli bir role sahiptir ve birçok daha karmaşık veri yapısının temelini oluşturur. Dizilerle çalışırken tüm elemanların aynı türde olması gerektiği her zaman dikkate alınmalıdır. Aynı dizi içinde aynı anda tam sayılar, gerçek sayılar ve metinsel değerler saklanmasına izin verilmez.

Örnek 3: Müzeye Gelen Ziyaretçi Sayısı

Beş gün boyunca bir müzedeki ziyaretçi sayısı kaydedilmiştir. Veriler bir dizide saklanmalı ve ekranda gösterilmelidir.

```
#include <iostream>
using namespace std;
int main()
 {
int ziyaretciler[5] = {120, 145, 132, 160, 151};

 cout << «1.Gün:» << ziyaretciler[0] << endl;
cout << «2.Gün:» << ziyaretciler[1] << endl;
 cout << «3.Gün:» << ziyaretciler[2] << endl;
 cout << «4.Gün:» << ziyaretciler[3] << endl;
cout << «5.Gün:» << ziyaretciler[4] << endl;

return 0;
}
```

Örnekte beş günün tüm verileri tek bir dizide saklanmıştır. Beş farklı değişken kullanmak yerine tüm değerler tek bir ortak ad altında düzenlenmiştir. Bu şekilde program daha düzenli ve bakımı daha kolay hâle gelir.

Diziler, aynı türden çok sayıda verinin işlenmesi gerektiğinde kullanılır. Notların, sıcaklıkların, finansal verilerin, spor sonuçlarının, sensörlerden alınan ölçümlerin ve daha birçok bilginin saklanmasında kullanılabilirler. Diziler olmadan daha büyük programlama çözümlerinin geliştirilmesi önemli ölçüde daha zor olurdu. Bu nedenle dizilerin öğrenilmesi, programlama becerilerinin geliştirilmesinde önemli bir adımdır. Temel kavram öğrenildikten sonra dizilerin elemanlarının girilmesi, işlenmesi ve analiz edilmesine geçilebilir. İşte bu işlemler, uygulamada kullanılan çok sayıda algoritmanın ve programlama çözümünün temelini oluşturur.

SORULAR VE TEKRARLAMA ALIŞTIRMALARI

SORULAR

1. Tek boyutlu dizi nedir?
2. Birden fazla ayrı değişken yerine dizi kullanmak ne zaman uygundur?
3. Bir dizi hangi tür verileri içerebilir?
4. Dizinin elemanları bellekte nasıl düzenlenir?
5. Dizileri kullanmanın avantajları nelerdir?

ÖDEVLER

1. Bir kütüphanede yedi gün boyunca ödünç alınan kitapların sayıları saklanmalıdır. Verileri bir diziye yerleştirip ekranda gösteren bir program yazınız.
2. Bir spor kulübünde beş müsabakada kazanılan puanlar kaydedilmektedir. Verileri bir diziye kaydeden ve ekranda gösteren bir program yazınız.
3. Bir meteoroloji istasyonunda altı günlük sıcaklık değerleri kaydedilmektedir. Sıcaklıkları bir diziye kaydeden ve ekrana yazdıran bir program yazınız.
4. Bir mağazada dört ürünün satışları takip edilmektedir. Miktarları bir diziye kaydeden ve ekranda gösteren bir program yazınız.
5. Bir okulda beş sınıftaki öğrenci sayıları kaydedilmektedir. Verileri bir diziye kaydeden ve ekranda gösteren bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Dizi, aynı türdeki elemanlardan oluşan bir bütündür.
- Dizinin tüm elemanlarının ortak bir adı vardır.
- Elemanlar ardışık bellek konumlarında saklanır.
- Diziler, çok sayıda verinin düzenli bir şekilde saklanmasını sağlar.
- Diziler, programlamadaki en önemli veri yapılarından biridir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilgisayar sistemlerinde çok büyük miktardaki veriler, dizileri temel alan yapılar kullanılarak düzenlenir. Sosyal ağlardaki kullanıcı listeleri, internet'teki arama sonuçları, bilimsel deneylerden elde edilen veriler ve istatistiksel analizler çoğu zaman basit dizilerle başlar. İlk bakışta basit bir yapı gibi görünseler de diziler, modern programlamada ve yazılım sistemlerinin geliştirilmesinde kullanılan birçok daha karmaşık yapının temelini oluşturur.

2.15

TEK BOYUTLU DIZILERIN TANIMLANMASI VE BAŞLATILMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu dizinin ne olduğunu açıklamayı;
- Dizi kullanımının gerekli olduğu durumları belirlemeyi;
- Değişken ile dizi arasındaki farkı ayırt etmeyi;
- Dizinin boyutunun ne olduğunu açıklamayı;
- Dizi elemanlarının 0'dan başlayarak numaralandırıldığını bileceksiniz.

TEKRARLAMA SORULARI

1. Tek boyutlu dizi nedir?
2. Çok sayıda ayrı değişken yerine dizi kullanmak neden daha iyidir?
3. Dizi tanımlanırken köşeli parantez içindeki sayı neyi belirtir?
4. Bir dizinin ilk elemanının konumu nedir?
5. Boyutu 8 olan bir dizinin son elemanının konumu nedir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Tek boyutlu dizileri tanımlamayı;
- Dizileri tanımlarken başlangıç değerleri vermeyi;
- Tanımlama ile başlatma arasındaki farkı ayırt etmeyi;
- Dizileri başlatmanın farklı yöntemlerini kullanmayı;
- Dizilerle çalışırken yapılan en yaygın hataları fark etmeyi.

Veriler bir diziye yerleştirilmeden önce, öncelikle bilgisayarın belleğinde alan oluşturulması gerekir. Bu işlem dizinin tanımlanması (deklarasyonu) olarak adlandırılır. Tanımlama sırasında saklanacak verilerin türü ve dizinin içereceği eleman sayısı belirlenir.

Tanımlamadan sonra dizinin elemanlarına başlangıç değerleri verilebilir. Bu işlem başlatma (initialization) olarak adlandırılır. C++ programlama dilinde dizileri başlatmanın birden fazla yolu vardır ve doğru yöntemin seçilmesi, verilerin daha anlaşılır ve daha kolay kullanılmasına katkı sağlar.

Bir diziyi tanımlamak, belirli sayıda aynı türden eleman için bellekte alan ayırma işlemidir. Tanımlama sırasında veri türü, dizinin adı ve dizinin boyutu mutlaka belirtilmelidir. Boyut, dizinin kaç eleman içerebileceğini belirler. Dizi tanımlandıktan sonra verileri saklamak ve işlemek için kullanılabilir. Tanımlama sırasında elemanlara değer verilmezse, bu elemanlar belirsiz değerlere sahip olur. Bu nedenle birçok durumda dizinin hemen başlatılması önerilir. Böylece program daha anlaşılır hâle gelir ve hata oluşma ihtimali önemli ölçüde azalır.

Örnek 1: Dizi tanımlama

*main.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int yillar[6];
8
9       return 0;
10  }
11
```

Örnekte yıllar adlı bir dizi tanımlanmıştır. Bu diziye altı tam sayı yerleştirilebilir. Bu aşamada bellekte alan ayrılmıştır, ancak elemanlara henüz herhangi bir değer verilmemiştir.

Değerler önceden biliniyorsa, bu değerler dizi tanımlanırken de verilebilir. Bu işlem başlatma (initialization) olarak adlandırılır. Başlatma sırasında değerler süslü parantezler içine yazılır ve virgüllerle birbirinden ayrılır. Değerlerin sayısı genellikle dizinin boyutuna eşittir. Bu yöntem, programları test ederken veya veriler önceden bilindiğinde sıkça kullanılır. Başlatılmış diziler, programın hemen işlenebilecek ve analiz edilebilecek verilere sahip olmasını sağlar.

Örnek 2: Dizinin başlatılması

main.cpp

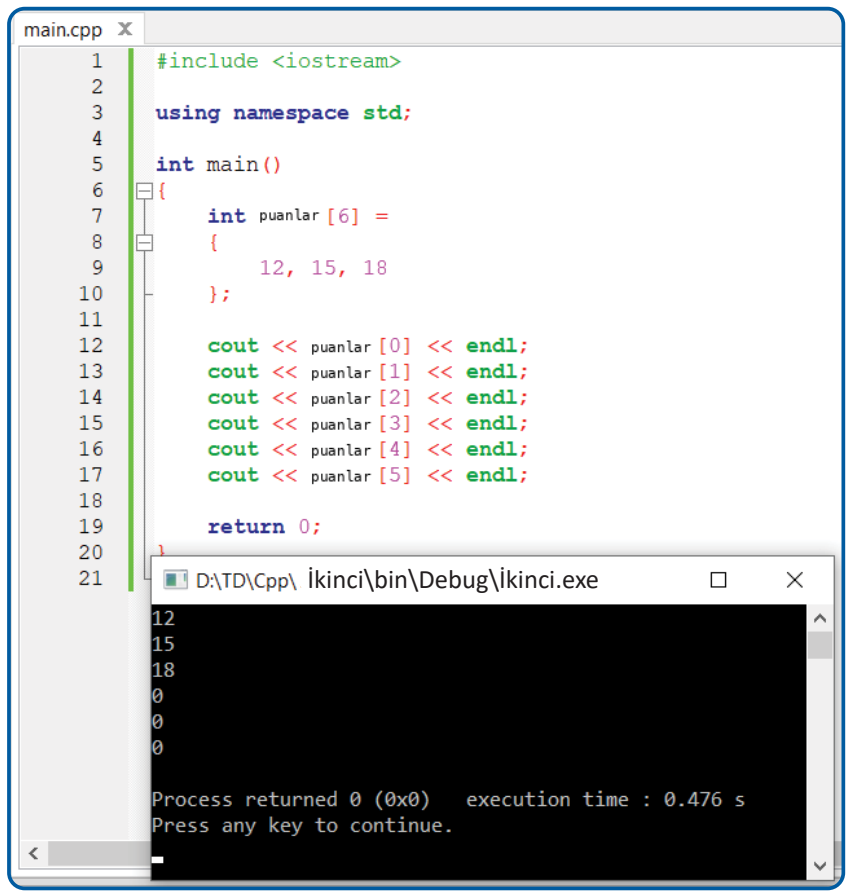
```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int sicakliklar[7] =
8       {
9           18, 20, 21, 19, 22, 24, 23
10      };
11
12      cout << sicakliklar[0] << endl;
13
14      return 0;
15  }
16
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
18

Process returned 0 (0x0)   execution time : 0.008 s
Press any key to continue.
```

Örnekte dizi, tanımlama sırasında başlatılmıştır. Her elemana bir başlangıç değeri verilmiş ve dizi hemen kullanıma hazır hâle gelmiştir.

Dizileri başlatırken tüm elemanlara değer verilmesi zorunlu değildir. Dizinin boyutundan daha az değer belirtilirse, ilk elemanlara belirtilen değerler atanır, kalan elemanlar ise 0 olarak ayarlanır. Bu başlatma yöntemine kısmi başlatma denir. Bu yaklaşım, dizinin oluşturulduğu anda verilerin yalnızca bir kısmının bilindiği durumlarda yararlı olabilir. Ancak bu tür dizilerle çalışırken hangi elemanların gerçek veriler içerdiği ve hangilerinin otomatik olarak değer aldığı dikkatlice kontrol edilmelidir. Başlatma yönteminin bilinmesi önemlidir, çünkü programın çalıştırılması sonucunda elde edilecek sonuçları doğrudan etkiler.

Örnek 3: Dizinin kısmi başlatılması



Örnekte ilk üç elemana 12, 15 ve 18 değerleri atanır. Kalan üç eleman ise otomatik olarak 0 değerini alır. Bu özellik sayesinde dizinin tüm elemanları için her zaman değer belirtmek gerekmez.

C++ programlama dilinde dizinin boyutunun, girilen değerlerin sayısına göre otomatik olarak belirlenmesi de mümkündür. Bu durumda köşeli parantezlerin içinde eleman sayısı belirtilmez. Derleyici, verilen değerlerin sayısına göre eleman sayısını kendisi hesaplar. Bu yaklaşım programın okunabilirliğini artırır ve elemanları sayarken yapılabilecek hataların olasılığını azaltır. Ancak bu yöntem yalnızca dizinin tanımlandığı anda tüm değerlerin bilindiği durumlarda uygundur. Veriler daha sonra girilecekse, dizinin boyutu önceden belirlenmelidir. Bu nedenle dizi tanımlama yönteminin seçimi, çözülen probleme ve dizinin oluşturulduğu anda verilerin mevcut olup olmamasına bağlıdır.

```
    Boyutun otomatik olarak belirlenmesi
    int aylar[] =
{
   31, 28, 31, 30,
   31, 30, 31, 31,
   30, 31, 30, 31
};
```

Bu örnekte dizinin boyutu belirtilmemiştir. On iki değer girildiği için derleyici otomatik olarak on iki elemanlı bir dizi oluşturur.

SORULAR VE TEKRARLAMA ALIŞTIRMALARI

SORULAR

1. Dizi deklarasyonu nedir?
2. Dizi başlatma (initialization) nedir?
3. Deklarasyon ile başlatma (initialization) arasındaki fark nedir?
4. Kısmi başlatma sırasında değer verilmeyen elemanlara ne olur?
5. Dizinin boyutunun otomatik olarak belirlenmesi hangi durumlarda kullanılabilir?

ÖDEVLER

1. On tam sayıdan oluşan, başlatma yapılmamış bir dizinin tanımlandığı bir program yazınız.
2. Bir hafta boyunca ölçülen yedi sıcaklık değerinin bulunduğu bir dizinin başlatıldığı bir program yazınız.
3. Bir sporcunun beş müsabakadaki sonuçlarını temsil eden beş değerlik bir dizinin oluşturulduğu bir program yazınız.
4. Sekiz elemanlı, kısmen başlatılmış bir dizinin kullanılacağı bir program yazınız. Dizide yalnızca ilk dört değer belirtilsin.
5. Dizinin boyutunun, girilen değerlerin sayısına göre otomatik olarak belirleneceği bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Deklarasyon, dizinin elemanları için bellekte yer ayırır.
- Başlatma (initialization), elemanlara başlangıç değerleri verir.
- Dizinin boyutu, saklanabilecek eleman sayısını belirler.
- Kısmi başlatmada, kalan elemanlar 0 değerini alır.
- Dizinin boyutu, deklarasyon sırasında tüm değerler belirtilmişse otomatik olarak belirlenebilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern programlama dillerinde, programın çalışması sırasında boyutunu otomatik olarak değiştirebilen yapılar bulunmaktadır. Bu tür yapılar, eleman sayısının önceden belirlenmesine gerek kalmadan eleman eklenmesine ve çıkarılmasına olanak sağlar. Bununla birlikte, klasik diziler veri depolamak için en hızlı ve en verimli yapılardan biri olmaya devam etmektedir. Basitlikleri ve verimlilikleri sayesinde, modern programlamada kullanılan birçok daha karmaşık yapının temelini oluştururlar.

2.16

DİZİ ELEMANININ İNDEKSİ VE DİZİ ELEMANLARINA ERİŞİM

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu dizinin ne olduğunu açıklamayı;
- Dizi tanımlamayı;
- Dizi başlatmayı;
- Tanımlama ile başlatma arasındaki farkı ayırt etmeyi;
- Dizi elemanlarının 0'dan başlayarak numaralandırıldığını bileceksiniz.

TEKRARLAMA SORULARI

1. Dizi tanımlama nedir?
2. Dizi başlatma (initialization) nedir?
3. int sayilar[8]; dizisi kaç eleman içerir?
4. Boyutu 10 olan bir dizinin son elemanının indeksi nedir?
5. Kısmi başlatma sırasında kalan elemanlara ne olur?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Eleman indeksinin ne olduğunu açıklamayı;
- Dizinin tek tek elemanlarına erişmeyi;
- Dizideki elemanların değerlerini değiştirmeyi;
- Geçersiz indeksleri fark etmeyi;
- İndekslerle çalışırken yapılan yaygın hatalardan kaçınmayı.

Bir dizinin tüm elemanları ortak bir adla ilişkilendirilmiş olsa da her elemana ayrı ayrı erişilebilmeli ve işlem yapılabilmelidir. Bunun için indeksler kullanılır. İndeks, elemanın dizideki konumunu belirler ve programın belirli bir değere erişmesini sağlar.

İndekslerin doğru kullanılması, dizilerle başarılı bir şekilde çalışmanın temelini oluşturur. Programlama hatalarının büyük bir bölümü tam olarak indekslerin yanlış kullanılmasından kaynaklanır. Bu nedenle elemanların nasıl numaralandırıldığının ve hangi değerlerin geçerli indeksler olduğunun dikkatlice anlaşılması gerekir.

Dizideki her elemanın indeks adı verilen bir konumu vardır. İndeks, üzerinde işlem yapılacak belirli elemanın belirlenmesini sağlar. C++ programlama dilinde numaralandırma 0'dan başlar. Bu nedenle ilk elemanın indeksi 0, ikinci elemanın indeksi 1, üçüncü elemanın indeksi 2 ve bu şekilde devam eder. Dizi beş eleman içeriyorsa son elemanın indeksi 5 değil, 4 olur. Bu kural, dizinin boyutu ne olursa olsun tüm diziler için geçerlidir. Elemanlara erişmek için dizinin adı ve köşeli parantezler içinde indeks belirtilir. Böylece program hangi elemanla işlem yapması gerektiğini tam olarak bilir.

Örnek 1: Dizinin Bir Elemanına Erişme

```
#include <iostream>

using namespace std;

int main()
{
   int puanlar[5] =
   {
      10, 15, 20, 25, 30
   };

   cout << puanlar[2] << endl;

   return 0;
}
```

Örnekte 2 indeksli eleman ekrana yazdırılır. Numaralandırma 0'dan başladığı için bu, dizinin üçüncü elemanıdır ve değeri 20'dir.

İndeksler yalnızca değerleri okumayı değil, dizideki elemanların değiştirilmesini de sağlar. Belirli bir elemanın değerini değiştirmek için o elemanın indeksi belirtilmeli ve ona yeni bir değer atanmalıdır. Bu şekilde tek tek elemanlar, dizideki diğer elemanları etkilemeden bağımsız olarak değiştirilebilir. Bu özellik özellikle veri işlemede önemlidir, çünkü tüm diziyi yeniden oluşturmaya gerek kalmadan belirli bilgilerin güncellenmesini sağlar.

Örnek 2: Dizideki bir değerin değiştirilmesi

```
    #include <iostream>

using namespace std;

int main()
{
   int stok[4] =
   {
      12, 18, 25, 30
   };

   stok[1] = 22;

   cout << stok[1] << endl;

   return 0;
}
```

Örnekte 1 indeksli elemanın değeri 18'den 22'ye değiştirilmiştir.

Dizilerle çalışırken özellikle geçerli indekslerin kullanılması çok önemlidir. Her elemana yalnızca dizinin sınırları içinde bulunan bir indeks aracılığıyla erişilebilir. Dizi beş eleman içeriyorsa geçerli indeksler: 0, 1, 2, 3 ve 4 olur. 5 veya daha büyük bir indeks kullanarak bir elemana erişmeye çalışmak hatadır. Aynı şekilde negatif indeksler de kullanılamaz. Bu tür hatalar programın yanlış çalışmasına veya beklenmeyen sonuçların ortaya çıkmasına neden olabilir. Bu nedenle kullanılan indeksin her zaman izin verilen aralıkta olup olmadığı dikkatlice kontrol edilmelidir. Bu kuralı anlamak, dizilerle başarılı bir şekilde çalışmanın en önemli koşullarından biridir.

Örnek 3: Birden fazla dizi elemanına erişme

Bir spor kulübünde beş müsabakada kazanılan puanlar kaydedilmiştir. İlk, üçüncü ve son sonucu ekrana yazdırınız.

```
    #include <iostream>

using namespace std;

int main()
{
   nt puanlar[5] =
   {
      12, 18, 15, 20, 25
   };

   cout << "İlk sonuç: "
        << puanlar[0]
        << endl;
```

```
    cout << "Üçüncü sonuç: "
         << puanlar[2]
         << endl;

    cout << "Son sonuç: "
         << puanlar[4]
         << endl;

    return 0;
}
```

Örnekte aynı dizinin üç farklı elemanına erişilmektedir. İlk elemanın indeksi 0, üçüncü elemanın indeksi 2, son elemanın indeksi ise 4'tür. Bu durum, elemanın dizideki konumunun günlük hayattaki sıra numarasıyla her zaman aynı olmadığını göstermektedir.

İndeksler, dizilerin elemanlarına erişmek için kullanılan temel mekanizmadır. İndeksler olmadan tek tek değerlerin okunması, değiştirilmesi veya işlenmesi mümkün olmazdı. Bu nedenle dizilerle çalışırken dizinin boyutunun ve izin verilen indeks aralığının bilinmesi gerekir. Tek tek elemanlara erişim öğrenildikten sonra, daha fazla verinin diziye girilmesi ve işlenmesine geçilebilir. Bu, tek boyutlu dizilerin öğrenilmesindeki bir sonraki adımdır.

SORULAR VE TEKRARLAMA ALIŞTIRMALARI

SORULAR

1. Eleman indeksi nedir?
2. Elemanların numaralandırılması hangi sayıdan başlar?
3. 12 elemanlı bir dizinin son elemanının indeksi nedir?
4. Bir dizinin elemanına nasıl erişilir?
5. Dizide yalnızca geçerli indekslerin kullanılması neden önemlidir?

ÖDEVLER

1. Bir kütüphanede beş günlük ziyaret kaydı tutulmaktadır. Öğrencilerin üyelik kartı numaraları: 35, 42, 38, 50 ve 47. Dizinin ilk ve son elemanını ekrana yazdıran bir program yazınız.
2. Bir mağazada altı ürünün satışları takip edilmektedir. Verileri bir diziye kaydeden ve dördüncü elemanı ekrana yazdıran bir program yazınız.
3. Bir spor turnuvasında bir takımın beş maçta kazandığı puanlar kaydedilmiştir. İkinci maçın değerini değiştiren ve yeni değeri ekrana yazdıran bir program yazınız.
4. Bir meteoroloji istasyonunda yedi sıcaklık değeri bir dizide saklanmaktadır. Birinci, üçüncü ve altıncı günün sıcaklıklarını ekrana yazdıran bir program yazınız.
5. Bir okul kayıt sisteminde beş sınıftaki öğrenci sayıları saklanmaktadır. Son elemanın değerini değiştiren ve sonucu ekrana yazdıran bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- İndeks, elemanın dizideki konumunu belirler.
- Elemanların numaralandırılması 0'dan başlar.
- Son elemanın indeksi, dizinin boyutunun 1 eksiğine eşittir.
- İndeksler, elemanları okumak ve değiştirmek için kullanılır.
- Yalnızca diziye ait geçerli indeksler kullanılmalıdır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

C++, Java ve C# gibi birçok programlama dilinde dizilerdeki elemanların numaralandırılması 0'dan başlar. Bu kural, bilgisayarın elemanların bellek adreslerini hesaplama biçiminden kaynaklanır. Başlangıçta alışılmadık görünse de bu numaralandırma biçimi, verilere erişilirken daha basit ve daha hızlı hesaplamalar yapılmasını sağlar. İşte bu nedenle 0'dan indeksleme, çok sayıda modern programlama dilinde standart olarak kabul edilmiştir.

2.17

DİZİ ELEMANLARININ GİRİLMESİ VE YAZDIRILMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu dizi tanımlamayı;
- Dizi başlatmayı;
- İndeks aracılığıyla dizi elemanlarına erişmeyi;
- Dizideki elemanların değerlerini değiştirmeyi;
- for döngüsünü kullanmayı bileceksiniz.

TEKRARLAMA SORULARI

1. Eleman indeksi nedir?
2. Dizideki elemanların numaralandırılması hangi sayıdan başlar?
3. 15 elemanlı bir dizinin son elemanının indeksi nedir?
4. Dizideki bir elemanın değeri nasıl değiştirilir?
5. İzin verilen aralığın dışındaki indeksler neden kullanılmamalıdır?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Diziye değerler girmeyi;
- Elemanları girmek için döngü kullanmayı;
- Dizi elemanlarını ekrana yazdırmayı;
- Dizileri ve döngüleri birlikte kullanmayı;
- Veri girişi sırasında yapılan yaygın hatalardan kaçınmayı.

Dizilerin en önemli avantajlarından biri, az sayıda program komutuyla çok sayıda verinin işlenebilmesidir. On, yirmi veya yüz değerin girilmesi gerektiğinde her değer için ayrı bir değişken kullanmak pratik değildir. Bunun yerine veriler bir dizi içinde saklanabilir.

Dizilerle çalışırken verilerin kullanıcı tarafından girilmesi çoğu zaman gereklidir. Bunun için aynı komutların tekrarlanarak birden fazla elemanın girilmesini sağlayan döngüler kullanılır. Benzer şekilde dizinin içeriği de döngü kullanılarak ekrana yazdırılabilir.

Diziye veri girerken her elemanın bir değer alması gerekir. Her eleman için ayrı bir komut yazmak yerine genellikle for döngüsü kullanılır. Döngünün sayacı, o anda işlenen elemanın konumunu temsil eder. Böylece dizide beş, on veya yüz değer bulunmasından bağımsız olarak tüm elemanların girilmesi kolayca sağlanabilir. Diziler ile döngülerin birlikte kullanılması, programlamadaki en önemli tekniklerden biridir. Çünkü çok sayıda verinin verimli bir şekilde işlenmesini sağlar.

Örnek 1: Diziye beş sayı girme

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int sayilari[5];
8
9      for(int i = 0; i < 5; i++)
10     {
11         cout << "Sayıyı girin: ";
12         cin >> sayilar[i];
13     }
14
15     return 0;
16 }
17
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Sayıyı girin: 1
Sayıyı girin: 2
Sayıyı girin: 5
Sayıyı girin: 3
Sayıyı girin: 4

Process returned 0 (0x0)   execution time : 5.555 s
Press any key to continue.
```

Örnekte döngü beş kez çalışır. Her çalıştırmada dizinin farklı bir elemanına bir değer girilir.

Veriler girildikten sonra çoğu zaman bunların ekranda gösterilmesi gerekir. Bu durumda da for döngüsü kullanılır. Her eleman için ayrı bir komut yazmak yerine döngü, dizideki tüm konumlara sırayla erişir. Bu yaklaşım programı daha kısa ve anlaşılır hâle getirir. Ayrıca aynı kod, önemli değişiklikler yapılmadan çok daha büyük diziler için de kullanılabilir.

Örnek 2: Dizinin elemanlarını ekrana yazdırma

main.cpp

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int sayilar[5] =
8       {
9           10, 20, 30, 40, 50
10      };
11
12      for(int i = 0; i < 5; i++)
13      {
14          cout << sayilar[i] << " ";
15      }
16
17      return 0;
18  }
19
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
10 20 30 40 50
Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

Örnekte dizinin tüm elemanları bir döngü kullanılarak ekrana yazdırılmaktadır. Döngü kullanılmasaydı, beş ayrı cout komutu yazılması gerekirdi.

Pratik programlama çözümlerinde genellikle yalnızca verilerin girilmesi yeterli değildir. Veri girişinden sonra çoğunlukla verilerin işlenmesi, kontrol edilmesi veya ekrana yazdırılması gerekir. Bu nedenle bir programda sıklıkla iki döngü kullanılır. İlk döngü verileri diziye girmek, ikinci döngü ise bu verileri ekrana yazdırmak için kullanılır. Bu yaklaşım, tüm verilerin önce bellekte saklanmasını, ardından farklı şekillerde işlenmesini sağlar. Eleman sayısı arttıkça bu yöntemin avantajları daha da belirgin hâle gelir. Onlarca veya yüzlerce veri için ayrı ayrı komutlar yazmak yerine yalnızca iki kısa döngü yeterlidir. Bu nedenle diziler ve döngülerin birlikte kullanılması, programlamada en sık kullanılan tekniklerden biridir.

Örnek 3: Ziyaretçi sayısının diziye girilmesi ve ekrana yazdırılması

Bir müzede beş gün boyunca ziyaretçi sayısı kaydedilmektedir. Verileri giriniz ve ardından ekrana yazdırınız.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int ziyaretciler[5];
    for(int i = 0; i < 5; i++)
    {
        cout << "Günlük ziyaretçi sayısı: "
             << i + 1
             << ": ";

        cin >> ziyaretciler[i];
    }
    cout << endl;
    cout << "Girilen veriler:" << endl;
    for(int i = 0; i < 5; i++)
    {
        cout << "Gün "
             << i + 1
             << ": "
             << ziyaretciler[i]
             << endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug

Günlük ziyaretçi sayısı: 1: 21
Günlük ziyaretçi sayısı: 2: 45
Günlük ziyaretçi sayısı: 3: 8
Günlük ziyaretçi sayısı: 4: 105
Günlük ziyaretçi sayısı: 5: 55
Girilen veriler:
Gün 1: 21
Gün 2: 45
Gün 3: 8
Gün 4: 105
Gün 5: 55


Process returned 0 (0x0)
Press any key to continue.
```

Örnekte ilk döngü verileri diziye girer, ikinci döngü ise verileri ekrana yazdırır. Bu şekilde tüm değerler, veri girişi tamamlandıktan sonra kolayca kontrol edilebilir.

Diziler ve döngülerle çalışırken özellikle tekrar sayısının doğru belirlenmesi çok önemlidir. Döngü, dizideki eleman sayısından daha fazla çalıştırılırsa var olmayan elemanlara erişilmeye çalışılır. Öte yandan, döngü daha az kez çalıştırılırsa bazı elemanlar işlenmez. Bu nedenle döngünün üst sınırı, dizinin boyutuyla uyumlu olmalıdır. Bu kural, ileride dizilerin elemanları üzerinde çeşitli hesaplamalar ve analizler yapmak için de kullanılacaktır.

TEKRARLAMA SORULARI VE ALIŞTIRMALAR

SORULAR

1. Dizi elemanlarını girerken neden genellikle döngü kullanılır?
2. Döngüde sayacın rolü nedir?
3. Tekrar sayısının dizinin boyutuna uygun olması neden önemlidir?
4. Elemanları ekrana yazdırmak için döngü kullanmanın avantajları nelerdir?
5. Diziler ve döngüler neden sıklıkla birlikte kullanılır?

ÖDEVLER

1. Bir arkeolojik kazı alanında farklı yaşlara sahip yedi nesne bulunmuştur. Yaşları yıl cinsinden ifade edilmektedir. Yaşları bir diziye girecek ve ardından giriş sırasına göre ekrana yazdıracak bir program yazınız.
2. Bir astronomi gözlemevinde on gece boyunca gözlemlenen meteor sayısı kaydedilmektedir. Verileri bir diziye girecek ve ardından tüm değerleri ekrana yazdıracak bir program yazınız.
3. Bir bisiklet fabrikasında altı çalışma vardiyası boyunca üretilen bisiklet sayısı kaydedilmektedir. Verileri bir diziye girecek ve ekrana yazdıracak bir program yazınız.
4. Bir milli parkta beş farklı turistik parkuru ziyaret eden kişi sayısı takip edilmektedir. Verileri bir diziye girecek ve ardından her parkur için bir mesajla ekrana yazdıracak bir program yazınız.
5. Bir uzay istasyonunda art arda sekiz gün boyunca tüketilen su miktarı (litre cinsinden) ölçülmektedir. Değerleri bir diziye girecek ve veri girişi tamamlandıktan sonra ekrana yazdıracak bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Dizi elemanlarının girilmesi genellikle for döngüsü ile yapılır.
- Döngünün sayacı, elemanların indeksi olarak kullanılır.
- Elemanların ekrana yazdırılması da döngü ile gerçekleştirilebilir.
- Dizinin boyutu, döngünün tekrar sayısıyla uyumlu olmalıdır.
- Diziler ve döngülerin birlikte kullanılması, çok sayıda verinin verimli bir şekilde işlenmesini sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern uygulamaların büyük bir bölümünde veriler, sensörler, internet hizmetleri, veritabanları veya mobil cihazlar gibi farklı kaynaklardan gelir. Kaynağı ne olursa olsun, bu veriler genellikle aynı anda birden fazla değerin işlenmesine olanak sağlayan yapılarda saklanır. Diziler, bu tür yapıların en basitlerinden biridir. Döngüler sayesinde binlerce veri, nispeten az sayıda program komutuyla işlenebilir. Bu özellik, istatistiksel analiz sistemlerinin, bilimsel araştırmaların, görüntü işlemenin, makine öğrenmesinin ve daha birçok modern bilişim alanının geliştirilmesinin temelini oluşturur.

2.18

DİZİ ELEMANLARIYLA TEMEL İŞLEMLER

BU KONUYU TAMAMLADIĞINIZDA...

- Dizi tanımlamayı ve başlatmayı;
- Dizi elemanlarını girmeyi ve ekrana yazdırmayı;
- İndeks aracılığıyla elemanlara erişmeyi;
- for döngüsünü kullanmayı;
- Dizileri ve döngüleri birlikte kullanmayı bileceksiniz.

TEKRARLAMA SORULARI

1. Eleman indeksi nedir?
2. Elemanların numaralandırılması hangi sayıdan başlar?
3. Dizilerle çalışırken neden for döngüsü kullanılır?
4. Bir diziye birden fazla eleman nasıl girilir?
5. Dizinin sınırları dışında bir indeks kullanılırsa ne olur?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ...

- Dizinin elemanlarının toplamını hesaplamayı;
- Dizinin elemanlarının çarpımını hesaplamayı;
- Verileri işlerken ek değişkenler kullanmayı;
- Aritmetik işlemleri ve dizileri birlikte kullanmayı;
- Tüm elemanların işlenmesini gerektiren pratik problemleri çözmeyi.

Birçok programlama çözümünde verilerin yalnızca girilmesi ve ekrana yazdırılması yeterli değildir. Çoğu zaman verilerin toplam değerinin, tüm değerlerin çarpımının veya başka bir ortak sonucun hesaplanması gerekir. Örneğin, üretilen toplam bal miktarının, toplam yolcu sayısının veya birden fazla etaptan oluşan bir yolculuğun toplam uzunluğunun hesaplanması gerekebilir.

Bu tür problemleri çözmek için genellikle dizinin tüm elemanlarını sırayla işleyen bir döngü kullanılır. Döngünün her tekrarında hesaplamanın sonucu güncellenir. Bu şekilde çok sayıda veri, nispeten az sayıda program komutuyla kolayca işlenebilir.

Bir dizinin elemanları işlenirken çoğu zaman bunların toplam değerinin hesaplanması gerekir. Bunun için başlangıçta 0 değerini alan ek bir değişken kullanılır. Daha sonra döngünün her tekrarında dizinin o anki elemanı bu değişkene eklenir. Son eleman da işlendiğinde, değişken dizideki tüm değerlerin toplamını içerir. Bu yöntem birçok pratik durumda kullanılabilir ve dizilerde veri işlemenin temel tekniklerinden biridir.

Örnek 1: Bal üretimi

Bir arıcılık işletmesinde beş kovandan üretilen bal miktarları ölçülmüştür. Toplam üretim miktarını hesaplayınız.

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int  bal  ={18,22,20,25,19};
8
9      int toplam=0;
10
11     for(int i=0;i<5;i++)
12     {
13         toplam = toplam + bal[i];
14     }
15     cout <<"Toplam üretim:  "<<toplam<<" kg";
16     return 0;
17 }
18
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Toplam üretim: 104 kg
Process returned 0 (0x0)   execution time : 0.164 s
Press any key to continue.
```

Örnekte üretilen tüm bal miktarları adım adım birbirine eklenir. Döngü tamamlandığında toplam bal üretimi elde edilir.

Her zaman toplam hesaplanması gerekmez. Bazı durumlarda birden fazla değerin çarpımının hesaplanması gerekir. Bu tür problemlerde başlangıçta 1 değerini alan ek bir değişken kullanılır. Bunun nedeni, bir sayının 1 ile çarpılmasının sayının değerini değiştirmemesi, 0 ile çarpılmasının ise her zaman sonucu 0 yapmasıdır. Döngünün her tekrarında, o anki dizi elemanı önceki sonuçla çarpılır.

Örnek 2: Toplam iletim katsayısı

Bir bilimsel deneyde beş adet yükseltme katsayısı ölçülmüştür. Bu katsayıların çarpımını hesaplayınız.

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    int carpan[5]={2,3,4,2,5};
    int  carpim  = 1;
    for(int i=0;i< 5;i++)
    {
        carpim = carpim * carpan[i];
    }
    cout << "Üretim: "
         << carpim ;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Üretim: 240
Process returned 0 (0x0)   execution time : 1.718 s
Press any key to continue.
```

Örnekte carpim değişkeni, dizinin her elemanı işlenirken adım adım güncellenir. Döngü tamamlandığında tüm değerlerin çarpımı elde edilir.

Bir dizide saklanan veriler işlenirken her zaman yalnızca toplamın veya çarpımın hesaplanması gerekmez. Birçok pratik durumda, her eleman üzerinde belirli bir hesaplama yapılması ve elde edilen sonuçların adım adım birleştirilmesi gerekir. Bu çalışma yöntemi, günlük yaşamda ve farklı bilim alanlarında çeşitli problemlerin çözülmesini sağlar. Hesaplamanın türü ne olursa olsun temel fikir aynıdır. Döngü, dizinin tüm elemanlarını sırayla

işler ve her elemana gerekli işlem uygulanır. Bu şekilde toplam tüketim, üretilen enerji, parasal tutar veya daha birçok farklı değer hesaplanabilir.

Örnek 3: Su tüketimi

Bir araştırma kampında beş seferin günlük su tüketimi kaydedilmektedir. Bir litre suyun fiyatı 2 denardır. Ödenmesi gereken toplam tutarı hesaplayınız.

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int  su [5]={120,150,110,140,130};
8      int toplam = 0;
9      for(int i=0;i<5;i++)
10     {
11         toplam+= su [i]*2;
12     }
13     cout << "Toplam tutar: " <<toplam<<"denar "<<endl;
14     return 0;
15 }
16
```

```
D:\TD\ İkinci\bin\Debug\İkinci.exe
Toplam tutar: 1300 denar
Process returned 0 (0x0)   execution time : 1.682 s
Press any key to continue.
```

Örnekte yalnızca dizideki değerler toplanmaz. Her eleman için önce fiyatı hesaplanır, ardından bu değer toplam tutara eklenir. Bu şekilde aritmetik işlemler ile dizi elemanlarının işlenmesi birleştirilmiş olur.

Diziler işlenirken gerçekleştirilecek işlemin seçimi, çözülen probleme bağlıdır. Bazı durumlarda toplam, bazı durumlarda çarpım, bazı durumlarda ise her eleman için belirli bir matematiksel ifade hesaplanır. Tüm bu problemlerin ortak özelliği, dizinin her elemanının tam olarak bir kez işlenmesidir. Bu sayede programlar, çok sayıda veriyle çalışsalar bile basit ve anlaşılır kalır. Bu tür veri işleme yöntemi, sonraki derslerde öğrenilecek birçok algoritmanın temelini oluşturur.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. Dizi işlenirken neden genellikle for döngüsü kullanılır?
2. Toplam değişkeni genellikle hangi başlangıç değeriyle, çarpım değişkeni ise hangi başlangıç değeriyle başlatılır?
3. Çarpım değişkeni neden çoğunlukla 1 değeriyle başlar?
4. Dizinin her elemanı üzerinde farklı bir aritmetik işlem gerçekleştirilebilir mi?
5. Tüm elemanların tek bir döngüyle işlenmesinin avantajı nedir?

ÖDEVLER

1. Bir rüzgâr santralinde altı rüzgâr türbini tarafından üretilen elektrik enerjisi miktarları ölçülmektedir. Toplam üretilen enerji miktarını hesaplayan bir program yazınız.
2. Bir fabrikada parçalar gruplar hâlinde paketlenmektedir. Bir dizide saklanan grup sayılarının çarpımını hesaplayan bir program yazınız.
3. Bir çevre kampanyasında beş yerleşim yerinde dikilen ağaçların sayısı kaydedilmektedir. Her ağaç için bakım amacıyla 50 denar ayrılmaktadır. Gerekli toplam tutarı hesaplayan bir program yazınız.
4. Bir robotik laboratuvarında dört robot tarafından başarıyla tamamlanan görevlerin sayısı kaydedilmektedir. Başarıyla tamamlanan toplam görev sayısını hesaplayan bir program yazınız.
5. Bir araştırma istasyonunda yedi gün boyunca toplanan örneklerin miktarı ölçülmektedir. Her örnek için 3 puan verilmektedir. Toplam kazanılan puanı hesaplayan bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Bir dizinin elemanları for döngüsü ile sırayla işlenebilir.
- Toplam değeri hesaplamak için genellikle bir toplam değişkeni kullanılır.
- Çarpımı hesaplamak için başlangıç değeri 1 olan bir değişken kullanılır.
- Dizinin her elemanına aritmetik bir işlem uygulanabilir.
- Tüm elemanların tek bir döngüyle işlenmesi, basit ve verimli programlama çözümleri sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilişim sistemlerinde çoğu zaman çok büyük miktarda veri işlenir. Örneğin sensörlerden alınan ölçümler, finansal işlemler, bilimsel deneyler veya istatistiksel veriler analiz edilebilir. Bu tür sistemler binlerce veya milyonlarca değerle çalışsa da temel fikir aynı kalır: veriler sırayla işlenir ve her eleman üzerinde gerekli işlemler gerçekleştirilir. Bu temel tekniklerin bilinmesi, daha karmaşık algoritmaların ve veri yapılarının öğrenilmesi için önemli bir temel oluşturur.

2.19

DİZİYE ELEMANLARIN GİRİLMESİ VE TEMEL HESAPLAMALAR

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu dizi tanımlamayı ve başlangıç değerlerini vermeyi;
- Dizinin elemanlarına erişmeyi;
- for döngüsünü kullanmayı;
- Önceden verilmiş elemanların toplamını ve çarpımını hesaplamayı;
- Dizi elemanlarını döngü ile işlemeyi bileceksiniz.

TEKRARLAMA ALIŞTIRMALARI

1. Tek boyutlu dizi nasıl tanımlanır?
2. Elemanların numaralandırılması hangi sayıdan başlar?
3. Dizilerle çalışırken neden genellikle for döngüsü kullanılır?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Bir diziye eleman girmeyi;
- Girilen verileri işlemeyi;
- Girilen değerlerin toplamını hesaplamayı;
- Girilen değerlerin çarpımını hesaplamayı;
- Girilen verilerin aritmetik ortalamasını hesaplamayı.

Birçok programlama çözümünde veriler önceden bilinmez. Program yazılırken programa eklenmek yerine, program çalıştırıldığı sırada girilir. Böylece aynı program, farklı veri kümelerini işlemek için kullanılabilir.

Dizilerle çalışırken veriler genellikle klavyeden bir döngü yardımıyla girilir. Veriler girildikten sonra bunlar üzerinde toplam, çarpım veya aritmetik ortalama gibi çeşitli hesaplamalar yapılabilir. Bu çalışma yöntemi, birçok pratik programlama probleminin temelini oluşturur.

Bir dizinin elemanlarını girerken genellikle for döngüsü kullanılır. Döngünün sayacı, o anda değer girilen elemanın konumunu belirler. Tüm değerler girildikten sonra bunlar ikinci bir döngü ile işlenebilir veya veri girişi sırasında doğrudan işlenebilir. Bu özellik sayesinde çok sayıda veri hızlı ve verimli bir şekilde işlenebilir. Bu yaklaşım, çeşitli

bilgi sistemlerinde geniş bir kullanım alanına sahiptir ve tek boyutlu dizilerle çalışmanın temel tekniklerinden biridir.

Örnek 1: Bisiklet yollarının toplam uzunluğu

Bir milli parkta beş bisiklet yolunun uzunlukları girilmektedir. Bu yolların toplam uzunluğunu hesaplayınız.

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int parkurlar[5];
8      int toplam = 0;
9
10     for(int i = 0; i < 5; i++)
11     {
12         cout << "Parkur uzunluğu  " <<i+1<< ": ";
13         cin >> parkurlar[i];
14         toplam += parkurlar[i];
15     }
16     cout << "Toplam uzunluk: " <<toplam<<" km"<< endl;
17     return 0;
18 }
19
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Parkur uzunluğu 1: 14
Parkur uzunluğu 2: 7
Parkur uzunluğu 3: 2
Parkur uzunluğu 4: 9
Parkur uzunluğu 5: 18
Toplam uzunluk: 50 km

Process returned 0 (0x0)   execution time : 8.285 s
Press any key to continue.
```

Örnekte veriler klavyeden girilir ve her yeni veri girildiğinde bu değer toplam değere eklenir. Döngü tamamlandığında tüm yolların toplam uzunluğu elde edilir.

Pratik programlama çözümlerinde her zaman toplam hesaplamak gerekmez. Bazı durumlarda birden fazla değerin çarpımının hesaplanması gerekir. Bu tür durumlarda başlangıçta 1 değerini alan ek bir değişken kullanılır. Her yeni veri girildiğinde, bu değer önceki sonuçla çarpılır. Böylece girilen tüm elemanların çarpımı adım adım elde edilir.

Örnek 2: Katsayıların çarpımı

Bir bilim laboratuvarında beş adet yükseltme katsayısı girilmektedir. Bu katsayıların çarpımını hesaplayınız.

main.cpp

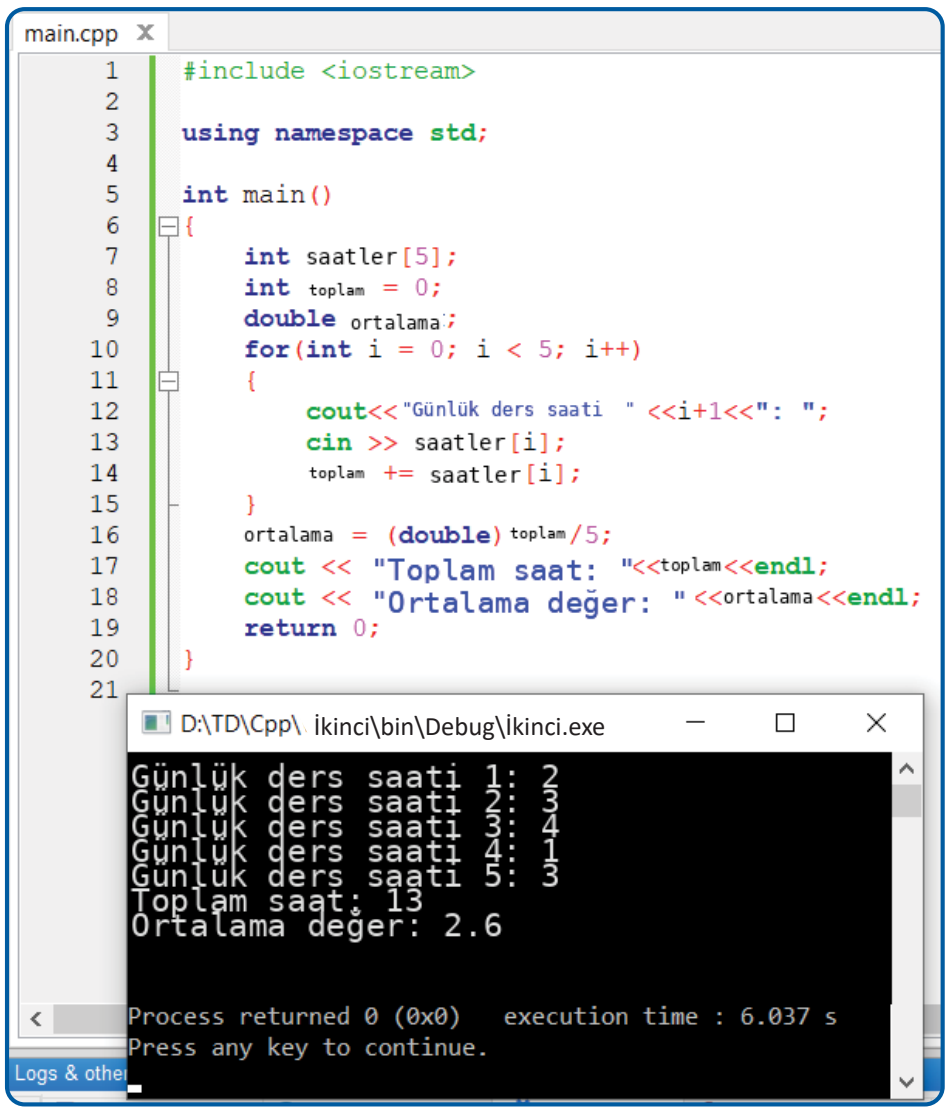
```
#include <iostream>

using namespace std;

int main()
{
    int carpan[5];
    int  carpim  = 1;
    for(int i = 0; i < 5; i++)
    {
        cout<<"Çarpan  "<<i+1<<": ";
        cin>> carpan[i];
        carpim *=carpan[i];
    }
    cout<<"Çarpım:  "<< carpim << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Çarpan 1: 2
Çarpan 2: 1
Çarpan 3: 2
Çarpan 4: 4
Çarpan 5: 3
Çarpım:  48

Process returned 0 (0x0)   execution time : 5.649 s
Press any key to continue.
```

Girilen veriler işlenirken çoğu zaman bunların ortalama değerinin hesaplanması gerekir. Bunun için önce dizideki tüm elemanların toplamı hesaplanır. Daha sonra elde edilen toplam, girilen elemanların sayısına bölünür. Bölme işleminin sonucu ondalık kısım içerebileceğinden, aritmetik ortalama hesaplanırken genellikle double türünde bir değişken kullanılır. Böylece daha hassas bir sonuç elde edilir ve ondalık değerlerin kaybolması önlenir. Bu yöntem, istatistiksel analizlerde, bilimsel araştırmalarda ve farklı türdeki verilerin işlenmesinde yaygın olarak kullanılmaktadır.

Örnek 3: Günlük ortalama güneşlenme süresi

Bir astronomi gözlemevinde beş gün boyunca güneş ışığı alınan saat sayısı kaydedilmektedir. Toplam saat sayısını ve günlük ortalama güneşlenme süresini hesaplayınız.



Örnekte veriler önce diziye girilir. Veri girişi sırasında toplamları adım adım hesaplanır. Döngü tamamlandıktan sonra toplam değer, eleman sayısına bölünerek aritmetik ortalama bulunur. Sonucu saklamak için double türünde bir değişken kullanılır. Böylece ondalık kısım da gösterilebilir.

Kullanıcı tarafından girilen veriler işlenirken, verilen probleme bağlı olarak farklı hesaplamalar yapılabilir. Bazı durumlarda toplam, bazı durumlarda çarpım, bazı durumlarda ise aritmetik ortalama hesaplanır. Bu problemlerin ortak özelliği, verilerin önce diziye girilmesi, ardından bir döngü yardımıyla işlenmesidir. Bu yöntem sayesinde aynı program, programda herhangi bir değişiklik yapılmasına gerek kalmadan farklı veri kümelerini işlemek için kullanılabilir. Bu yaklaşım, sonraki derslerde öğrenilecek birçok programlama çözümünün temelini oluşturur.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. Bir diziye veri girerken for döngüsü kullanmanın avantajı nedir?
2. Aritmetik ortalama hesaplanırken neden genellikle double veri türü kullanılır?
3. Birden fazla eleman girerken neden değerleri önceden tanımlamak yerine klavyeden veri girişi yapılır?

ÖDEVLER

1. Bir botanik bahçesinde yeni dikilmiş altı ağacın boyları girilmektedir. Toplam boylarını hesaplayan bir program yazınız.
2. Bir robotik merkezinde beş robotun başarıyla tamamladığı görev sayıları girilmektedir. Girilen değerlerin çarpımını hesaplayan bir program yazınız.
3. Bir oşinografi istasyonunda dalgaların yüksekliğine ilişkin beş günlük ölçüm girilmektedir. Aritmetik ortalamalarını hesaplayan bir program yazınız.
4. Bir arkeoloji gezisinde yedi farklı bölgede bulunan eserlerin sayısı girilmektedir. Toplam eser sayısını hesaplayan bir program yazınız.
5. Bir yenilenebilir enerji merkezinde beş güneş enerjisi sisteminin ürettiği enerji miktarı girilmektedir. Toplam ve ortalama üretilen enerji miktarını hesaplayan bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Dizinin elemanları for döngüsü kullanılarak klavyeden girilebilir.
- Girilen veriler hemen işlenebilir.
- Toplam değeri hesaplamak için bir toplam değişkeni kullanılır.
- Çarpımı hesaplamak için başlangıç değeri 1 olan bir değişken kullanılır.
- Aritmetik ortalama, toplamın eleman sayısına bölünmesiyle elde edilir.
- Ortalama hesaplanırken genellikle double veri türü kullanılır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Günümüz bilgi sistemlerinde verilerin büyük bir kısmı önceden bilinmez, veriler programın çalışması sırasında girilir. Örneğin sensörlerden alınan ölçümler, finansal işlemler, tıbbi analizler ve spor sonuçları sürekli olarak yeni bilgilerle güncellenir. Bu nedenle programların kullanıcıdan veri alabilmesi ve bu verileri hemen işleyebilmesi gerekir. İşte bu özellik, tek boyutlu dizilerin ve döngülerin günümüz programlamasındaki en önemli kullanım alanlarından biridir.

2.20

DİZİDEKİ EN BÜYÜK VE EN KÜÇÜK ELEMANIN BELİRLENMESİ

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu bir diziyi tanımlamayı ve başlangıç değerlerini vermeyi;
- Diziye eleman girmeyi;
- Verileri for döngüsü ile işlemeyi;
- Toplamı, çarpımı ve aritmetik ortalamayı hesaplamayı;
- Veri girişi ile veri işlemeyi birleştirmeyi bileceksiniz.

TEKRARLAMA ALIŞTIRMALARI

1. Bir diziye birden fazla eleman nasıl girilir?
2. Dizilerle çalışırken neden genellikle for döngüsü kullanılır?
3. Toplam değişkeni hangi başlangıç değeriyle başlatılır?
4. Aritmetik ortalama nasıl hesaplanır?
5. Verilerin girildikten hemen sonra işlenmesi neden yararlıdır?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Dizideki en büyük elemanı belirlemeyi;
- Dizideki en küçük elemanı belirlemeyi;
- Dizi elemanlarını karşılaştırmayı;
- Karşılaştırma sırasında ek değişkenler kullanmayı;
- Koşullu yapıları kullanarak verileri işlemeyi.

Verileri işlerken çoğu zaman belirli bir veri kümesindeki en büyük veya en küçük değerin belirlenmesi gerekir. Örneğin ölçülen en yüksek değerin, en düşük yağış miktarının veya ulaşılan en yüksek hızın belirlenmesi gerekebilir.

Bu tür problemlerin çözümünde dizi elemanlarının karşılaştırılması kullanılır. Öncelikle bir değer başlangıç değeri olarak seçilir, ardından diğer elemanlar sırayla bu değerle karşılaştırılır. Daha büyük veya daha küçük bir değer bulunursa, başlangıçta saklanan değer yeni değerle değiştirilir.

En büyük veya en küçük elemanı belirlerken öncelikle dizideki bir değer başlangıç değeri olarak seçilmelidir. Genellikle bu değer dizinin ilk elemanıdır. Daha sonra bir döngü yardımıyla diğer elemanlar sırayla incelenir. Elemanlardan biri o anda saklanan en büyük değerden daha büyükse, en büyük değer yeni değerle değiştirilir. Aynı yöntemle en küçük değer de belirlenebilir. Bu işlem sayesinde dizideki tüm veriler tek geçişte işlenebilir.

Örnek 1: En yüksek dağ zirvesi

Bir dağcılık rehberinde beş dağ zirvesinin deniz seviyesinden yükseklikleri girilmektedir. En yüksek zirveyi belirleyiniz.

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    int zirveler[5];
    int enBuyuk;
    for(int i=0;i<5;i++)
    {
        cout<<"Tepe yüksekliği  "<<i+1<<": ";
        cin >> zirveler[i];
    }
    enBuyuk = zirveler[0];
    for(int i=1;i<5;i++)
    {
        if (zirveler[i] > enBuyuk)
        {
            enBuyuk = zirveler[i];
        }
    }
    cout << "En yüksek tepe: " << enBuyuk << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Tepe yüksekliği 1: 845
Tepe yüksekliği 2: 2514
Tepe yüksekliği 3: 3254
Tepe yüksekliği 4: 1564
Tepe yüksekliği 5: 2455
En yüksek tepe: 3254

Press any key to continue.
```

Örnekte ilk eleman başlangıç değeri olarak seçilir. Daha sonra diğer tüm elemanlar bu değerle karşılaştırılır. Daha büyük bir değer bulunursa, bu değer yeni en büyük değer olur.

En küçük elemanı belirlerken de aynı yöntem kullanılır. Fark, daha büyük bir değer aranması yerine, mevcut elemanın o anda saklanan en küçük değerden daha küçük olup olmadığının kontrol edilmesidir. Bu şekilde, ek ve karmaşık hesaplamalara gerek kalmadan bir veri kümesindeki en küçük değer belirlenebilir.

Örnek 2: En az yakıt miktarı

Bir araştırma gezisinde altı tanktaki yakıt miktarları girilmektedir. En az yakıt bulunan tankı belirleyiniz.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int yakit[6];
    int enKucuk;
    for(int i=0;i<6;i++)
    {
        cout<<"Rezervuar  "<<i+1<<": ";
        cin >> yakit[i];
    }
    enKucuk = yakit[0];
    for(int i=1;i<6;i++)
    {
        if (yakit[i] < enKucuk)
        {
            enKucuk = yakit[i];
        }
    }
    cout << "En küçük miktar: " << enKucuk << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Rezervuar 1: 230
Rezervuar 2: 540
Rezervuar 3: 155
Rezervuar 4: 652
Rezervuar 5: 145
Rezervuar 6: 211
En küçük miktar: 145

Process returned 0 (0x0)   execution time : 15.757 s
Press any key to continue.
```

Örnekte aynı programlama fikri uygulanmaktadır, ancak bu kez girilen veriler arasından en küçük değer belirlenmektedir.

Pratik programlama problemlerinde çoğu zaman aynı veri kümesindeki en büyük ve en küçük değerin aynı anda belirlenmesi gerekir. Bunun için dizinin iki kez işlenmesine gerek yoktur. Elemanlar üzerinden yapılan tek bir geçiş sırasında hem en büyük hem de en küçük değer aynı anda güncellenebilir. İşlem tamamlandıktan sonra elde edilen sonuçlar üzerinde ek hesaplamalar da yapılabilir. Bunlardan biri, verilerin aralığının belirlenmesidir. Aralık, en büyük değer ile en küçük değer arasındaki fark olarak elde edilir.

Örnek 3: Ölçülen değerlerin aralığı

Bir bilim istasyonunda altı ölçüm değeri girilmektedir. Ölçümlerin en büyük değerini, en küçük değerini ve aralığını belirleyiniz.

main.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int olcumler[6];
    int enBuyuk;
    int enKucuk;
    int aralik;
    for(int i=0;i<6;i++)
    {
        cout<<"Ölçüm  "<<i+1<<": ";
        cin >> olcumler[i];
    }
    enBuyuk = olcumler[0];
    enKucuk = olcumler[0];

    for(int i=1;i<6;i++)
    {
        if (olcumler[i] > enBuyuk)
        {
            enBuyuk = olcumler[i];
        }
        if (olcumler[i] < enKucuk)
        {
            enKucuk = olcumler[i];
        }
    }
    aralik = enBuyuk - enKucuk;
    cout << "En büyük değer: " << enBuyuk << endl;
    cout << "En küçük değer: " << enKucuk << endl;
    cout << "Aralık: " << aralik << endl;
    return 0;
}
```

```
D:\TD...
Ölçüm 1: 55
Ölçüm 2: 24
Ölçüm 3: 68
Ölçüm 4: 144
Ölçüm 5: 65
Ölçüm 6: 42
En büyük değer: 144
En küçük değer: 24
Aralık: 120

Process returned 0 (0x0)
execution time : 14.949 s
Press any key to continue.
```

Örnekte, dizi üzerinde tek bir geçişte hem en büyük hem de en küçük değer belirlenmektedir. Daha sonra bu değerler kullanılarak verilerin aralığı hesaplanmaktadır.

Verilerin işlenmesinde en büyük ve en küçük değerlerin belirlenmesi geniş bir kullanım alanına sahiptir. Örneğin en yüksek sıcaklık, en düşük enerji tüketimi veya ölçülen en yüksek hız belirlenebilir. Verilerin türü ne olursa olsun, temel işlem aynıdır. Öncelikle bir başlangıç değeri seçilir, ardından diğer tüm elemanlar sırayla bu değerle karşılaştırılır ve gerektiğinde karşılaştırma sonucundaki değerler güncellenir.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. En büyük değer belirlenirken başlangıç değeri olarak neden genellikle dizinin ilk elemanı seçilir?
2. Dizideki en küçük eleman nasıl belirlenir?
3. En büyük ve en küçük değer tek bir dizi geçişiyle belirlenebilir mi?
4. Verilerin aralığı neyi ifade eder?
5. Bu tür problemlerde neden genellikle for döngüsü kullanılır?

ÖDEVLER

1. Meteoroloji istasyonunda beş dağ merkezinde ölçülen kar miktarları girilmektedir. Ölçülen en büyük miktarı belirleyen bir program yazınız.
2. Bir hayvan koruma merkezinde altı genç ayının ağırlıkları girilmektedir. Ölçülen en küçük ağırlığı belirleyen bir program yazınız.
3. Bir oşinografi araştırmasında yedi ölçümün derinlikleri girilmektedir. En büyük ve en küçük derinliği belirleyen bir program yazınız.
4. Hava kalitesinin araştırılmasında beş ölçüm istasyonundan günlük değerler girilmektedir. Ölçülen değerlerin aralığını hesaplayan bir program yazınız.
5. Bir arkeoloji araştırmasında bulunan altı eserin yaşları girilmektedir. En büyük yaşı, en küçük yaşı ve aralarındaki farkı belirleyen bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- En büyük ve en küçük değer, dizinin elemanları karşılaştırılarak belirlenebilir.
- Başlangıç değeri genellikle dizinin ilk elemanıdır.
- Daha büyük veya daha küçük bir değer bulunduğunda sonuç güncellenir.
- En büyük ve en küçük değer, dizi üzerinde tek bir geçiş yapılarak belirlenebilir.
- Verilerin aralığı, en büyük değer ile en küçük değer arasındaki fark olarak elde edilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

En büyük ve en küçük değerin belirlenmesi, veri analizinde en sık kullanılan işlemlerden biridir. Bilimde ölçülen en yüksek sıcaklık, tıpta belirli bir göstergenin ölçülen en yüksek değeri, sporda ise elde edilen en iyi sonuç belirlenebilir. Modern bilgi sistemleri çok büyük miktarda veri işlese de temel fikir aynıdır: Veriler sırayla karşılaştırılır ve daha büyük veya daha küçük bir değer bulunduğunda elde edilen sonuçlar güncellenir.

2.21

DİZİDEKİ ELEMANLARIN SAYILMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu bir diziye eleman girmeyi;
- Elemanları for döngüsü kullanarak işlemeyi;
- Toplam ve çarpım hesaplamayı;
- En büyük ve en küçük elemanı belirlemeyi;
- Verileri işlerken koşullu yapıları kullanmayı öğreneceksiniz.

TEKRARLAMA ALIŞTIRMALARI

1. Bir dizi işlenirken neden genellikle for döngüsü kullanılır?
2. Dizideki en büyük eleman nasıl belirlenir?
3. En küçük eleman nasıl belirlenir?
4. Birden fazla değer dizi üzerinde tek bir geçişte işlenebilir mi?
5. Verilerin aralığı neyi ifade eder?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Belirli bir koşulu sağlayan elemanları saymayı;
- Sayma işlemlerinde bir değişken kullanmayı;
- Döngüleri koşullu yapılarla birleştirmeyi;
- Dizileri işlerken mantıksal ifadeler kullanmayı;
- Verileri farklı ölçütlere göre analiz etmeyi.

Verilerin işlenmesi sırasında çoğu zaman yalnızca toplamlarının veya en büyük değerlerinin belirlenmesi yeterli değildir. Pek çok pratik durumda, belirli bir özelliğe sahip kaç eleman bulunduğunun belirlenmesi gerekir. Örneğin, yağışlı günlerin sayısının, izin verilen hız sınırını aşan araçların sayısının veya başarıyla gerçekleştirilen ölçümlerin sayısının belirlenmesi gerekebilir.

Bu tür problemlerin çözümünde sayma değişkeni kullanılır. Başlangıçta bu değişkene 0 değeri atanır. Her bir eleman işlenirken belirlenen koşulun sağlanıp sağlanmadığı kontrol edilir. Koşul sağlanıyorsa sayma değişkeninin değeri 1 artırılır. Tüm elemanlar işlendiğinde, bu değişken koşulu sağlayan elemanların toplam sayısını içerir.

Bir dizinin elemanlarını sayarken öncelikle başlangıç değeri 0 olan bir sayma değişkeni oluşturulur. Daha sonra tüm elemanlar bir döngü kullanılarak sırayla işlenir. Her bir elemanın belirlenen koşulu sağlayıp sağlamadığı kontrol edilir. Koşul sağlanıyorsa sayma değişkeninin değeri 1 artırılır. Son elemanın işlenmesinden sonra sayma değişkeni nihai sonucu içerir. Bu yöntem, farklı türdeki verilerin istatistiksel olarak işlenmesi ve analiz edilmesinde geniş bir kullanım alanına sahiptir.

Örnek 1: Uzun bitkiler

Bir botanik bahçesinde yedi bitkinin boy uzunlukları girilmektedir. 100 cm'den daha uzun olan kaç bitki bulunduğunu belirleyiniz.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int yukseklik[7];
    int sayi =0;
    for(int i=0;i<7;i++)
    {
        cout<<"Yükseklik "<<i+1<<": ";
        cin>>yukseklik[i];
        if(yukseklik[i] > 100)
        {
            sayi++;
        }
    }
    cout << "Bitki sayısı: "<<sayi<< endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Yükseklik 1: 105
Yükseklik 2: 92
Yükseklik 3: 56
Yükseklik 4: 213
Yükseklik 5: 84
Yükseklik 6: 155
Yükseklik 7: 142
Bitki sayısı: 4

Process returned 0 (0x0)   execution time : 13.936 s
Press any key to continue.
```

Örnekte girilen her bir değer kontrol edilir. Bitkinin boyu 100 cm'den uzunsa, sayma değişkeninin değeri 1 artırılır.

Pratik uygulamalarda koşulun değerlerin büyüklüğüyle ilişkili olması gerekmez. Çoğu zaman belirli bir matematiksel özelliğe sahip elemanların sayılması gerekir. En yaygın özelliklerden biri sayıların tek veya çift olmasıdır. Bu amaçla, bölme işlemindeki kalanı belirlemeyi sağlayan % operatörü kullanılır.

Örnek 2: Çift Sonuçlar

Bir spor merkezinde altı sonuç girilmektedir. Bu sonuçlardan kaç tanesinin çift sayı olduğunu belirleyiniz.

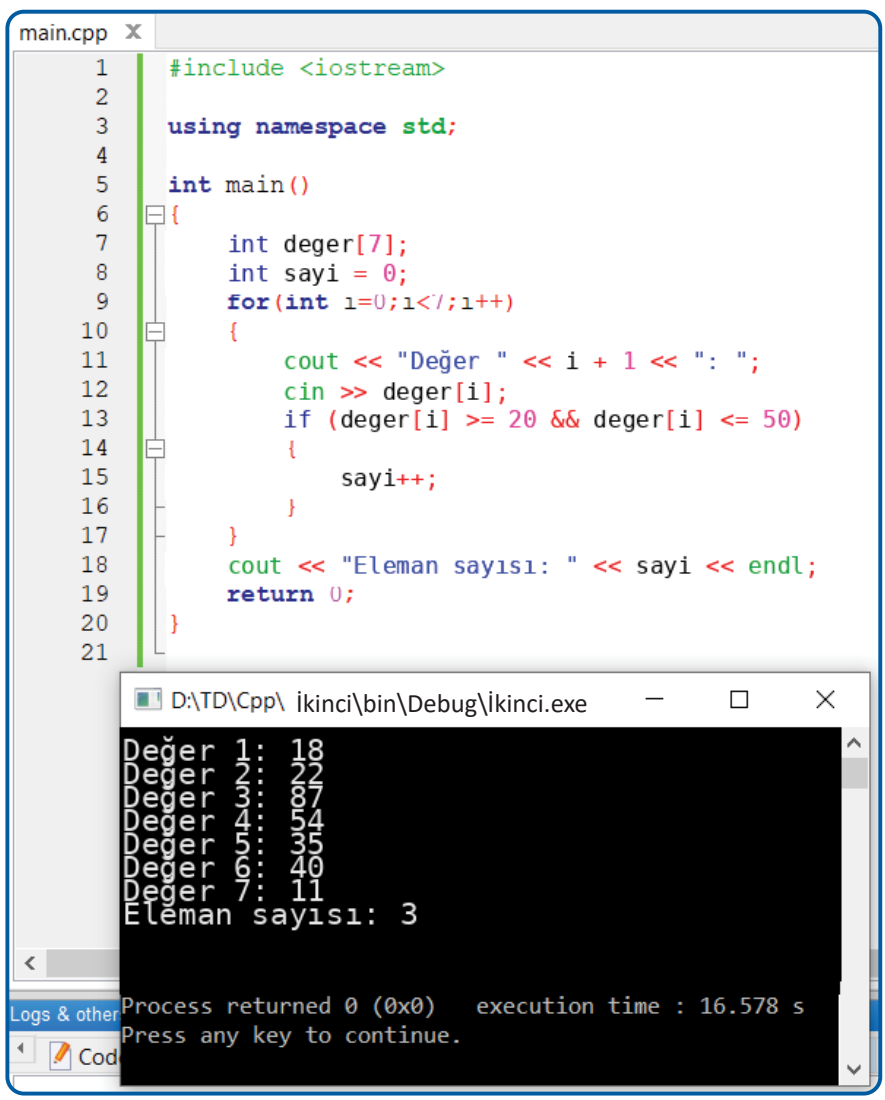
Örnekte girilen her bir değerin çift sayı olup olmadığı kontrol edilir. Koşul sağlanıyorsa çift sayıların sayısını tutan değişken 1 artırılır.

Verilerin işlenmesi sırasında çoğu zaman belirli bir aralıkta bulunan elemanların sayısının belirlenmesi gerekir. Örneğin, iki değer arasındaki sıcaklıkların sayısının, belirli bir aralıktaki notlara sahip öğrencilerin sayısının veya belirlenen koşulları sağlayan ürünlerin sayısının belirlenmesi gerekebilir. Bu tür durumlarda her bir eleman için

birden fazla koşul aynı anda kontrol edilir. Tüm koşullar sağlanıyorsa sayma değişkeninin değeri 1 artırılır. Bu şekilde, belirlenen aralığa ait elemanların sayısı kolayca belirlenebilir.

Örnek 3: Belirli bir aralıktaki değerler

Bir araştırma istasyonunda yedi ölçüm sonucu girilmektedir. Bu sonuçlardan kaç tanesinin 20 ile 50 arasındaki aralıkta bulunduğunu belirleyiniz.



Örnekte girilen her bir değerin belirlenen aralığa ait olup olmadığı kontrol edilir. Koşul sağlanıyorsa sayma değişkeninin değeri 1 artırılır. Tüm elemanlar işlendikten sonra nihai sonuç elde edilir.

Elemanları sayarken sağlanması gereken koşullar çok farklı olabilir. Bazı problemlerde değerin belirli bir sayıdan büyük olup olmadığı, bazılarında çift sayı olup olmadığı, bazılarında ise belirli bir aralığa ait olup olmadığı kontrol edilir. Problem ne olursa olsun, temel işlem aynı kalır. Elemanlar sırayla işlenir, her bir eleman için koşul kontrol edilir ve koşul sağlandığında sayma değişkeninin değeri artırılır. Bu teknik sayesinde çok sayıda pratik programlama problemi çözülebilir.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. Sayma değişkeni ne için kullanılır?
2. Sayma değişkeni genellikle hangi başlangıç değeriyle başlatılır?
3. Sayma değişkeninin değeri ne zaman artırılır?
4. Bir sayının çift olup olmadığı nasıl kontrol edilir?
5. Bir elemanın belirli bir aralığa ait olup olmadığı nasıl kontrol edilir?

ÖDEVLER

1. Bir milli parkta sekiz şelalenin yükseklikleri girilmektedir. 50 metreden daha yüksek olan kaç şelale bulunduğunu belirleyen bir program yazınız.
2. Bir robotik laboratuvarında altı robotun başarıyla tamamladığı görevlerin sayıları girilmektedir. Sonuçlardan kaç tanesinin tek sayı olduğunu belirleyen bir program yazınız.
3. Bir yenilenebilir enerji merkezinde yedi güneş enerjisi sisteminin günlük üretim miktarları girilmektedir. 100 ile 300 arasında bulunan kaç değer olduğunu belirleyen bir program yazınız.
4. Bir oşinografik araştırmada altı ölçümün derinlik değerleri girilmektedir. 500 metreden daha derin olan kaç ölçüm bulunduğunu belirleyen bir program yazınız.
5. Bir astronomi gözlemevinde beş gece boyunca gözlemlenen meteorların sayıları girilmektedir. 20'den fazla meteorun kaydedildiği kaç gece olduğunu belirleyen bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Elemanları saymak için bir sayma değişkeni kullanılır.
- Sayma değişkeni genellikle 0 değeriyle başlatılır.
- Belirlenen koşul sağlandığında sayma değişkeninin değeri artırılır.
- Koşul basit olabilir veya birden fazla kontrolden oluşabilir.
- Dizi üzerinde tek bir geçişle, belirli bir özelliğe sahip elemanların sayısı belirlenebilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Elemanların sayılması, veri analizinde en sık kullanılan tekniklerden biridir. Tıpta belirli bir belirtiye sahip hastaların sayısı, ekolojide yağışlı günlerin sayısı, sporda ise başarıyla gerçekleştirilen denemelerin sayısı belirlenebilir. Modern bilişim sistemleri çok büyük miktarda veriyi işlese de temel yaklaşım değişmez: Her veri kontrol edilir ve belirlenen koşulu sağlıyorsa sayma sonucuna eklenir.

2.22

KARAKTER DİZİLERİ VE KARAKTER ARAMA

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu tam sayı dizilerini tanımlayabilecek ve kullanabileceksiniz;
- Bir dizinin elemanlarını girebilecek ve işleyebileceksiniz;
- for döngüsünü kullanabileceksiniz;
- Koşullu yapıları uygulayabileceksiniz;
- Belirli bir koşulu sağlayan elemanları sayabileceksiniz.

TEKRARLAMA ALIŞTIRMALARI

1. Tek boyutlu dizi nedir?
2. Elemanların numaralandırılması hangi sayıdan başlar?
3. Dizilerle çalışırken neden for döngüsü kullanılır?
4. Sayma değişkeni ne için kullanılır?
5. Bir elemanın belirli bir koşulu sağlayıp sağlamadığı nasıl kontrol edilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Karakter dizilerini kullanmayı;
- Bir diziye karakterleri girmeyi;
- Bir dizide belirli bir karakteri aramayı;
- Bir karakterin dizide bulunup bulunmadığını belirlemeyi;
- Bir karakterin kaç kez tekrarlandığını saymayı.

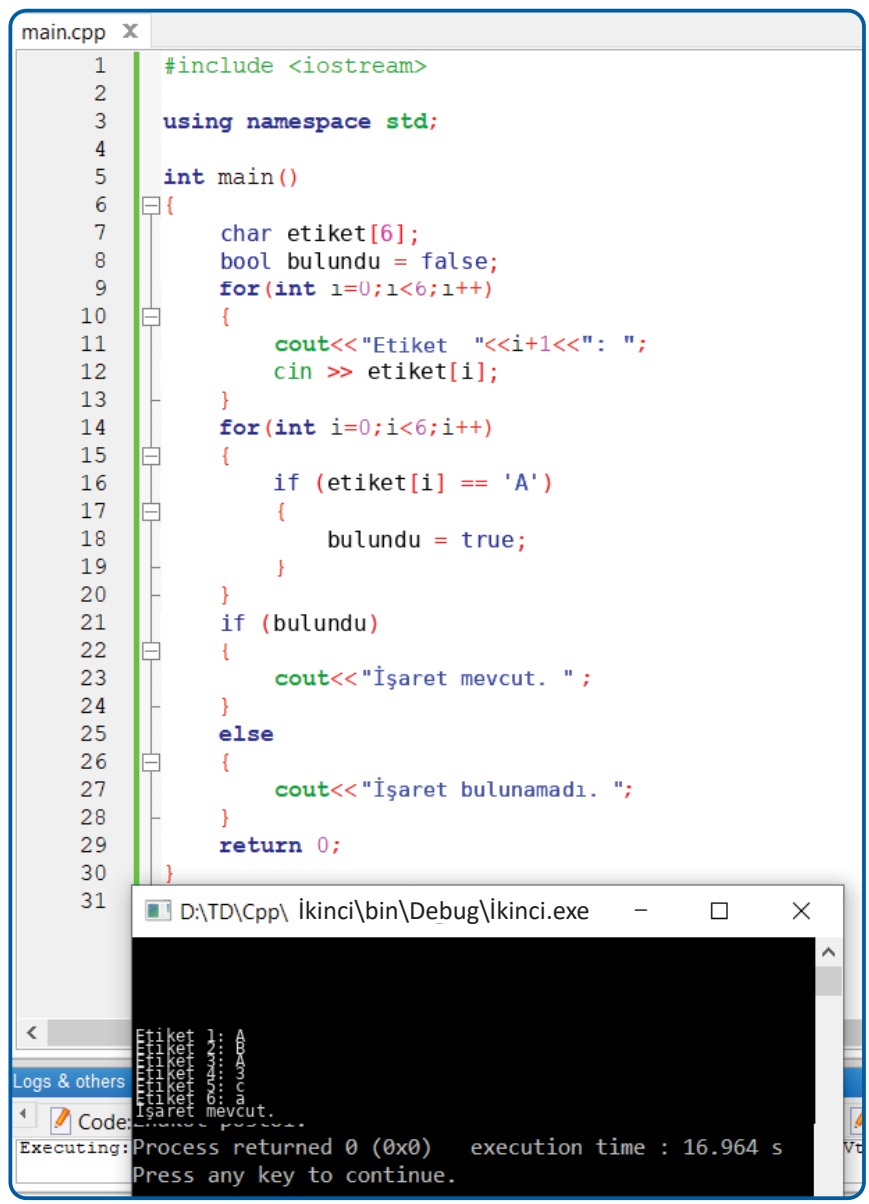
Sayıların yanı sıra programlarda karakterlerin de işlenmesine sıkça ihtiyaç duyulur. Örneğin, belirli bir harfin bir kodda bulunup bulunmadığının, belirli bir sembolün bir metinde geçip geçmediğinin veya belirli bir karakterin kaç kez kullanıldığının kontrol edilmesi gerekebilir.

C++ programlama dilinde tek tek karakterleri saklamak için char veri tipi kullanılır. Birden fazla karakter, bir karakter dizisinde saklanabilir. Böylece her karakter dizide kendine ait bir konuma sahip olur ve karakterler tek tek işlenebilir.

Karakter dizisi, tam sayı dizileriyle aynı şekilde tanımlanır. Aradaki fark, int veri tipi yerine char veri tipinin kullanılmasıdır. Dizinin her elemanı bir karakter içerir. Bu elemanlar harf, rakam veya başka semboller olabilir. Dizi işlenirken her karakter belirli bir değerle karşılaştırılabilir ve karşılaştırmanın sonucuna göre karar verilebilir. Bu özellik sayesinde metinsel verilerin işlenmesiyle ilgili çeşitli pratik problemler çözülebilir.

Örnek 1: Bir karakterin bulunup bulunmadığının kontrolü

Bir kontrol sistemine altı adet kod girilmektedir. Bunların arasında A harfinin bulunup bulunmadığını kontrol ediniz.

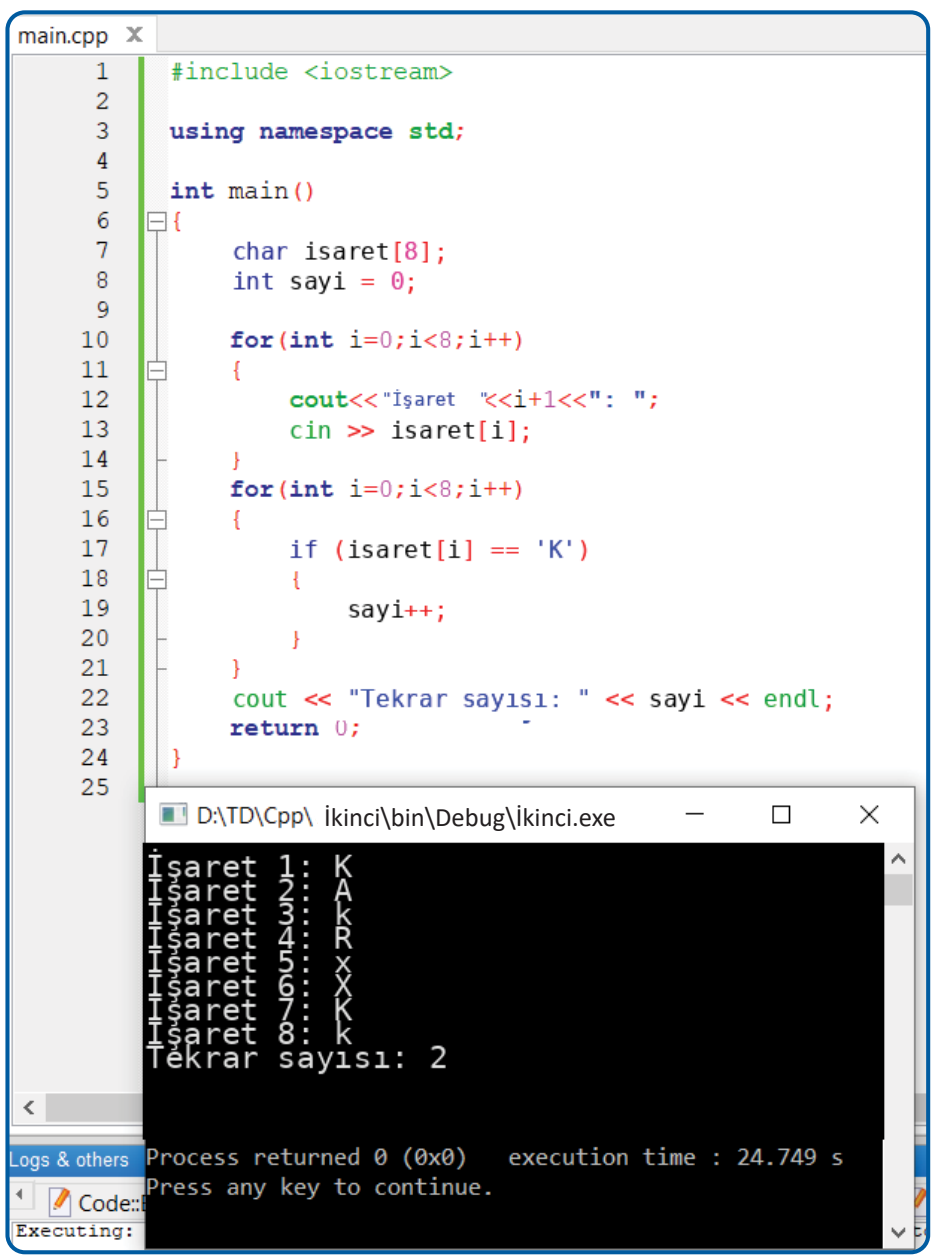


Örnekte tüm karakterler önce bir diziye girilir. Daha sonra dizideki karakterler sırayla kontrol edilerek herhangi birinin A harfine eşit olup olmadığı belirlenir. A harfi bulunduğunda sonuç kaydedilir.

Karakterlerin işlenmesinde çoğu zaman yalnızca belirli bir karakterin bulunup bulunmadığını belirlemek yeterli değildir. Pek çok durumda, söz konusu karakterin kaç kez tekrarlandığının belirlenmesi gerekir. Bu amaçla, aranan karakter her bulunduğunda değeri 1 artırılan bir sayma değişkeni kullanılır.

Örnek 2: Tekrar sayısını belirleme

Bir iletişim sistemine sekiz adet kod girilmektedir. K harfinin kaç kez tekrarlandığını belirleyiniz.



Örnekte sayma tekniği uygulanır. K harfi her bulunduğunda, sayma değişkeninin değeri 1 artırılır.

Karakter dizileriyle çalışırken her zaman yalnızca belirli bir harfin aranması gerekmez. Pratik programlama problemlerinde çoğu zaman belirli bir sembolün bulunup bulunmadığı kontrol edilir. Bu tür semboller, bazı sistemlerde özel anlam taşıyan #, @, * gibi karakterler veya başka işaretler olabilir. İşlem basamakları değişmez. Dizinin her elemanı sırayla aranan karakterle karşılaştırılır ve karakter bulunduğunda buna göre karar verilir.

Örnek 3: Özel bir karakterin bulunup bulunmadığının kontrolü

Bir erişim sistemine yedi adet kod girilmektedir. Bunların arasında # sembolünün bulunup bulunmadığını kontrol ediniz.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    char etiket[7];
    bool bulundu = false;
    for(int i=0;i<7;i++)
    {
        cout<<"Etiket  "<<i+1<<": ";
       cin >> etiket[i];
    }
    for(int i=0;i<7;i++)
    {
        if (etiket[i] == '#')
        {
            bulundu = true;
        }
    }
    if (bulundu)
    {
        cout<<"# işareti bulundu. "<<endl;
    }
    else
    {
        cout<<"# işareti bulunamadı. "<<endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Etiket 1: a
Etiket 2: s
Etiket 3: d
Etiket 4: s
Etiket 5: F
Etiket 6: W
Etiket 7: Q
# işareti bulunamadı.

Process returned 0 (0x0)   execution time : 9.477 s
Press any key to continue.
```

Örnekte girilen tüm karakterler sırayla # sembolüyle karşılaştırılır. # sembolü bulunduğunda program, kullanıcıya bu sembolün dizide bulunduğunu bildirir.

Karakter dizilerinin işlenmesinde arama işlemi için farklı koşullar belirlenebilir. Bazı problemlerde belirli bir harfin bulunup bulunmadığı aranırken, bazılarında belirli bir karakterin kaç kez tekrarlandığı sayılır, diğerlerinde ise belirli bir sembolün bulunup bulunmadığı kontrol edilir. Koşulun türü ne olursa olsun, temel işlem basamakları aynıdır. Dizinin her elemanı işlenir, karşılaştırma yapılır ve karşılaştırmanın sonucuna göre karar verilir. Bu teknik, programlama dillerinde metinsel verilerin işlenmesinin temelini oluşturur.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. Karakter dizisi nedir?
2. Tek bir karakteri saklamak için hangi veri tipi kullanılır?
3. Belirli bir karakterin dizide bulunup bulunmadığı nasıl kontrol edilir?
4. Belirli bir karakterin kaç kez tekrarlandığı nasıl belirlenir?
5. Karakter dizisinde harfler, rakamlar ve özel semboller saklanabilir mi?

ÖDEVLER

1. Bir kontrol sistemine sekiz adet kod girilmektedir. Bunların arasında M harfinin bulunup bulunmadığını kontrol eden bir program yazınız.
2. Bir iletişim merkezine altı adet karakter girilmektedir. 5 rakamının kaç kez tekrarlandığını belirleyen bir program yazınız.
3. Bir bilişim laboratuvarına on adet karakter girilmektedir. @ sembolünün girilip girilmediğini kontrol eden bir program yazınız.
4. Bir kayıt sistemine yedi adet kod girilmektedir. S harfinin kaç kez tekrarlandığını belirleyen bir program yazınız.
5. Bir araştırma istasyonuna dokuz adet sembol girilmektedir. * sembolünün bulunup bulunmadığını kontrol eden bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Karakter dizisi, tek tek karakterlerden oluşan bir dizidir.
- Bir karakteri saklamak için char veri tipi kullanılır.
- Karakterler harf, rakam veya özel sembol olabilir.
- Bir karakterin aranması, dizinin elemanlarının aranan karakterle karşılaştırılması yoluyla gerçekleştirilir.
- Bir sayma değişkeni kullanılarak belirli bir karakterin kaç kez tekrarlandığı belirlenebilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilişim sistemlerinde metinsel verilerin işlenmesi, en önemli programlama etkinliklerinden biridir. Kullanıcı adları, parolalar, ürün kodları veya çeşitli kodlar girilirken programlar çoğu zaman belirli karakterlerin ya da sembollerin kullanılıp kullanılmadığını kontrol eder. Bu derste tek tek karakterler ele alınsa da aynı ilkeler daha sonra sözcükler, cümleler ve daha büyük metinsel verilerle çalışırken de uygulanır.

2.23

DİZİ ELEMANLARINA MATEMATİKSEL FONKSİYONLARIN UYGULANMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu bir dizinin elemanlarını girebileceksiniz;
- Elemanları for döngüsü ile işleyebileceksiniz;
- Toplam, çarpım ve ortalama hesaplayabileceksiniz;
- En büyük ve en küçük elemanı belirleyebileceksiniz;
- Matematik kütüphanesindeki fonksiyonları kullanabileceksiniz.

TEKRARLAMA ALIŞTIRMALARI

1. cmath kütüphanesi ne için kullanılır?
2. Karekökü hesaplayan fonksiyon hangisidir?
3. Üs alma işlemi için hangi fonksiyon kullanılır?
4. abs() fonksiyonu ne için kullanılır?
5. Matematik kütüphanesindeki fonksiyonlar bir dizinin elemanları üzerinde uygulanabilir mi?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Bir dizinin elemanları üzerinde matematiksel fonksiyonları uygulamayı;
- Dizileri cmath kütüphanesiyle birlikte kullanmayı;
- Girilen verileri matematiksel hesaplamalarla işlemeyi;
- Elde edilen sonuçları karşılaştırmayı;
- Öğrendiğiniz teknikleri bir arada kullanarak pratik problemleri çözmeyi.

Verilerin işlenmesi sırasında çoğu zaman bir dizinin her elemanı üzerinde belirli bir matematiksel işlemin gerçekleştirilmesi gerekir. Pratik programlama çözümlerinde girilen verilerin karekökünün, karesinin veya mutlak değerinin hesaplanması gerekebilir. Bu amaçla C++ programlama dilinde cmath matematik kütüphanesi kullanılır.

Bu kütüphanedeki fonksiyonlar dizinin her elemanı üzerinde uygulanabilir. Bir döngü yardımıyla tüm değerler sırayla işlenir ve her bir değer üzerinde gerekli matematiksel işlem gerçekleştirilir. Bu şekilde bilim, teknik ve günlük yaşamla ilgili çeşitli pratik problemler çözülebilir.

Bir dizinin elemanları üzerinde matematiksel fonksiyonlar uygulanırken öncelikle veriler girilir, ardından her bir eleman sırayla işlenir. Her bir elemanın karekökü, karesi, mutlak değeri veya başka bir matematiksel fonksiyona göre değeri hesaplanabilir. Elde edilen sonuçlar doğrudan ekrana yazdırılabilir veya daha sonra başka işlemlerde kullanılabilir. Bu yöntem sayesinde çok sayıda veri, az sayıda programlama komutuyla hızlı bir şekilde analiz edilip işlenebilir.

Örnek 1: Ölçülen mesafelerin karekökü

Bir bilimsel laboratuvarda beş ölçülmüş mesafe değeri girilmektedir. Her bir mesafenin karekökünü hesaplayınız.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double mesafe[5];
    for(int i=0;i<5;i++)
    {
        cout<<"Mesafe   "<<i+1<<": ";
        cin >> mesafe[i];
    }
    cout<<endl;
    for(int i=0;i<5;i++)
    {
        cout << "Karekök: " << sqrt(mesafe[i]) << endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Mesafe 1: 5
Mesafe 2: 4
Mesafe 3: 3
Mesafe 4: 81
Mesafe 5: 6
Karekök: 2.23607
Karekök: 2
Karekök: 1.73205
Karekök: 9
Karekök: 2.44949
```

Örnekte girilen her veri bağımsız olarak işlenir. sqrt() fonksiyonu kullanılarak her bir değerin karekökü hesaplanır ve sonuç hemen ekrana yazdırılır.

Verilerin işlenmesi sırasında çoğu zaman bir dizinin tüm elemanlarına aynı matematiksel işlemin uygulanması gerekir. Örneğin, tüm ölçülen değerlerin karelerinin hesaplanması veya her bir ölçümün mutlak değerinin belirlenmesi gerekebilir. Bu tür durumlarda her eleman sırayla işlenir ve ilgili matematiksel fonksiyon elemana uygulanır. Bu yaklaşım sayesinde çok sayıda veri, basit bir programlama işlemiyle hızlı bir şekilde işlenebilir.

Örnek 2: Ölçülen Değerlerin Kareleri

Bir araştırma merkezinde beş ölçülmüş değer girilmektedir. Her bir değerin karesini hesaplayınız.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double deger[5];
    for(int i=0;i<5;i++)
    {
        cout << "Değer " << i + 1 << ": ";
        cin >> deger[i];
    }
    cout << endl;
    for(int i=0;i<5;i++)
    {
        cout << "Kare: " << pow(deger[i], 2) << endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Değer 1: 4
Değer 2: 2
Değer 3: 3
Değer 4: 4
Değer 5: 6
Kare: 16
Kare: 4
Kare: 9
Kare: 16
Kare: 36
```

Örnekte dizinin her elemanı ayrı ayrı işlenir. pow() fonksiyonu kullanılarak girilen her değerin karesi hesaplanır.

Verilerin pratik olarak işlenmesi sırasında negatif değerlerle de sıkça karşılaşılır. Bazı durumlarda sayıların işareti değil, yalnızca büyüklüğü önemlidir. Bu amaçla, bir sayının mutlak değerini belirleyen abs() fonksiyonu kullanılır. Tüm elemanlar işlendikten sonra elde edilen sonuçlar üzerinde ek analizler de yapılabilir.

Örnek 3: En büyük mutlak değer

Bir bilimsel laboratuvarda pozitif veya negatif olabilen beş ölçüm değeri girilmektedir. En büyük mutlak değeri belirleyiniz.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int olcum[5];
    int enBuyuk;
    for(int i=0;i<5;i++)
    {
        cout<<"Ölçüm  "<<i+1<< ": ";
        cin >> olcum[i];
    }
    enBuyuk = abs(olcum[0]);
    for(int i=1;i<5;i++)
    {
        if (abs(olcum[i]) > enBuyuk)
        {
            enBuyuk = abs(olcum[i]);
        }
    }
    cout << "En büyük mutlak değer: " << enBuyuk << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Ölçüm 1: 45
Ölçüm 2: -23
Ölçüm 3: 48
Ölçüm 4: 65
Ölçüm 5: -87
En büyük mutlak değer: 87

Process returned 0 (0x0)   execution time : 19.213 s
Press any key to continue.
```

Logs & others

Örnekte her bir elemana öncelikle abs() fonksiyonu uygulanır. Daha sonra elde edilen değerler karşılaştırılarak en büyük mutlak değer belirlenir.

TEKRARLAMA SORULARI VE ALIŞTIRMALARI

SORULAR

1. Programda cmath kütüphanesinin bulunması neden gereklidir?
2. Bir matematiksel fonksiyon bir dizinin tüm elemanlarına nasıl uygulanabilir?
3. Matematiksel fonksiyonların diziler ve döngülerle birlikte kullanılması neden yararlıdır

ÖDEVLER

1. Bir jeodezi biriminde beş mesafe değeri girilmektedir. Her bir mesafenin karekökünü hesaplayan bir program yazınız.
2. Bir fizik laboratuvarında altı ölçüm değeri girilmektedir. Her bir değerin karesini hesaplayan bir program yazınız.
3. Bir araştırma istasyonunda beş pozitif ve negatif ölçüm değeri girilmektedir. En büyük mutlak değeri belirleyen bir program yazınız.
4. Bir astronomi gözlemevinde yedi mesafe değeri girilmektedir. Her bir mesafenin küpünü hesaplayan bir program yazınız.
5. Bir mühendislik laboratuvarında altı ölçüm değeri girilmektedir. Her bir ölçümün mutlak değerini hesaplayan ve elde edilen sonuçları ekrana yazdıran bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- cmath kütüphanesindeki fonksiyonlar, bir dizinin elemanları üzerinde uygulanabilir.
- Diziler, döngüler ve matematiksel fonksiyonların birleştirilmesiyle çeşitli pratik problemler çözülebilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilişim sistemlerinde çok sayıda veri, matematiksel fonksiyonlar kullanılarak işlenmektedir. Bilim alanında ölçüm sonuçları, mühendislikte çeşitli fiziksel büyüklükler, finansal sistemlerde ise farklı istatistiksel göstergeler analiz edilebilir. Modern programlar çok büyük miktarda bilgiyi işlese de temel yaklaşım değişmez: Her veri, uygun bir matematiksel fonksiyon kullanılarak işlenir, elde edilen sonuçlar daha sonra analiz edilebilir ve karşılaştırılabilir.

2.24

FONKSİYONLARIN VE TEK BOYUTLU DİZİLERİN BİRLİKTE KULLANILMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Tek boyutlu dizileri kullanabileceksiniz;
- Elemanları girebilecek ve işleyebileceksiniz;
- Fonksiyonları tanımlayabilecek ve çağırabileceksiniz;
- for döngüsünü kullanabileceksiniz;
- Farklı programlama yapılarını bir arada kullanabileceksiniz.

TEKRARLAMA SORULARI

1. Bir fonksiyonun argümanı nedir?
2. Fonksiyon çağrısı nedir?
3. Bir fonksiyon bir program içinde birden fazla kez çağrılabilir mi?
4. Dizilerin işlenmesinde neden çoğunlukla döngü kullanılır?
5. Fonksiyonların ve dizilerin birlikte kullanılmasının ne gibi bir avantajı vardır?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Bir dizinin elemanları üzerinde fonksiyonları uygulamayı;
- Döngüleri ve fonksiyonları bir arada kullanmayı;
- Verileri kullanıcı tanımlı fonksiyonlarla işlemeyi;
- Dizilerle çalışırken geri dönüş değerlerini kullanmayı;
- Pratik programlama problemlerini çözmeyi.

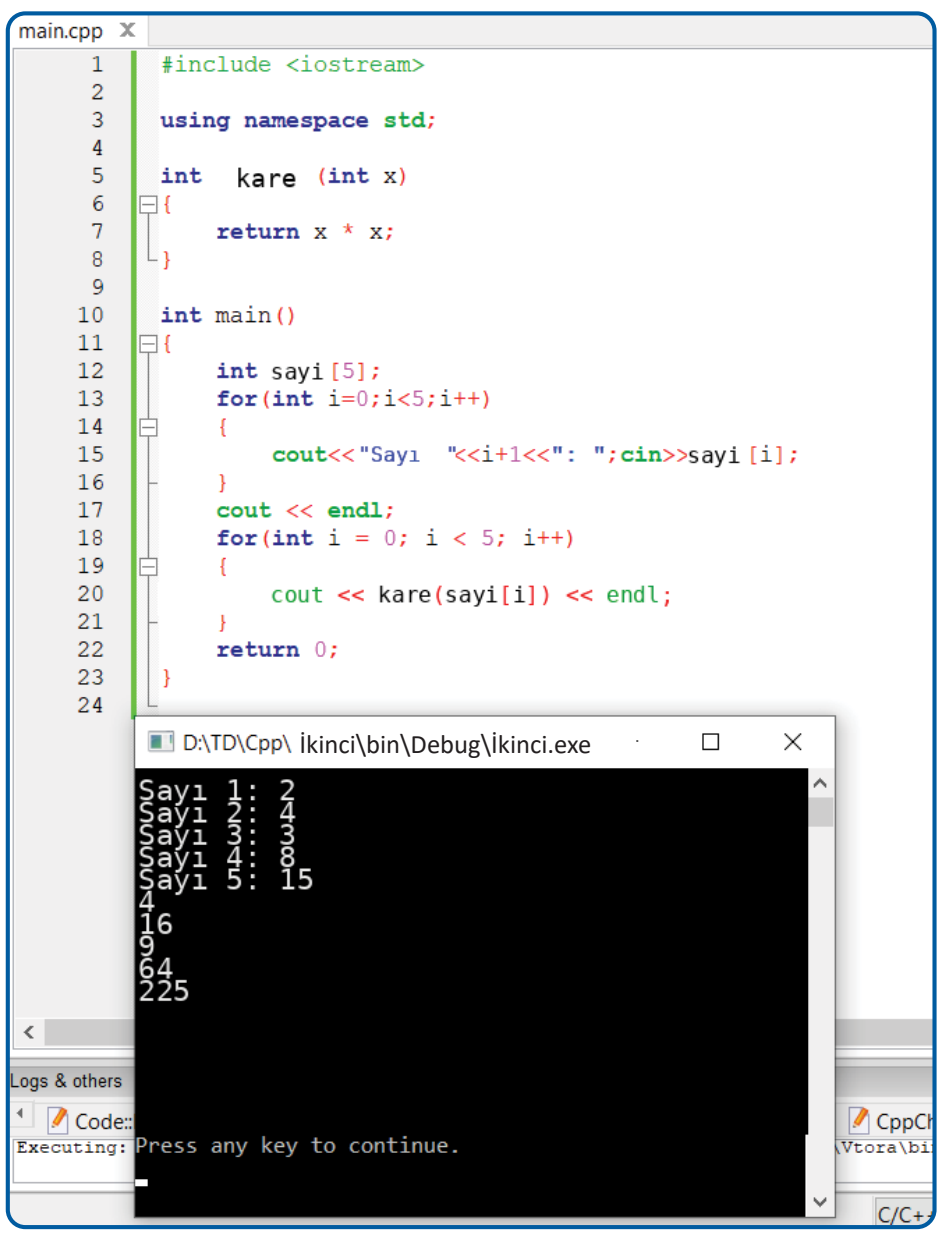
Verilerin işlenmesi sırasında çoğu zaman aynı işlemin bir dizinin birden fazla elemanı üzerinde gerçekleştirilmesi gerekir. Aynı programlama komutlarını tekrar tekrar yazmak yerine, bu komutlar bir fonksiyon içinde tanımlanabilir. Daha sonra fonksiyon, dizinin her elemanı için sırayla çağrılabilir.

Fonksiyonların ve tek boyutlu dizilerin birlikte kullanılması, programların daha düzenli ve anlaşılır olmasını sağlar. Aynı fonksiyon, aynı programlama komutlarının yeniden yazılmasına gerek kalmadan dizinin tüm elemanlarını işlemek için kullanılabilir.

Fonksiyonlar ve diziler birlikte kullanılırken çoğunlukla fonksiyon bir elemanı işler, döngü ise fonksiyonun dizinin tüm elemanları için çağrılmasını sağlar. Böylece her değer aynı şekilde işlenir ve program basit ve anlaşılır kalır. Bu yaklaşım, çeşitli programlama problemlerinin çözümünde geniş bir kullanım alanına sahiptir ve fonksiyonlar ile tek boyutlu dizilerin öğrenilmesinin doğal bir devamıdır.

Örnek 1: Elemanların karesi

Bir araştırma laboratuvarında beş değer girilmektedir. Bir fonksiyon kullanarak her bir değerin karesini ekrana yazdırınız.



Örnekte fonksiyon bir elemanı işler, döngü ise aynı fonksiyonun dizinin tüm elemanları için çağrılmasını sağlar.

Fonksiyonlar ve diziler birlikte kullanılırken fonksiyonun yalnızca matematiksel bir değer hesaplaması gerekmez. Pek çok durumda belirli bir elemanın verilen bir koşulu sağlayıp sağlamadığının kontrol edilmesi gerekir. Bu tür durumlarda fonksiyon bir mantıksal değer döndürebilir. Koşul sağlanıyorsa fonksiyon true, sağlanmıyorsa false döndürür. Bir döngü yardımıyla aynı fonksiyon dizinin tüm elemanları için çağrılabilir ve elde edilen sonuçlar daha sonra işlenebilir.

Örnek 2: Çift sayıların kontrolü

Bir araştırma merkezinde altı sayı girilmektedir. Bir fonksiyon kullanarak bu sayılardan kaç tanesinin çift olduğunu belirleyiniz.

main.cpp

```
#include <iostream>

using namespace std;

bool ciftMi(int x)
{
    if(x % 2 == 0)
    {
        return true;
    }
    return false;
}

int main()
{
    int sayi[6];
    int toplam = 0;
    for(int ı=0;ı<6;ı++)
    {
        cout<<"Sayı  "<<i+1<<": ";
        cin >> sayi[i];
    }
    for(int i = 0; i < 6; i++)
    {
        if (ciftMi(sayi[i]))
        {
            toplam++;
        }
    }
    cout << "Çift sayılar: " << toplam << endl;
    return 0;
}
```

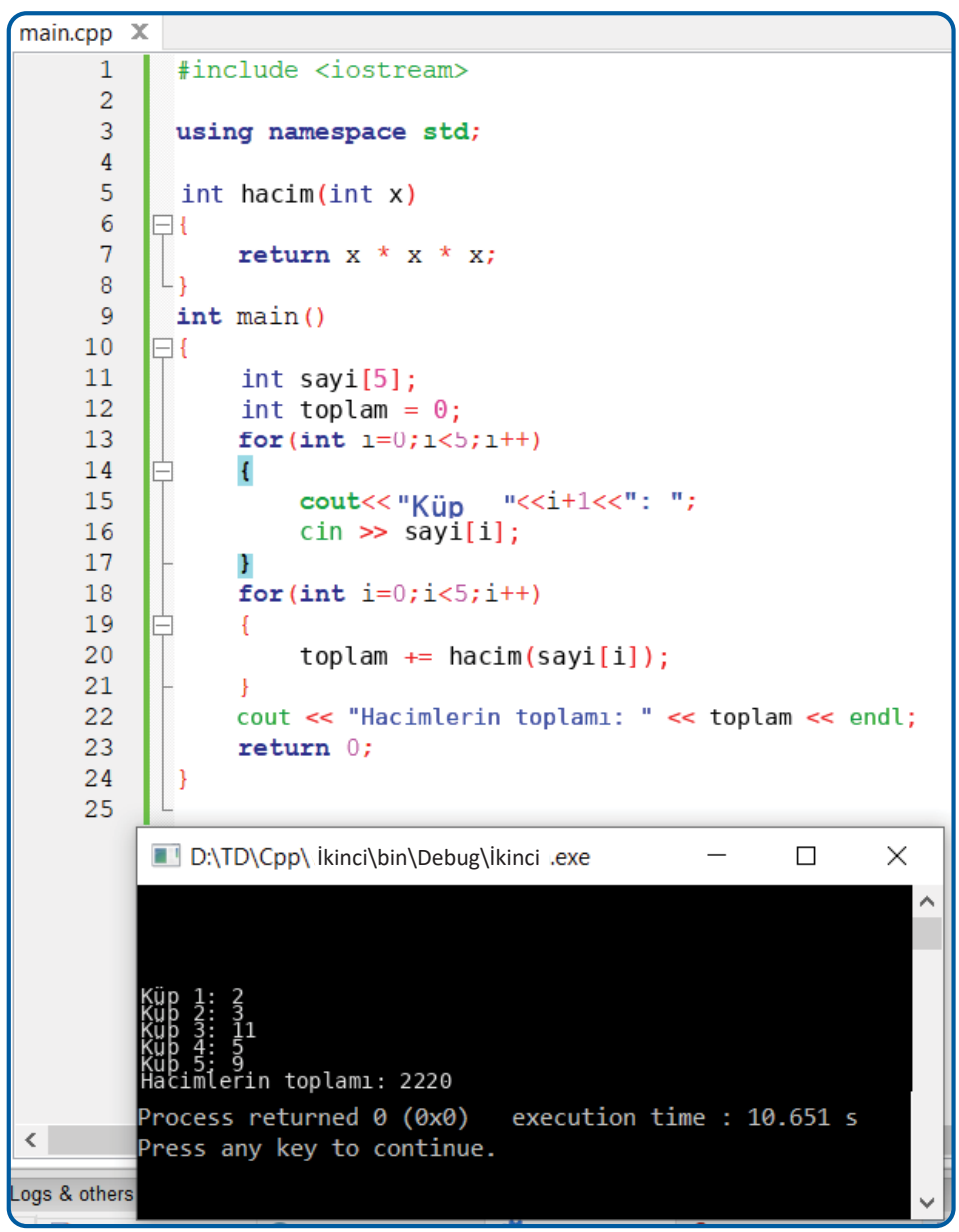
```
D:\TD\Cpp\ İkinci\bin\Deb
Sayı 1: 2
Sayı 2: 5
Sayı 3: 45
Sayı 4: 62
Sayı 5: 188
Sayı 6: 145
Çift sayılar: 3

Process returned 0 (0x0)
Press any key to continue
```

Örnekte fonksiyon, bir sayının çift olup olmadığını kontrol eder. Daha sonra aynı fonksiyon dizinin her elemanı için çağrılır ve koşulu sağlayan tüm sayılar sayılır.

Fonksiyonlarla çalışırken çoğu zaman bir dizinin her elemanı için belirli bir değerin hesaplanması ve ardından elde edilen sonuçların daha ileri işlemler için kullanılması gerekir. Bu şekilde daha önce ayrı ayrı öğrenilen farklı programlama teknikleri bir arada kullanılabilir. Bu yaklaşım sayesinde programlar daha düzenli ve bakımı daha kolay hâle gelir.

Örnek 3: Küplerin hacimleri ve toplamları

Bir teknik laboratuvarda uzaya gönderilecek 5 cismin çevresindeki alanın en uygun şekilde düzenlenmesi gerekmektedir. Bu 5 cismin tamamı küp şeklindedir. Küp şeklindeki cisimlere ait beş ölçüm bulunmaktadır. Bir fonksiyon kullanarak tüm ölçümlerin hacimleri toplamını hesaplayınız.



main.cpp

```cpp
#include <iostream>

using namespace std;

int kup(int x)
{
    return x * x * x;
}
int main()
{
    int sayi[5];
    int toplam = 0;
    for(int ı=0;ı<5;ı++)
    {
        cout<<"Ölçüm  "<<i+1<<": ";
        cin >> sayi[i];
    }
    for(int i=0;i<5;i++)
    {
        toplam += kup(sayi[i]);
    }
    cout << "Küplerin toplamı: " << toplam << endl;
    return 0;
}
```

```
D:\TD\Cpp\ İkinci\bin\Debug\İkinci.exe
Ölçüm 1: 2
Ölçüm 2: 5
Ölçüm 3: 14
Ölçüm 4: 22
Ölçüm 5: 4
Küplerin toplamı: 13589

Process returned 0 (0x0)   execution time : 9.686 s
Press any key to continue.
```

Örnekte fonksiyon, bir elemana ait değerden küpün hacmini hesaplar. Döngü ise bu fonksiyonun dizinin tüm elemanları üzerinde uygulanmasını sağlar. Elde edilen sonuçlar aşamalı olarak toplanır ve sonunda toplamları ekrana yazdırılır.

SORULAR VE TEKRARLAMA ALIŞTIRMALARI

SORULAR

1. Bir dizinin tek bir elemanını işleyen bir fonksiyon hangi işlemleri gerçekleştirebilir?
2. Döngü, aynı fonksiyonun birden fazla elemana uygulanmasını nasıl sağlar?
3. Bir fonksiyon hangi durumlarda mantıksal bir değer döndürebilir?
4. Veri işleme işlemlerinin bir fonksiyon içinde tanımlanması neden yararlıdır?
5. Fonksiyonların ve tek boyutlu dizilerin birlikte kullanılmasının avantajları nelerdir?

ÖDEVLER

1. Bir meteoroloji istasyonunda altı sıcaklık değeri girilmektedir. Sıcaklığın üç katını (3 ile çarpımını) hesaplayan bir fonksiyon kullanarak sonuçları ekrana yazdıran bir program yazınız.
2. Bir robotik merkezinde beş sonuç değeri girilmektedir. Bir sayının pozitif olup olmadığını kontrol eden bir fonksiyon kullanarak girilen pozitif değerlerin sayısını belirleyen bir program yazınız.
3. Bir bilimsel laboratuvarda altı ölçüm değeri girilmektedir. Bir sayının 5 ile bölümünden kalanı hesaplayan bir fonksiyon kullanarak elde edilen değerleri ekrana yazdıran bir program yazınız.
4. Bir spor merkezinde beş sonuç değeri girilmektedir. Her bir sonucun yarısını hesaplayan bir fonksiyon kullanarak elde edilen değerleri ekrana yazdıran bir program yazınız.
5. Bir enerji merkezinde altı ölçüm değeri girilmektedir. Bir değerin 100'den büyük olup olmadığını belirleyen bir fonksiyon kullanarak bu tür ölçümlerin sayısını hesaplayan bir program yazınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Fonksiyon, bir dizinin tek bir elemanını işleyebilir.
- Aynı fonksiyon, dizinin tüm elemanları için çağrılabilir.
- Fonksiyon, sayısal veya mantıksal bir değer döndürebilir.
- Döngü, fonksiyonun dizinin tüm elemanları üzerinde uygulanmasını sağlar.
- Fonksiyonların ve dizilerin birlikte kullanılması, programların daha düzenli ve bakımı daha kolay olmasını sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern programlamada fonksiyonlar, program kodunu düzenlemenin en önemli yöntemlerinden biridir. Aynı komutları programın birden fazla yerinde tekrar tekrar yazmak yerine, bu komutlar birden çok kez çağrılabilen fonksiyonlar içinde tanımlanır. Bu yaklaşım tek boyutlu dizilerle birleştirildiğinde, büyük veri kümeleri kolayca işlenebilir. Bu çalışma yöntemi, bilimsel ölçümlerin işlenmesi, finansal analizler, istatistiksel veriler ve diğer birçok modern bilişim sisteminde uygulanmaktadır.

2.25

DİZİLERLE ALIŞTIRMA PROBLEMLERİ

Kolay Ödevler

1. Bir dağ evinde bir hafta boyunca ziyaretçi sayısı kaydedilmektedir. Toplam ziyaretçi sayısını hesaplayan bir program yazınız.
2. Bir fidanlıkta on genç fidanın boy uzunlukları kaydedilmektedir. Ortalama boy uzunluğunu belirleyen bir program yazınız.
3. Bir müzede art arda sekiz gün boyunca ziyaretçi sayısı kaydedilmektedir. En fazla ziyaretçinin bulunduğu günü belirleyen bir program yazınız.
4. Bir rüzgâr enerji santralinde dokuz jeneratörün günlük elektrik üretimi kaydedilmektedir. En düşük üretime sahip jeneratörü belirleyen bir program yazınız.
5. Bir yaban hayatı parkında yedi farklı yaşam alanında doğan yavruların sayısı kaydedilmektedir. Üçten fazla yavrunun doğduğu yaşam alanlarının sayısını belirleyen bir program yazınız.
6. Bir kütüphaneye farklı sayılarda kitap içeren teslimatlar gelmektedir. On teslimatta gelen kitap sayıları kaydedilmektedir. Teslimatlardan herhangi birinin tam olarak 100 kitap içerip içermediğini kontrol eden bir program yazınız.
7. Bir meyve suyu fabrikasında sekiz saat boyunca üretilen şişe sayıları kaydedilmektedir. En başarılı saat ile en düşük üretimin gerçekleştiği saat arasındaki farkı belirleyen bir program yazınız.
8. Bir milli parkta dokuz gözlem boyunca görülen geyiklerin sayısı kaydedilmektedir. En az beş geyiğin görüldüğü gözlem sayısını belirleyen bir program yazınız.
9. Bir astronomi kampında on gece boyunca gözlemlenen meteorların sayısı kaydedilmektedir. Çift numaralı gecelerde gözlemlenen toplam meteor sayısını hesaplayan bir program yazınız.
10. Bir akvaryumda sekiz havuzdaki yeni doğan balıkların sayısı takip edilmektedir. Yeni doğan balık sayısı ortalama sayıdan fazla olan havuzların sayısını belirleyen bir program yazınız.

Orta Zorlukta Ödevler

11. Bir madende sekiz kare şeklindeki kazı alanının yüzölçümleri kaydedilmektedir. Her bir kazı alanının bir kenar uzunluğunu belirleyen bir program yazınız.
12. Bir fidanlıkta yedi kare şeklindeki çiçek yatağının kenar uzunlukları kaydedilmektedir. Her bir çiçek yatağının alanını hesaplayan bir program yazınız.
13. Bir jeoloji laboratuvarında dokuz ölçüm boyunca arazide meydana gelen pozitif ve negatif yer değiştirmeler kaydedilmektedir. Yönü dikkate almadan en büyük yer değiştirmeyi belirleyen bir program yazınız.

14. Bir spor hazırlık merkezinde altı sporcunun koştuğu mesafeler takip edilmektedir. Her sporcunun koştuğu mesafenin karesini hesaplayan ve elde edilen değerleri ekrana yazdıran bir program yazınız.
15. Bir mimarlık stüdyosunda dairesel çeşmeler tasarlanmaktadır. Sekiz çeşmenin yarıçapları bilinmektedir. Her bir çeşmenin alanını hesaplayan bir program yazınız.
16. Bir inşaat şirketinde dekoratif eleman olarak kullanılacak yedi küpün kenar uzunlukları girilmektedir. Her bir küpün hacmini hesaplayan bir program yazınız.
17. Bir astronomi gözlemevinde gök cisimlerinin pozitif ve negatif sapmaları kaydedilmektedir. Yönlerini dikkate almadan tüm sapmaların toplam büyüklüğünü hesaplayan bir program yazınız.
18. Bir bilimsel laboratuvarda kare şeklindeki örneklerin alanları girilmektedir. Hangi örneğin en uzun kenara sahip olduğunu belirleyen bir program yazınız.
19. Bir yenilenebilir enerji merkezinde altı dairesel güneş enerjisi platformunun yarıçapları bilinmektedir. Tüm platformların toplam alanını hesaplayan bir program yazınız.
20. Bir jeodezi biriminde ölçüm sırasında oluşan pozitif ve negatif hatalar kaydedilmektedir. Hataların pozitif veya negatif olmasına bakılmaksızın ortalama hata büyüklüğünü hesaplayan bir program yazınız.

Zor Ödevler

21. Bir arıcılık merkezinde her kovana verimlilik değerlendirmesi yapılmaktadır. Bir kovan, en az 25 kg bal ürettiğinde başarılı kabul edilmektedir. Sezon boyunca sekiz kovanın bal üretimi kaydedilmektedir. Başarılı kovanların sayısını belirleyen bir program yazınız.
22. Bir dağcılık kulübünde on zirvenin yükseklikleri kaydedilmektedir. Bir zirve, deniz seviyesinden en az 2000 metre yüksekte olduğunda yüksek kabul edilmektedir. Kaydedilen yüksek zirvelerin sayısını belirleyen bir program yazınız.
23. Bir akvaryumda dokuz havuzdaki yeni doğan balıkların sayısı takip edilmektedir. Bir havuzda 50'den fazla balık doğduğunda havuz başarılı kabul edilmektedir. Başarılı havuzların sayısını belirleyen bir program yazınız.
24. Bir astronomi gözlemevinde yedi gök cisminin uzaklıkları kaydedilmektedir. Bir gök cismi, uzaklığı önceden belirlenmiş bir değerden küçük olduğunda yakın kabul edilmektedir. Yakın gök cisimlerinin sayısını belirleyen bir program yazınız.
25. Bir milli parkta on ağacın yaşları kaydedilmektedir. Bir ağaç en az 100 yaşında olduğunda asırlık kabul edilmektedir. Asırlık ağaçların sayısını belirleyen bir program yazınız.
26. Bir hayvan kurtarma merkezinde sekiz hayvanın vücut ağırlıkları kaydedilmektedir. Her hayvan, önceden belirlenmiş bir sınır değerine göre hafif veya ağır olarak sınıflandırılmaktadır. Ağır hayvanların sayısını belirleyen bir program yazınız.
27. Bir spor kampında dokuz yarışmacının atlama mesafeleri kaydedilmektedir. Bir atlayış, belirlenen sınırdan daha uzun olduğunda başarılı kabul edilmektedir. Gerçekleştirilen başarılı atlayışların sayısını belirleyen bir program yazınız.
28. Bir rüzgâr santralinde yedi türbinin enerji üretimi kaydedilmektedir. Bir türbin, belirlenen miktardan daha fazla enerji ürettiğinde verimli kabul edilmektedir. Verimli türbinlerin sayısını belirleyen bir program yazınız.
29. Bir arkeolojik keşif gezisinde bulunan sekiz eserin yaşları kaydedilmektedir. Bir eser, önceden belirlenmiş bir değerden daha eski olduğunda tarihî açıdan önemli kabul edilmektedir. Önemli eserlerin sayısını belirleyen bir program yazınız.
30. Bir yenilenebilir enerji merkezinde dokuz güneş enerjisi sisteminin günlük üretimi takip edilmektedir. Bir sistem, belirlenen miktardan daha fazla enerji ürettiğinde başarılı kabul edilmektedir. Başarılı sistemlerin sayısını belirleyen bir program yazınız.

31. Bir motosiklet kulübünde sekiz sürücünün katettiği mesafeler kaydedilmektedir. Bir sürücü en az 300 km yol katettiğinde maceracı statüsü kazanmaktadır. Bir sürücünün durumunu kontrol eden işlemi bir fonksiyonda tanımlayan ve maceracı statüsü kazanan sürücülerin sayısını belirleyen bir program yazınız.
32. Bir arı çiftliğinde yedi kovanın bal üretimi takip edilmektedir. Bir kovan en az 25 kg bal ürettiğinde verimli kabul edilmektedir. Bir kovanı kontrol eden işlemi bir fonksiyonla gerçekleştiren ve verileri bir dizi kullanarak işleyen bir program yazınız.
33. Bir uzay görevinde dokuz asteroidin büyüklükleri kaydedilmektedir. Bir asteroid, büyüklüğü belirlenen sınırdan fazla olduğunda riskli kabul edilmektedir. Tüm asteroidleri bir dizi kullanarak işleyen ve bir fonksiyon aracılığıyla bir asteroidin riskli olup olmadığını belirleyen bir program yazınız.
34. Bir arkeolojik keşif gezisinde sekiz eserin yaşı girilmektedir. Bir eser 1000 yıldan daha eski olduğunda önemli kabul edilmektedir. Bir eserin önemini belirleyen bir fonksiyon kullanan ve önemli eserlerin sayısını ekrana yazdıran bir program yazınız.
35. Bir fabrikada altı robotun ürettiği parça sayıları kaydedilmektedir. Bir robot 120'den fazla parça ürettiğinde verimli kabul edilmektedir. Bir robotun verimliliğini kontrol eden bir fonksiyon kullanan ve tüm robotların sonuçlarını bir dizide saklayan bir program yazınız.
36. Bir hayvan kurtarma merkezinde yedi hayvanın vücut ağırlıkları girilmektedir. Bir hayvanın ağırlığı önceden belirlenmiş sınırdan fazla olduğunda ağır kabul edilmektedir. Bir hayvanın durumunu kontrol eden bir fonksiyon kullanan ve ağır hayvanların sayısını hesaplayan bir program yazınız.
37. Akıllı bir şehirde sekiz binanın günlük elektrik tüketimi takip edilmektedir. Bir bina, günlük tüketimi 500 birimden fazla olduğunda yüksek tüketimli kabul edilmektedir. Bir binanın yüksek tüketimli olup olmadığını belirleyen bir fonksiyon kullanan bir program yazınız.
38. Bir spor kampında dokuz uzun atlama sonucu girilmektedir. Bir atlayış, belirlenen sınırdan daha uzun olduğunda başarılı kabul edilmektedir. Bir atlayışın başarılı olup olmadığını kontrol eden bir fonksiyon kullanan ve başarılı atlayışların sayısını ekrana yazdıran bir program yazınız.
39. Bir çevre devriyesi, sekiz farklı noktada toplanan atık miktarlarını kaydetmektedir. Bir nokta, atık miktarı belirlenen sınırdan fazla olduğunda kritik kabul edilmektedir. Bir noktanın durumunu kontrol eden bir fonksiyon kullanan ve kritik noktaların sayısını belirleyen bir program yazınız.
40. Bir su altı görevinde yedi dalışın derinlikleri girilmektedir. Bir dalış, derinliği 40 metreden fazla olduğunda derin kabul edilmektedir. Bir dalışın derin olup olmadığını kontrol eden bir fonksiyon kullanan ve tüm ölçümleri bir dizide saklayan bir program yazınız.

Konu Hakkında

Modern toplumda her gün büyük miktarda veri oluşturulmakta ve işlenmektedir. Veriler, okullarda, bankalarda, hastanelerde, kütüphanelerde, mağazalarda, devlet kurumlarında ve daha birçok alanda kullanılmaktadır. Verilerin etkin bir şekilde saklanması, düzenlenmesi ve işlenmesi için veri tabanlarından yararlanılır. Veri tabanları, bilgilere hızlı bir şekilde ulaşılmasını, bilgilerin güncellenmesini ve yönetilmesini sağlar. Bu sayede farklı kuruluşların çalışmaları daha verimli hâle gelir ve ihtiyaç duyulan bilgilere daha hızlı ve güvenilir bir şekilde erişilir.

Bu konu kapsamında veri tabanları ve veri tabanı yönetim sistemleri ile ilgili temel kavramlar ele alınmaktadır. Özellikle verilerin tablolarda düzenlenmesine, kayıtlar ve alanlara, veriler arasındaki ilişkilerin oluşturulmasına, ayrıca bilgilerin aranmasına, girilmesine ve güncellenmesine önem verilmektedir. Uygulamalı etkinlikler aracılığıyla günlük yaşamın farklı alanlarında ve modern çalışma ortamlarında veri tabanlarını kullanmaya yönelik temel bilgi ve beceriler kazanılmaktadır.

TEMEL KAVRAMLAR

- Veri tabanı;
- Veri tabanı yönetim sistemi;
- Birincil anahtar;
- İlişki;
- Arama;
- Güncelleme;
- Sıralama;
- Filtreleme;
- Sorgu (query);
- Rapor.

BU KONUYU TAMAMLADIĞINIZDA...

- Verileri girebilecek ve düzenleyebileceksiniz;
- Bilgileri tablolarda düzenleyebileceksiniz;
- Farklı bilgisayar programlarını kullanabileceksiniz;
- Bilgileri arayabileceksiniz;
- Verileri analiz edebilecek ve karşılaştırabileceksiniz;
- Pratik problemleri çözerken dijital teknolojileri uygulayabileceksiniz;
- Bilgileri işlemek için bilişim araçlarını kullanabileceksiniz.

KONU 3

3.1

VERİ TABANLARININ TEMELLERİ

BU KONUYU TAMAMLADIĞINIZDA...

- Farklı bilgisayar programlarını kullanabileceksiniz;
- Verileri girebilecek ve işleyebileceksiniz;
- Bilgileri tablo biçiminde düzenleyebileceksiniz;
- İnternette bilgi arayabileceksiniz;
- Farklı dijital hizmetleri kullanabileceksiniz;
- Verileri analiz edebilecek ve karşılaştırabileceksiniz.

DÜŞÜNME SORULARI

1. Bankalar, müşterilerinin verilerini nasıl saklar?
2. Okul, öğrencilerin ve öğretmenlerin kayıtlarını nasıl tutar?
3. İnternet mağazaları hangi ürünlerin mevcut olduğunu nasıl bilir?
4. Binlerce kullanıcı aynı bilgilere aynı anda nasıl erişebilir?
5. Verilerin kolayca bulunabilmesi ve güncellenebilmesi neden önemlidir?

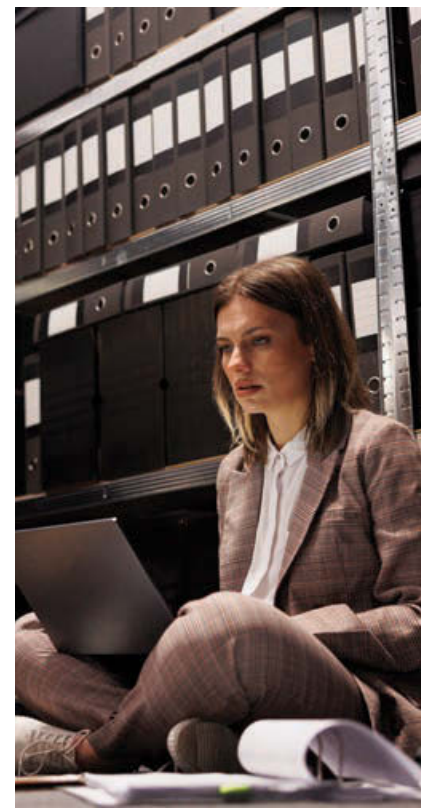
AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Veri tabanının ne olduğunu açıklamayı;
- Veri tabanlarının kullanım alanlarına örnekler vermeyi;
- Veri tabanlarının neden kullanıldığını açıklamayı;
- Veri tabanlarının kullanımının avantajlarını analiz etmeyi;
- Veri tabanlarının modern bilişim sistemlerindeki önemini.

Modern toplumda her gün çok büyük miktarda bilgi oluşturulmaktadır. Bir mağazada yapılan her alışveriş, ATM'den yapılan her para çekme işlemi, internette yapılan her arama ve sosyal medya üzerinden gönderilen her mesaj, saklanması ve işlenmesi gereken birer veridir. Bunun yanı sıra okullar, üniversiteler, bankalar, hastaneler, devlet kurumları ve daha birçok kuruluş da büyük miktarda bilgi üretmektedir. Bu kuruluşların düzgün çalışabilmesi için verilerin her an erişilebilir olması, hızlı bir şekilde bulunabilmesi ve işlenebilmesi gerekir.

Bilişim teknolojilerinin gelişmesiyle birlikte bilgilerin yönetilmesine yönelik gereksinimler de önemli ölçüde artmıştır. Verilerin yalnızca saklanması yeterli değildir, aynı zamanda güvenilir, doğru ve kullanma yetkisine sahip kullanıcılar için kolay erişilebilir olması gerekir. Ayrıca yeni bilgilerin kolayca eklenmesi, mevcut bilgilerin değiştirilmesi ve gereksiz verilerin silinmesi gerekir. Bu nedenlerle modern bilişim sistemlerinde büyük miktardaki verileri düzenlemek ve yönetmek için özel çözümler kullanılmaktadır.

3.1.1 DOSYALARDAN MODERN SİSTEMLERE



Bilişim sistemlerinin ilk dönemlerinde veriler çoğunlukla ayrı dosyalarda saklanıyordu. Her program, bilgileri düzenlemek için kendi yöntemini kullanıyor ve çalışması için gerekli veriler farklı belgelere kaydediliyordu. Bilgi miktarı görece az olduğu ve bilişim sistemlerini sınırlı sayıda kullanıcı kullandığı sürece bu yöntem yeterliydi.

Zamanla veri miktarı önemli ölçüde arttı. Aynı bilgiler farklı programlarda kullanılmaya başlandı ve çoğu zaman aynı verinin birden fazla yere girilmesi gerekti. Örneğin, bir kullanıcının adresi veya telefon numarası değiştiğinde, bu değişikliğin söz konusu bilgilerin bulunduğu tüm dosyalarda yapılması gerekiyordu. Dosyalardan biri güncellenmediğinde aynı veri için farklı değerler ortaya çıkabiliyor ve bu durum hatalara ve sistemin güvenilirliğinin azalmasına neden oluyordu.

Bilgilerin aranması da bir sorun hâline geldi. Veri miktarının artmasıyla birlikte bilgileri bulmak giderek daha fazla zaman ve kaynak gerektirdi. Ayrıca aynı bilgilerin tekrar tekrar saklanması, depolama alanının gereksiz yere kullanılmasına ve verilerin işlenmesinin zorlaşmasına neden oluyordu. Bu tür sistemlerin bakımı giderek daha karmaşık ve maliyetli hâle geliyordu.

Bilişim teknolojilerinin gelişmesi, verilerin farklı bir şekilde düzenlenmesini gerekli kıldı. Veriler çok sayıda bağımsız dosyaya dağıtılmak yerine, merkezi olarak saklanmalarını, daha kolay güncellenmelerini ve bilgilere daha hızlı erişilmesini sağlayan bütünleşik sistemlerde düzenlenmeye başlandı. Bu yaklaşım, modern veri tabanlarının gelişiminin temelini oluşturmuştur.

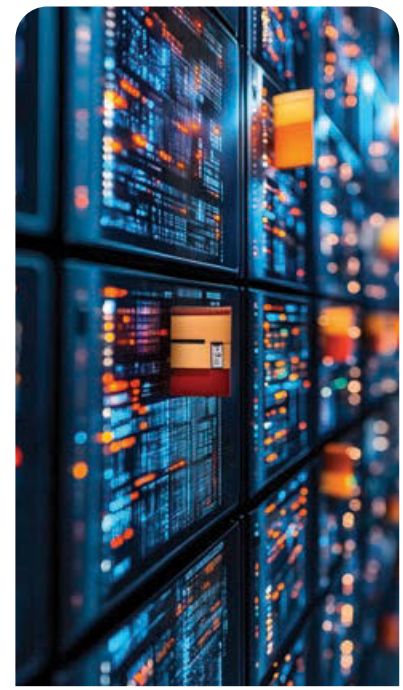
3.1.2 VERİ TABANI NEYİ İFADE EDER?

Veri tabanı, verilerin güvenli bir şekilde saklanmasını, kolayca aranmasını ve verimli bir biçimde işlenmesini sağlayan düzenli bir veri topluluğudur. Temel amacı, bilgilerin ihtiyaç duyulduğu anda erişilebilir olmasını ve kolayca eklenebilmesini, değiştirilebilmesini ve kullanılabilmesini sağlamaktır. Verilerin birbirinden bağımsız dosyalarda klasik yöntemlerle saklanmasından farklı olarak veri tabanları, bilgilerin doğruluğunu ve erişilebilirliğini artıracak şekilde düzenlenmesini sağlar.

Veri tabanlarının en önemli avantajlarından biri, aynı bilgilerin birden fazla kullanıcı ve farklı bilişim sistemleri tarafından kullanılabilmesidir. Yeni bir veri eklendiğinde veya mevcut bir bilgi değiştirildiğinde, yapılan değişiklik erişim yetkisine sahip tüm kullanıcılar için hemen kullanılabilir hâle gelir. Böylece verilerin gereksiz yere tekrarlanması önlenir ve hata oluşma olasılığı azalır.

Veri tabanları yalnızca bilgileri saklamak için kullanılmaz. Aynı zamanda verilerin aranmasını, karşılaştırılmasını, sıralanmasını ve analiz edilmesini sağlar. Bu özellikler sayesinde büyük miktarda veri hızlı bir şekilde işlenebilir ve farklı kararların alınmasında kullanılabilir. Bu nedenle veri tabanları, modern bilişim sistemlerinin en önemli bileşenlerinden biridir.

3.1.3 VERİ TABANLARININ KULLANIM ALANLARI



Veri tabanları günümüzde insan faaliyetlerinin neredeyse tüm alanlarında kullanılmaktadır. Eğitim alanında öğrenciler, öğretmenler, dersler ve eğitim-öğretim faaliyetleriyle ilgili kayıtların tutulmasında yararlanılmaktadır. Sağlık kuruluşlarında tıbbi bilgilerin ve hastalara ilişkin verilerin düzenli bir şekilde yönetilmesini sağlarken, bankacılık sektöründe finansal hizmetlerin ve işlemlerin güvenli biçimde yürütülmesine olanak tanımaktadır.

Veri tabanları, çağdaş bilişim sistemlerinde de geniş bir kullanım alanına sahiptir. Çevrim içi mağazalarda, sosyal medya platformlarında, elektronik posta sistemlerinde, uzaktan eğitim platformlarında ve daha birçok dijital hizmette kullanılmaktadır. Bir kullanıcı herhangi bir web sitesine giriş yaptığında, bilgi aradığında veya elektronik ödeme gerçekleştirdiğinde, arka planda çoğu zaman bir veri tabanı çalışmaktadır.

Bulut teknolojilerinin, mobil uygulamaların ve yapay zekânın gelişmesi, veri tabanlarının önemini daha da artırmaktadır. Modern bilişim sistemleri saniyede milyarlarca veriyi işleyebilmekte; bu süreçte veri tabanlarının güvenilirliği, doğruluğu ve işlem hızı, günümüzde her gün kullandığımız dijital hizmetlerin başarılı ve kesintisiz biçimde çalışabilmesi açısından temel unsurlar olarak öne çıkmaktadır.

3.1.4 VERİ TABANLARININ AVANTAJLARI

Veri tabanlarının kullanılması, bilgilerin klasik yöntemlerle saklanmasına kıyasla birçok avantaj sağlar. Veriler sistematik bir şekilde düzenlendiğinden, bilgilere daha hızlı erişilebilir ve veriler daha kolay güncellenebilir. Aynı zamanda hata oluşma ve bilgilerin gereksiz yere tekrarlanma olasılığı azalır.

Bir diğer önemli avantaj, birden fazla kullanıcının aynı veriler üzerinde aynı anda çalışabilmesidir. Bazı kullanıcılar yeni bilgiler girerken diğer kullanıcılar mevcut verileri arayabilir veya analiz edebilir. Bu özellik, özellikle çok sayıda kullanıcıya hizmet veren modern bilişim sistemleri açısından büyük önem taşır.

Veri tabanları ayrıca bilgilerin daha güvenli bir şekilde saklanmasını sağlar. Verilere erişim, kullanıcıların ihtiyaçlarına ve yetkilerine göre sınırlandırılabilir. Modern sistemler, bilgilerin korunması ve güvenli bir şekilde saklanması için çeşitli mekanizmalar da sunar. Bu özellikleri sayesinde veri tabanları, modern bilişim teknolojilerinin vazgeçilmez bir parçası hâline gelmiş ve kullanım alanları sürekli genişlemiştir.

Her eğitim-öğretim yılında yeni öğrencilerin kayıt olduğu bir ortaöğretim kurumunu ele alalım. Öğrencilerin kişisel bilgilerinin yanı sıra sınıfların, öğretmenlerin, derslerin, ders programlarının, başarı durumlarının ve daha birçok bilginin de kaydedilmesi gerekir. Yıl boyunca bazı bilgiler değişir, yeni öğrenciler kayıt olur ve bazı öğrenciler eğitimlerini tamamlar. Girilen bu verilerin ileride de erişilebilir olması gerekir. Eğitimini tamamlayan bir kişinin, aradan uzun bir süre geçtikten sonra bile bu bilgiyi gelecekteki sistemlere veya şirketlere sunması

gerekebilir. Herhangi bir belge kaybolduğunda, eğitimin ne zaman tamamlandığına bakılmaksızın bu belgenin yeniden düzenlenebilmesi gerekir.

Tüm bu bilgilerin çok sayıda farklı belgede tutulduğunu varsayalım. Bir öğrencinin telefon numarasını, belirli bir sınıftaki öğrencilerin listesini veya belirli bir öğretmenle ilgili bilgiyi bulmak gerektiğinde, birçok farklı belge arasında arama yapmak gerekecektir. Ayrıca her değişikliğin birden fazla yerde yapılması gerekeceğinden, hata yapma olasılığı da artacaktır.

Bir veritabanı kullanılarak tüm bilgiler tek bir sistem içerisinde düzenlenebilir. Yeni veriler kolayca eklenebilir, mevcut bilgiler güncellenebilir ve ihtiyaç duyulan veriler hızlı bir şekilde bulunabilir. Böylece işlerin organizasyonu büyük ölçüde iyileşir ve bilgilerin işlenmesi için gereken süre azalır.

Aynı mantık birçok başka alanda da uygulanmaktadır. Hastaneler hastaların kayıtlarını, bankalar müşterilerini ve işlemlerini, kütüphaneler kitaplarını, günümüz internet hizmetleri ise milyonlarca kullanıcı hesabını ve çeşitli dijital içerikleri yönetmektedir. Veritabanları olmadan bu sistemlerin işleyişi çok daha zor olurdu.

Veritabanları, bilgilerin merkezî bir şekilde saklanmasını ve ihtiyaç duyan kullanıcıların bu bilgilere erişebilmesini sağlar. Değişiklikler kolayca yapılabilir, veriler hızlı bir şekilde bulunabilir ve işlenebilir. Bu özellikleri sayesinde veritabanları, günümüz bilgi sistemlerinin temel unsurlarından biridir. Bilgilerin düzenli bir şekilde saklanması yalnızca işlemlerin daha hızlı yapılmasını sağlamakla kalmaz, aynı zamanda verilerin güvenliğini ve doğruluğunu da artırır. Günümüzde bilgi teknolojileri, her gün kullanılan çeşitli dijital hizmetlerin sorunsuz çalışmasını sağlamak için bu tür sistemlere dayanmaktadır.

Bilgi teknolojilerinin gelişmesi, çok büyük miktarlarda verinin oluşturulmasına ve işlenmesine yol açmıştır. Verilerin bağımsız dosyalarda klasik yöntemlerle saklanması zamanla birçok dezavantaj ortaya çıkarmıştır. Bunlar arasında bilgilerin tekrarlanması, güncelleme işlemlerinin karmaşık olması ve verilere ulaşmada yaşanan zorluklar bulunmaktadır. Bu nedenle verileri daha verimli bir şekilde düzenlemek ve yönetmek için yeni bir yönteme ihtiyaç duyulmuştur.

Veritabanları, bilgilerin güvenli bir şekilde saklanması ve işlenmesi için geliştirilmiş modern bir çözümdür. Verilerin kolayca eklenmesini, değiştirilmesini ve birden fazla kullanıcı tarafından kullanılmasını sağlar. Böylece çalışmaların doğruluğu ve verimliliği artırılır. Bu özellikleri sayesinde veritabanları günümüzde modern toplumun birçok farklı alanında yaygın olarak kullanılmaktadır. Veritabanları hakkında temel bilgi edinmek, bilgi teknolojileri okuryazarlığının önemli bir parçasıdır. Veritabanlarının önemini ve kullanım alanlarını anlamak, modern bilgi sistemlerini ve bunların pratik uygulamalarını öğrenmenin temelini oluşturur.

BİLİYOR MUSUNUZ?

Dünyada her gün çok büyük miktarda yeni veri oluşturulduğunu biliyor musunuz? Bu verilerin büyük bir bölümü cep telefonlarının, internet hizmetlerinin, sosyal medya platformlarının ve çeşitli dijital cihazların kullanılmasıyla ortaya çıkmaktadır. Bu verilerin düzenlenmesi ve işlenmesi için çok büyük miktarda bilgiyi depolayabilen ve yönetebilen modern veri tabanları kullanılmaktadır.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Veri tabanı, düzenli bir veri topluluğudur.
Modern bilişim sistemleri büyük miktarda bilgiyi işlemektedir.
Verilerin birbirinden bağımsız dosyalarda klasik yöntemlerle saklanmasının bazı dezavantajları vardır.
Veri tabanları bilgilerin daha kolay saklanmasını, aranmasını ve güncellenmesini sağlar.
Farklı alanlarda, modern toplumun birçok bölümünde yaygın olarak kullanılmaktadır.
Veri tabanları, modern bilişim teknolojilerinin önemli bir parçasıdır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern bilişim sistemleri çok büyük miktarda veriyi işleyebilmektedir. Büyük internet şirketleri, bilimsel araştırma merkezleri ve çeşitli dijital platformlar her gün sürekli olarak oluşturulan ve güncellenen milyarlarca bilgiyle çalışmaktadır. Bu tür verilerin yönetimi, bilişim alanında özel bir çalışma alanı oluşturmaktadır. Yeni teknolojilerin geliştirilmesi ise bilgilerin daha hızlı ve daha verimli bir şekilde işlenmesini sürekli olarak mümkün kılmaktadır.

BİLGİNİZİ TEST EDİN

1. Soruları cevaplayınız.

a) Veri tabanı nedir?
b) Veri tabanları neden kullanılır?
c) Veri tabanlarının kullanıldığı üç alan belirtiniz.

2. Doğru mu, yanlış mı?

a) Veri tabanları yalnızca bilgi saklamak için kullanılır.
b) Veri tabanındaki veriler güncellenebilir.
c) Modern bilişim sistemleri veri tabanlarını kullanmaz.

3. Düşünün ve açıklayınız.

Bir banka, hastane veya okul, bilgilerini düzenlemek için veri tabanı kullanmasaydı ne gibi büyük sorunlarla karşılaşabilirdi?

3.2

VERİ TABANINDA TABLO OLUŞTURMA

BU KONUYU TAMAMLADIĞINIZDA...

- Veri tabanının ne olduğunu açıklayabileceksiniz;
- Veri tabanlarının kullanım alanlarına örnekler verebileceksiniz;
- Veri tabanlarının neden kullanıldığını açıklayabileceksiniz;
- Veri tabanlarının kullanımının avantajlarını analiz edebileceksiniz;
- Bilgileri düzenlemek için tablolar kullanabileceksiniz.

DÜŞÜNME SORULARI

- Çok miktardaki bilgi en kolay nasıl düzenlenebilir?
- Verilerin sistematik bir şekilde düzenlenmesi neden önemlidir?
- Bir veri tabanında bilgiler nasıl düzenlenir?
- Veriler anlaşılır bir şekilde düzenlenmezse ne olur?
- Tüm bilgilerin tek bir tabloda bulunması gerekir mi?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- İlişkisel veri tabanının ne olduğunu açıklamayı;
- Tablo, alan ve kayıt kavramlarını birbirinden ayırt etmeyi;
- Verilerin tablolarda nasıl düzenlendiğini analiz etmeyi;
- Basit bir tablonun yapısını planlamayı;
- Veri tabanlarıyla çalışmak için kullanılan programları kullanmayı;
- Basit bir tablo oluşturmayı.

Günlük yaşamda bilgiler çoğu zaman tablolar hâlinde düzenlenir. Okullar öğrencilerle ilgili, kütüphaneler kitaplarla ilgili, spor kulüpleri müsabakalarla ilgili, çeşitli kurumlar ise kullanıcıları ve sundukları hizmetlerle ilgili tablolar oluşturur. Tablo biçimindeki gösterim, bilgilerin anlaşılır olmasını, kolayca karşılaştırılmasını ve ihtiyaç duyulan verilerin hızlı bir şekilde bulunmasını sağlar.

Aynı yaklaşım veri tabanlarında da kullanılır. Modern bilişim sistemleri çok büyük miktarda bilgi işlese de verilerin düzenlenmesi çoğunlukla tablolarla başlar. Bu yöntem sayesinde veriler sistematik bir şekilde girilebilir, güncellenebilir ve farklı amaçlarla kullanılabilir.

Tablo biçimindeki gösterimin günlük çalışmalar ile veri tabanları arasında bir bağlantı oluşturması da dikkat çekicidir. Birçok program, verilerin tablo biçiminde dışa aktarılmasına olanak tanırken farklı programlarda oluşturulan tablolar da veri tabanlarına aktarılabilir. Örneğin, bir tabloda düzenlenmiş veriler bir veri tabanına içe aktarılabilir, veri tabanındaki veriler ise daha sonra işlenmek ve analiz

edilmek üzere yeniden dışa aktarılabilir. Bu özellik, farklı bilişim sistemleri arasındaki bilgi alışverişini önemli ölçüde kolaylaştırır.

3.2.1 VERİLERI DÜZENLEMENİN BİR YOLU OLARAK TABLO

Çok sayıda bilginin saklanması gerektiğinde, bu bilgilerin mantıklı bir şekilde düzenlenmesi gerekir. Bilgileri düzenlemenin en basit ve anlaşılır yollarından biri tablo biçiminde göstermektir. Tabloda veriler, kolayca girilebilecek, görüntülenebilecek ve karşılaştırılabilecek şekilde düzenlenir.

Bir okulda bulunan bilgisayarların kayıtlarının tutulması gerektiğini düşünelim. Her bilgisayar için demirbaş numarası, üretici, model, işletim sistemi ve kullanıldığı sınıf gibi çeşitli bilgilerin bilinmesi gerekir. Bu bilgilerin bir tabloda düzenlenmesi, her verinin kendine ait bir yere sahip olmasını sağlar ve bilgilerin kullanılmasını önemli ölçüde kolaylaştırır.

Modern veri tabanlarında tablolar, bilgileri düzenlemenin temel yöntemini oluşturur. Tablolar, verilerin sistematik bir şekilde düzenlenmesini ve kolayca erişilebilir olmasını sağlarken aynı zamanda daha sonraki işlemler ve analizler için de sağlam bir temel oluşturur.

Uygulamalı Örnek

Aşağıdaki tabloyu inceleyelim.

KAYIT NUMARASI	ÜRETİCİ	MODEL	İŞLETİM SİSTEMİ	KONUM
1001	Lenovo	ThinkCentre	Windows 11	Kabin 1
1002	Dell	OptiPlex	Windows 11	Kabin 2
1003	HP	ProDesk	Linux	Kabin 3
1004	Lenovo	ThinkCentre	Windows 11	Kabin 4

Tablodan, her bilgisayarın birden fazla farklı bilgiyle tanımlandığı kolayca görülebilir. Yeni bir bilgisayar satın alındığında, ona ait bilgiler kolayca eklenebilir. Bir bilgisayar başka bir odaya taşındığında ise yalnızca ilgili bilginin değiştirilmesi gerekir.

Bu tür bir düzenleme, verilerin kolayca incelenmesini ve karşılaştırılmasını sağlar. Örneğin, belirli bir işletim sistemini kullanan bilgisayarların sayısı veya belirli bir odada bulunan cihazlar hızlı bir şekilde belirlenebilir.

Tabloyu Nasıl Analiz Edebiliriz?

Önceki örneğe dikkatlice bakıldığında, tablonun sütunlardan ve satırlardan oluştuğu görülebilir. Her sütun belirli bir bilgi türünü içerirken, her satır bir bilgisayarın eksiksiz tanımını temsil eder. Bu düzenleme sayesinde veriler anlaşılır ve kolayca takip edilebilir durumdadır.

Veritabanlarında sütunlara alanlar denir, çünkü her sütun belirli bir bilgi türü için kullanılır. Satırlara ise kayıtlar denir ve belirli bir nesneye ilişkin bilgileri içerir. Bu düzenleme, veritabanındaki verilerin organize edilmesinin temelini oluşturur.

Bilgilerin iyi bir şekilde organize edilmesi, bilgilerin daha hızlı girilmesini, daha kolay yönetilmesini ve daha verimli kullanılmasını sağlar. Bu nedenle tabloların doğru şekilde oluşturulması, bir veritabanı oluşturulurken en önemli adımlardan biridir.

3.2.2 İLİŞKİSEL VERİTABANI

Uygulamada tüm verilerin tek bir tabloya yerleştirilmesi oldukça nadirdir. Eğer öğrenciler, öğretmenler, dersler, notlar, ders programı ve devamsızlıklarla ilgili tüm bilgiler tek bir tabloda tutulursa, bu tablo çok büyük ve yönetilmesi zor hâle gelir ve hatalara daha açık olur. Bu nedenle modern veritabanları bilgileri birbiriyle bağlantılı birden fazla tabloda organize eder. Bu organizasyon biçimine ilişkisel model, bu modeli kullanan veritabanlarına ise ilişkisel veritabanları denir.

İlişkisel modelin temel fikri, her tablonun belirli bir konu veya varlıkla ilgili verileri içermesidir. Örneğin, bir okul bilgi sisteminde öğrenciler için ayrı bir tablo, öğretmenler için ayrı bir tablo, dersler için ayrı bir tablo ve notlar için ayrı bir tablo bulunabilir. Aynı verilerin birden fazla yerde tekrarlanması yerine tablolar, ortak alanlar aracılığıyla birbirine bağlanır. Böylece hata yapma olasılığı azalır ve veriler daha düzenli ve verimli bir şekilde saklanır.

İlişkisel veritabanları günümüzde en yaygın kullanılan veritabanı türüdür. Eğitim kurumlarında, bankalarda, sağlık kuruluşlarında, devlet kurumlarında, internet mağazalarında ve daha birçok bilgi sisteminde kullanılırlar. Yapıları sayesinde büyük miktardaki veriler hızlı bir şekilde girilebilir, aranabilir, güncellenebilir ve analiz edilebilir.

Verilerin birden fazla tabloda organize edilmesine örnek

Öğrenciler ve onların notlarıyla ilgili kayıtların tutulması gerektiğini varsayalım.

Tüm bilgileri tek bir tabloda saklamak yerine iki tablo oluşturulabilir.

Tablo: Öğrenciler

ID	Ad	Soyad
1	Ana	Petrova
2	Marko	Stoyanov
3	Elena	Nikolovska

Tablo: Notlar

ID	ÖğrenciID	Ders	Not
1	1	Bilişim	5
2	1	Matematik	4
3	2	Bilişim	5
4	3	Matematik	5

İlk tabloda öğrencilerle ilgili temel bilgiler, ikinci tabloda ise onların notları saklanır. Öğrencinin adı ve soyadı her seferinde tekrarlanmak yerine, iki tablonun birbirine bağlanmasını sağlayan bir kimlik numarası (ID) kullanılır. Böylece verilerin tekrarlanması azaltılır ve bilgilerin düzenlenmesi daha iyi hâle gelir.

Neden Birden Fazla Tablo Kullanılır?

İlişkisel veritabanlarının en önemli avantajlarından biri, tekrarlanan verilerin azaltılmasıdır. Öğrenciyle ilgili bilgiler yalnızca bir kez saklanırsa, herhangi bir değişiklik yalnızca tek bir yerde yapılır. Öğrenci soyadını değiştirirse veya herhangi bir bilginin düzeltilmesi gerekirse, değişiklik kolayca yapılabilir ve kayıtların bir kısmının güncellenmeden kalması riski ortadan kalkar.

Verilerin birden fazla tabloda organize edilmesi, bilgilerin daha hızlı aranmasını da sağlar. Program, binlerce sütun ve satırdan oluşan devasa bir tabloyu taramak yerine daha küçük ve daha iyi organize edilmiş tablolarla çalışır. Bu durum, milyonlarca kaydı işleyen modern bilgi sistemlerinde özellikle önemlidir.

Ayrıca bu çalışma yöntemi, veritabanının daha kolay genişletilmesini sağlar. Gelecekte yeni bilgilere ihtiyaç duyulursa, mevcut yapıyı önemli ölçüde değiştirmeden yeni bir tablo eklenebilir. Bu nedenle ilişkisel veritabanları modern bilgi sistemlerinde bir standart hâline gelmiştir.

3.2.3 VERİTABANLARIYLA ÇALIŞMA PROGRAMLARI

Verilerin oluşturulması, düzenlenmesi ve işlenmesi için veritabanı yönetim sistemleri olarak bilinen özel programlar kullanılır. Bu programlar, kullanıcıların bilgisayarda bilgilerin fiziksel olarak nasıl saklandığıyla doğrudan ilgilenmesine gerek kalmadan veri girişi yapmasını, verileri düzenlemesini, aramasını ve görüntülemesini sağlayan araçlar sunar.

Eğitimde ve veritabanlarının temellerini öğrenirken genellikle Microsoft Access ve LibreOffice Base programları kullanılır. Bu programlar, kullanıcının tablolar oluşturabileceği, veri girebileceği, tablolar arasında bağlantılar kurabileceği ve formlar, sorgular ve raporlar hazırlayabileceği grafiksel bir çalışma ortamı sağlar. Görsel yapıları sayesinde bu programlar, veritabanlarıyla çalışmanın temel ilkelerini öğrenmek için uygundur.

Büyük şirketlerde ve kurumlarda kullanılan başka birçok veritabanı yönetim sistemi de bulunmaktadır. Ancak çalışma prensipleri benzerdir. Basit bir eğitim aracı veya karmaşık bir kurumsal sistem kullanılsa da veriler tablolarda organize edilir, tablolar arasında bağlantılar kurulur ve verilerin aranması ve işlenmesi sağlanır.

Programın veritabanının kendisi değil, veritabanıyla çalışmak için kullanılan bir araç olduğunu anlamak önemlidir. Veritabanı, düzenlenmiş bir bilgi topluluğudur, program ise kullanıcının bu bilgilerle çalışmasını sağlayan araçları sunar.

Tablo Oluşturma

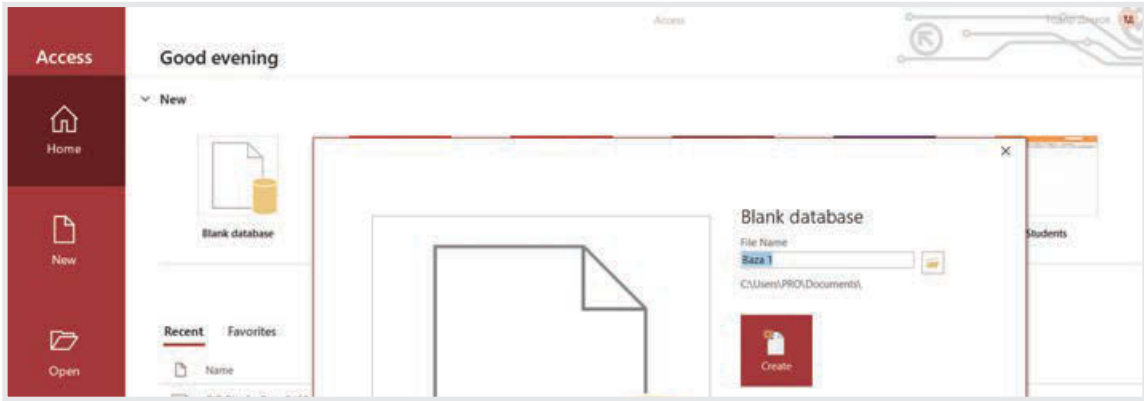
Veri girişine başlamadan önce, veritabanında hangi bilgilerin saklanacağının dikkatlice planlanması gerekir. Bu adım önemlidir çünkü organizasyonun kalitesi, çalışmanın verimliliğini doğrudan etkiler. Tablo kötü tasarlanırsa daha sonra veri girme, arama ve güncelleme işlemlerinde sorunlar ortaya çıkabilir.

Bir tablo oluşturulurken öncelikle tabloda bulunacak alanlar belirlenir. Her alan belirli bir bilgi türü için kullanılır. Örneğin, öğrenciler için bir tablo oluşturulacaksa ad, soyad, doğum tarihi, adres, telefon numarası ve gerekli diğer bilgiler için alanlar belirlenmelidir.

Alanların doğru seçilmesi, verilerin düzenli, anlaşılır ve kolay işlenebilir olmasını sağlar. Bu nedenle tablo oluşturmak yalnızca teknik bir işlem değil, aynı zamanda veritabanında saklanacak bilgilerin dikkatli bir şekilde planlanması sürecidir.

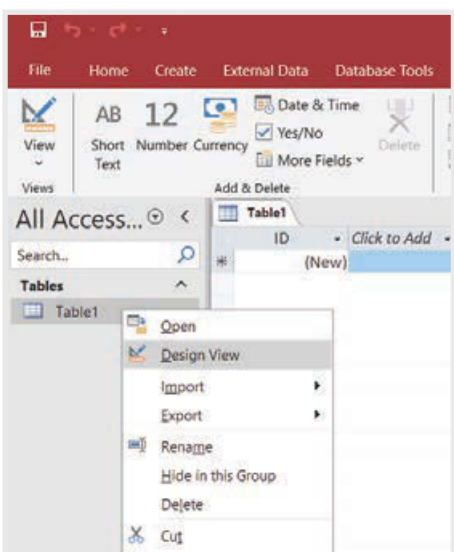
Örnek: Öğrenciler için tablo planlama

MS Access programı başlatıldıktan sonra boş bir veritabanı seçilmelidir. Yeni veritabanına bir ad verilir ve oluşturulacak veritabanının tamamının kaydedileceği konum seçilir.



Okulun öğrencilerle ilgili elektronik kayıt tutmak istediğini varsayalım.

Tablo oluşturulmadan önce, her öğrenci için hangi bilgilerin gerekli olacağı düşünülmelidir. Öğrencinin adı, soyadı, doğum tarihi, adresi, telefon numarası ve e-posta adresinin saklanması gerektiği belirlenebilir.

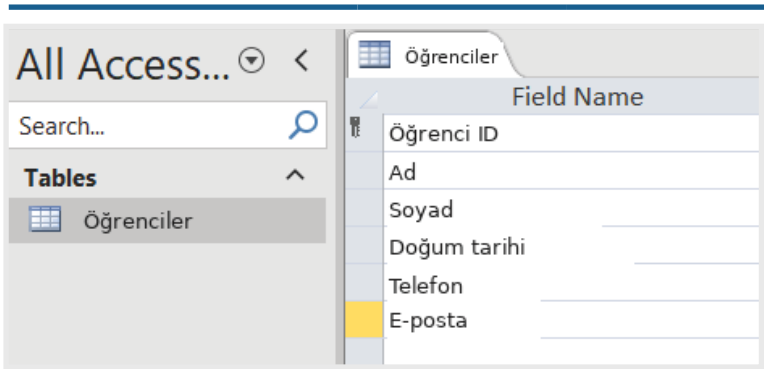


Gerekli alanları belirlemek için tablonun Tasarım Görünümü (Design View) bölümüne geçilir.

Seçim yapıldıktan hemen sonra tablo, kullanıcı tarafından verilen bir adla kaydedilir. Tabloya, içinde ne tür verilerin bulunduğunu ifade eden anlamlı adlar verilmesi önerilir.

Bu analize dayanarak aşağıdaki tablo yapısı planlanabilir:

Öğrenci ID, Ad,	Soyad,	Doğum tarihi,	Adres,	Telefon,	E-posta



Örnekte, her sütunun tüm öğrenciler için girilecek belirli bir veri türünü temsil ettiği görülmektedir. Bu tür bir planlama, tablo oluşturma sürecinin ilk adımını oluşturur.

Örneğin Analizi

Tablo oluştururken yalnızca farklı isimlere sahip sütunlar eklemek yeterli değildir. Hangi bilgilerin gerçekten gerekli olduğunun dikkatlice belirlenmesi gerekir. Gereksiz alanlar eklenirse tablo karmaşıklaşır ve bakımı zorlaşır. Öte yandan, önemli alanlar eksik olursa daha sonra tablonun yapısında ek değişiklikler yapılması gerekir.

Tabloların planlanması, bir veritabanı tasarlanırken en önemli adımlardan biridir. Uygulamada, sistem oluşturulmaya başlamadan önce genellikle saklanması gereken bilgiler ve bu bilgilerin nasıl kullanılacağı ayrıntılı olarak analiz edilir. Böylece verilerin daha verimli bir şekilde organize edilmesi ve verilerle daha kolay çalışılması sağlanır.

SONUÇ:

Tablolar, ilişkisel veritabanlarında verileri düzenlemenin temel yolunu oluşturur. Tablolar sayesinde bilgiler daha anlaşılır hâle gelir, kolayca girilebilir ve basit bir şekilde işlenebilir. Tabloların doğru şekilde organize edilmesi, kaliteli bir veritabanı oluşturmanın ilk adımıdır.

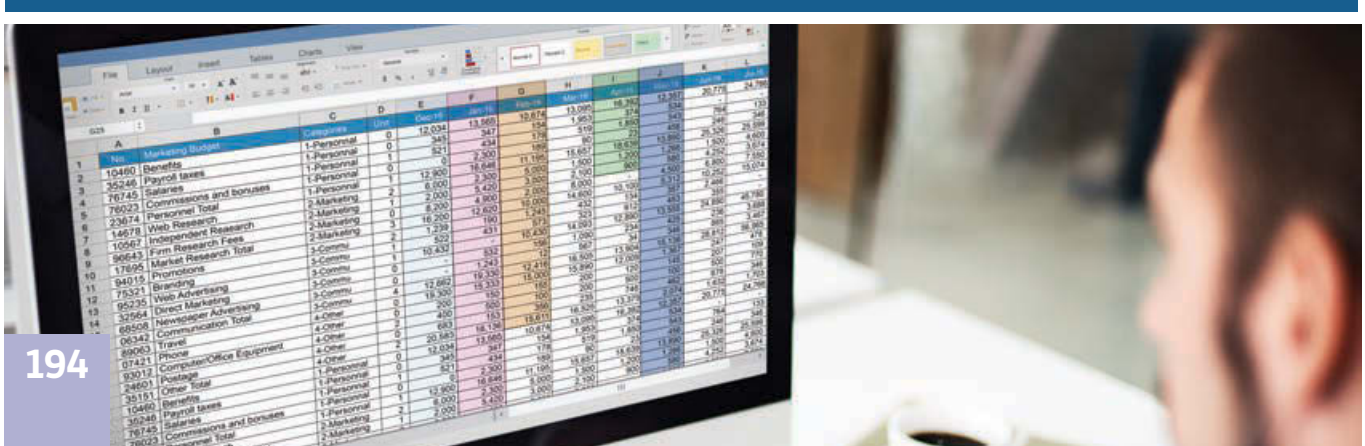
Her tablo, bilgilerin sistematik bir şekilde düzenlenmesini sağlayan alanlar ve kayıtlardan oluşur. Bu yapı sayesinde veriler daha kolay aranabilir, güncellenebilir ve farklı bilgi sistemlerinde kullanılabilir.

BİLİYOR MUSUNUZ?

Günlük olarak kullandığımız verilerin büyük bir kısmının tablolarda organize edildiğini biliyor muydun? Okul kayıtları, bankacılık sistemi veya internet mağazası söz konusu olsun, bilgilerin düzenlenmesinin temelinde çoğunlukla bilgilerin tablo şeklinde gösterilmesi bulunur.

TEKRARLAMA SORULARI

1. İlişkisel veritabanı nedir?
2. Tablo nedir?
3. Tablodaki alanın işlevi nedir?
4. Kayıt nedir?
5. Veriler neden birden fazla tabloda organize edilir?
6. Veritabanlarıyla çalışmak için hangi programlar kullanılabilir?
7. Tablo oluşturulmadan önce planlama yapılması neden önemlidir?



PRATİK ALIŞTIRMALAR

Ödev 1

Bir kütüphanedeki kitaplarla ilgili bilgilerin saklanacağı bir tablo için plan oluşturunuz.

Ödev 2

Aşağıdaki bilgilerden hangilerinin alan olduğunu belirleyiniz:

- Öğrencinin adı
- Marko Petrov
- Telefon
- Oda 12
- Doğum tarih

Ödev 3

Aşağıdaki tabloda kaç alan ve kaç kayıt olduğunu belirtiniz.

Ad	Soyad	Sınıf
Ana	Petrova	I-1
Marko	Stoyanov	I-2
Elena	Nikolovska	I-1

Ödev 4

Bir okula ait tüm verilerin tek bir tabloda saklanmasının neden iyi bir fikir olmadığını açıklayınız.

ARAŞTIRMA ÖDEVİ

Günümüzde veritabanlarıyla çalışmak için en sık kullanılan programları araştırınız. Her biri için aşağıdaki bilgileri bulunuz:

- Adı;
- Üreticisi;
- Ücretsiz mi yoksa ticari mi olduğu;
- Kullanım alanı.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Tablo, ilişkisel veritabanının temel öğesidir.

Alan, tablodaki bir sütunu temsil eder.

Kayıt, tablodaki bir satırı temsil eder.

İlişkisel veritabanları, birbiriyle bağlantılı birden fazla tablo içerir.

Veriler tablo biçiminde içe ve dışa aktarılabilir.

İyi planlanmış bir tablo, verilerle daha kolay çalışmayı sağlar.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Büyük şirketlerde ve kurumlarda veritabanları binlerce tablo ve milyonlarca kayıt içerebilir. Buna rağmen, tabloların iyi organize edilmesi sayesinde ihtiyaç duyulan bilgiler saniyenin çok kısa bir bölümünde bulunabilir.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

Alan neyi temsil eder?

a) Satır
b) Sütun
c) Tablo
d) Veritabanı

Kayıt neyi temsil eder?

a) Sütun
b) Alan
c) Satır
d) Veritabanı

Doğru veya yanlış

1. Bir veritabanı birden fazla tablo içerebilir.
2. Alan ve kayıt aynı kavramlardır.
3. Tablolar verileri düzenlemek için kullanılır.

3.3

VERİ TÜRLERİ

BU KONUYU TAMAMLADIĞINIZDA...

- Tablonun ne olduğunu açıklarsın;
- Alan ile kayıt arasındaki farkı ayırt edersin;
- Basit bir tablonun yapısını planlarsın;
- Veritabanında tablo oluşturursun;
- Bilgileri tablo biçiminde organize edersininiz.

DÜŞÜNME SORULARI

- Tüm alanlara aynı türde veriler girilebilir mi?
- Bilgisayar, belirli bir değerin metin, sayı veya tarih olduğunu nasıl anlar?
- Verilerin girilmesiyle ilgili kuralların olması neden önemlidir?
- Kullanıcı yanlış bir değer girerse ne olabilir?
- Veritabanındaki verilerin doğruluğu nasıl artırılabilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Farkli veri türlerini ayirt edersin;
- Uygun veri türünü seçersin;
- Alanlarin özelliklerini açiklarsin;
- Doğrulama kurallarini uygularsin;
- Girilen verileri kontrol etme yöntemlerini analiz edersin;
- Kaliteli ve doğru bilgilerin önemini fark edersin.

Tablo oluştururken yalnızca içereceği alanları belirlemek yeterli değildir. Her alan için, o alana ne tür verilerin girileceğinin de belirlenmesi gerekir. Bunun nedeni, farklı bilgilerin farklı şekillerde kaydedilmesidir. Örneğin ad ve soyad harf ve karakterlerden oluşur, doğum tarihi tarih biçiminde girilir, devamsızlık sayısı veya notlar ise genellikle sayısal değerler olarak girilir.

Uygun veri türünün belirlenmesi, programın bilgileri doğru şekilde işlemesini sağlar. Sayılar için ayrılmış bir alana metin veya tarih alanına uygun olmayan bir değer girilirse, verilerin işlenmesi sırasında hatalar meydana gelebilir. Bu nedenle, tablonun oluşturulması sırasında her alanın hangi değerleri içerebileceği belirlenir.

Modern veritabanı sistemlerinde birden fazla veri türü bulunur. İsimleri programdan programa biraz farklılık gösterebilse de kullanım amaçları benzerdir. En sık kullanılan veri türleri metinsel veriler, sayısal veriler, tarih ve saat, mantıksal değerler ve uzun metin alanlarıdır.

3.3.1 EN SIK KULLANILAN VERİ TÜRLERİ

Metinsel Veriler

Metin veri türü, kelimelerin, adların, adreslerin, telefon numaralarının ve matematiksel hesaplamalarda kullanılmayan diğer bilgilerin girilmesi için kullanılır. Telefon numarası rakamlardan oluşsa da genellikle metin olarak saklanır, çünkü telefon numaraları toplanmaz, çıkarılmaz veya çarpılmaz.

Örnekler:

- Todor Dimov
- Sveti Nikole
- II-3
- 071123456

Sayısal Veriler

Sayısal veri türü, verilerin matematiksel olarak işlenmesi gerektiğinde kullanılır. Bu verilerle hesaplamalar yapılabilir, ortalamalar, maksimum ve minimum değerler belirlenebilir ve diğer istatistiksel analizler gerçekleştirilebilir.

Örnekler:

- 5
- 18
- 125
- 98.5

Tarih ve Saat

Bu veri türü, tarih ve saat bilgilerinin girilmesi için kullanılır. Bu sayede öğrencilerin yaşı, belirli etkinliklerin süresi veya iki olay arasındaki zaman farkı hesaplanabilir.

Örnekler:

- 15.09.2025
- 21.05.2026
- 08:30

Mantıksal Veriler

Mantıksal veri türü yalnızca iki değerin girilmesine olanak sağlar. Farklı programlarda bunlar Evet/Hayır, Doğru/Yanlış veya Açık/Kapalı şeklinde gösterilebilir.

Örnekler:

- Evet
- Hayır

yada

- True
- False

Uzun Metin

Bazen daha uzun açıklamaların, notların veya yorumların girilmesi gerekir. Bu gibi durumlarda daha fazla miktarda metin depolamaya olanak sağlayan veri türü kullanılır.

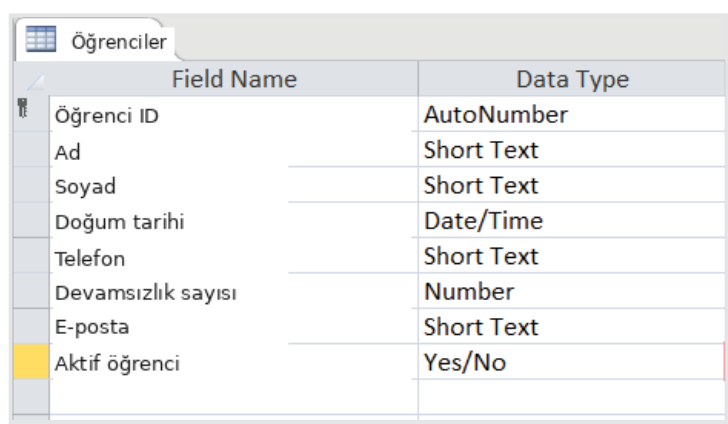
Örnek:

"Öğrenci, bilişim yarışmalarına aktif olarak katılmakta ve bölgesel ve ulusal yarışmalarda birden fazla ödül kazanmıştır."

Pratik Örnek

Öğrenciler için hazırlanmış bir tabloyu inceleyelim.

Alan	Veri türü
Öğrenci ID	Sayı
Ad	Metin
Soyad	Metin
Doğum tarihi	Tarih/Saat
Devamsızlık sayısı	Sayı
E-posta	Metin
Aktif öğrenci	Mantıksal



Resim 1: Veri türü

Örnekte, her alanın belirli bir veri türüne sahip olduğu görülmektedir. Bu düzenleme sayesinde program, hangi değerlerin girilebileceğini ve bu değerlerin nasıl işlenmesi gerektiğini bilir. Örneğin devamsızlık sayısı toplanabilir ve analiz edilebilirken, ad için böyle bir işlem anlamlı değildir.

Veri türünü doğru seçmek neden önemlidir?

Uygun veri türünün seçilmesi, veritabanıyla çalışmanın doğruluğunu ve verimliliğini etkiler. Tüm bilgiler normal metin olarak saklansaydı program hesaplamaları doğru şekilde yapamaz, verileri sıralayamaz veya girilen değerlerin doğru olup olmadığını kontrol edemezdi.

Ek olarak, doğru veri türünün seçilmesi belleğin daha verimli kullanılmasına ve bilgilerin daha hızlı işlenmesine katkı sağlar. Milyonlarca kayıt içeren büyük veritabanlarında bu tür optimizasyonlar sistemin çalışması üzerinde önemli bir etkiye sahiptir.

Her alan için veri türünün dikkatli bir şekilde belirlenmesi, güvenli ve verimli bir veritabanı için sağlam bir temel oluşturur. Bu adım, tabloların oluşturulmasındaki en önemli aşamalardan biridir.

Alanların Özellikleri

Tablo oluşturulurken her alan için yalnızca adı ve veri türü belirlenmez. Ayrıca bilgilerin girilme, görüntülenme ve işlenme biçimini belirleyen çeşitli özellikler de ayarlanabilir. Bu özellikler sayesinde veriler daha kontrollü bir şekilde girilir ve hata oluşma olasılığı azalır.

Bir tabloda öğrencinin adı için bir alan bulunduğunu varsayalım. Herhangi bir kısıtlama belirlenmezse kullanıcı boş bir değer, çok uzun bir metin veya anlamsız rastgele karakterler girebilir. Alan özellikleri kullanılarak, daha kaliteli veri girişini sağlayacak kurallar önceden belirlenebilir.

Alan özellikleri, veritabanındaki bilgilerin kontrol edilmesi için önemli bir mekanizmadır. Veri girişinin standartlaştırılmasını, verilerin daha iyi organize edilmesini ve işlenmeleri sırasında daha fazla güvenilirlik sağlanmasını mümkün kılar.

En Sık Kullanılan Özellikler

Alan Boyutu

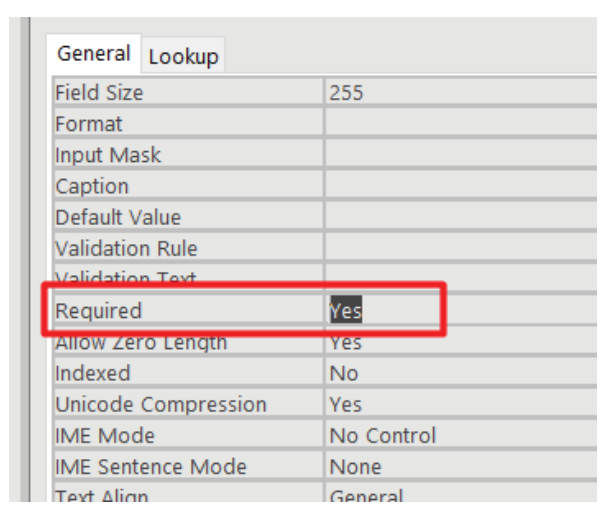
Metinsel verilerde, alana girilebilecek maksimum karakter sayısı sıklıkla belirlenir. Örneğin, öğrenci kodunun her zaman en fazla on karakter içerdiği biliniyorsa, çok daha uzun bir metnin girilmesine izin vermeye gerek yoktur.

Alan boyutunun sınırlandırılması, uygun olmayan verilerin girilme olasılığını azaltır ve bellek alanının daha verimli kullanılmasını sağlar.

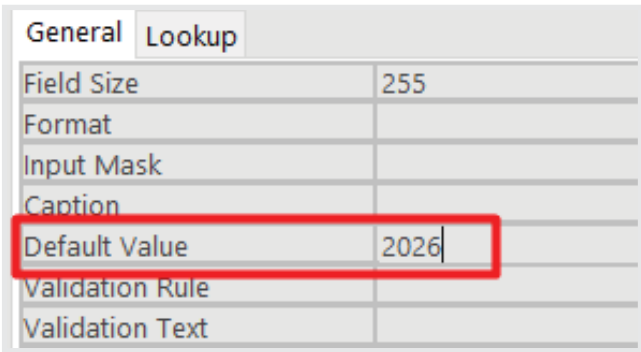
Zorunlu Alan

Bazen bazı bilgilerin mutlaka girilmesi gerekir. Örneğin, okul kayıtlarında adı veya soyadı olmayan bir öğrencinin bulunması kabul edilemez. Bu gibi durumlarda, alanın boş bırakılması hâlinde kaydın kaydedilmesine izin vermeyen bir özellik kullanılır.

Bu özellik veri kalitesinin artırılmasına katkı sağlar ve eksik kayıtların oluşmasını önler. MS Access programında bu ayar tablonun Genel (General) bölümünde yapılır. Buradaki seçeneklerden biri olan Gerekli (Required) özelliği Evet (Yes) olarak seçilir



Resim 2: Zorunlu alan



Resim 3: Otomatik olarak görünecek değer

Varsayılan Değer

Bazı durumlarda çok sayıda kayıtta aynı başlangıç değeri bulunur. Kullanıcının aynı bilgiyi sürekli olarak girmesi yerine, varsayılan bir değer önceden belirlenebilir.

Örneğin, öğrencilerin çoğu mevcut eğitim-öğretim yılına aitse, yeni bir kayıt girilirken bu değer otomatik olarak görünebilir. Kullanıcı yalnızca gerektiğinde bu değeri değiştirebilir.

Veri Biçimi

Bazı bilgilerin standart bir şekilde görüntülenmesi gerekir. Tarihler, sayılar ve parasal değerler genellikle önceden belirlenmiş bir biçime sahiptir.

Bu özellik sayesinde aynı türdeki tüm veriler tek tip ve anlaşılır bir şekilde görüntülenir. Böylece bilgilerin okunabilirliği ve anlaşılması kolaylaşır.

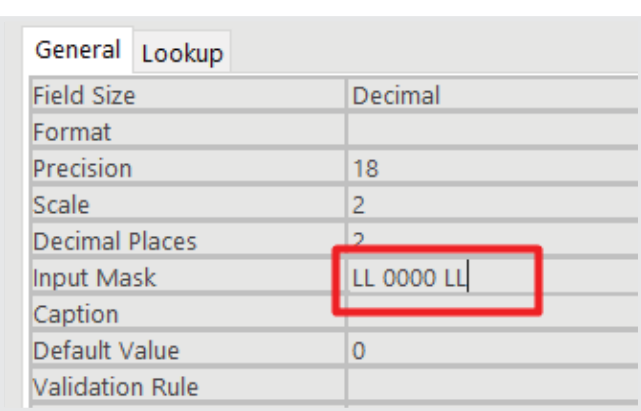
Veri Doğrulama

Her veritabanında kullanıcıların yanlışlıkla hatalı bilgiler girme ihtimali vardır. Sayı yerine metin girilebilir, tarih yerine rastgele bir değer girilebilir veya sistemin kurallarını karşılamayan bir veri girilebilir. Bu tür durumları önlemek için veri doğrulama kullanılır.

Veri doğrulama, girilen değerin önceden belirlenmiş koşulları karşılayıp karşılamadığını kontrol etme işlemidir. Değer kurallara uymuyorsa sistem kullanıcıyı uyarır ve hatalı bilgilerin kaydedilmesine izin vermez.

Örneğin, SMS mesajı aracılığıyla otopark ücretinin hesaplanacağı bir sistem tasarlanıyorsa, girilen plaka numarasının doğru formatta olup olmadığının doğrulanması gerekir. İnsanlar plaka numaralarını girerken sık sık hata yapabilir. Genel (General) sekmesindeki Input Mask (Giriş Maskesi) alanında bir doğrulama kuralı oluşturulabilir. Sisteme plaka formatında harf ve rakamların girilmesi bekleniyorsa, plakanın iki harf, ardından dört rakam ve son olarak iki harften oluşacağı belirtilir.

Burada "L", ilgili konumda yalnızca harf bulunabileceğini, "0" ise ilgili konumda rakam bulunması gerektiğini belirtir. Bu şekilde oluşturulan maske, plaka numarasının doğru girilip girilmediğini kontrol eder.



Resim 4: Kayıt maskesinin ayarlanması

Bu mekanizma, büyük miktarda veriyi işleyen bilgi sistemlerinde büyük önem taşır. Veri girişindeki küçük bir hata bile arama, analiz veya rapor hazırlama sırasında sorunlara yol açabilir. Bu nedenle modern veritabanları, girilen bilgileri kontrol etmek için çeşitli yöntemler kullanır.

Doğrulama Örnekleri

Aşağıdaki durumları inceleyelim:

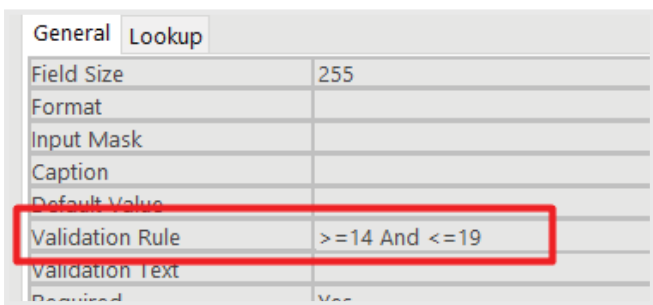
Örnek 1

Alan: Yaş

Kural: Değer 14 ile 19 arasında olmalıdır.

√ 16
√ 18
x -5
x 120

Bu durumda sistem yalnızca öğrencilerin yaşları için gerçekçi değerleri kabul edecektir.



Resim 5: Doğrulama kurallarının ayarlanması

Veri doğrulama bu şekilde yapılmalıdır.

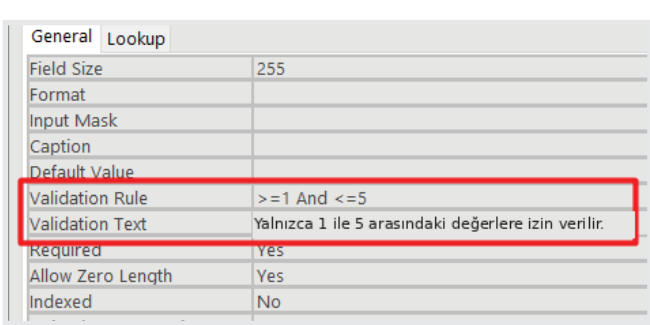
Örnek 2

Alan: Not

Kural: Yalnızca 1 ile 5 arasındaki değerlere izin verilir.

√ 1
√ 5
x 7
x 0

Bu kural, notlandırma sisteminde bulunmayan notların girilmesini önler.



Resim 6: Doğrulama kurallarının ve metninin ayarlanması

Doğrulama metni, alanın ne içermesi gerektiğini ayrıca açıklar. Doğrulama kullanıldığında doğrulama metninin de ayarlanması iyi bir uygulamadır. Program uyarı verdiğinde yalnızca bir ses duyulmaz, bu durumda hatayı açıklayan bir mesaj da görüntülenir.

Örnek 3

Alan: E-posta

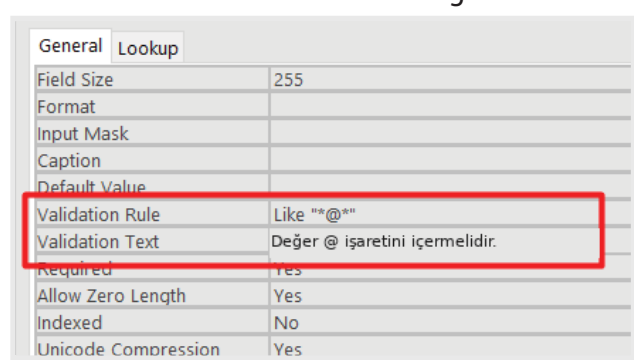
Kural: Değer @ işaretini içermelidir.

√ učenik@uciliste.mk

x ucenik.gmail.com

Doğrulama, veri girişi sırasında açıkça görülen hataların tespit edilmesine yardımcı olur.

Resim 7: Zorunlu karakter ile doğrulama



İstenen karakterin önünde ve arkasında bulunan yıldız işaretleri (*), bu karakterin öncesinde ve sonrasında mutlaka metin bulunması gerektiğini belirtir.

Örneklerin Analizi

Önceki örneklerden, veri doğrulamanın verileri güzelleştirmek için değil, veritabanını yanlış bilgilerden korumak için kullanıldığı görülebilir. Veri girişi yapan kullanıcıların sayısı arttıkça, değerlerin otomatik olarak kontrol edilmesine duyulan ihtiyaç da artar.

Veri türleri, alan özellikleri ve doğrulama kurallarının bir arada kullanılması, bilgilerle çalışmak için güvenilir bir temel oluşturur. Bu mekanizmalar sayesinde veriler daha tutarlı, daha doğru ve işlenmesi daha kolay hâle gelir.

SONUÇ:

Veri türü ve alan özellikleri, veritabanındaki bilgilerin kalitesi üzerinde büyük bir etkiye sahiptir. Doğru seçim, verilerin daha kolay işlenmesini, aranmasını ve analiz edilmesini sağlar.

Doğrulama ve alan özellikleri, girilen bilgilerin kontrol edilmesini sağlayan önemli mekanizmalardır. Hata olasılığı azaltmaya yardımcı olur ve verilerin daha güvenilir ve doğru olmasını sağlar.

BİLİYOR MUSUNUZ?

Bilgisayarın metinleri, sayıları ve tarihleri farklı şekillerde işlediğini biliyor musun? Veri türleri sayesinde sistem, belirli bir değer üzerinde hesaplama yapıp yapamayacağını veya yalnızca görüntülemesi gerekip gerekmediğini bilir.

TEKRARLAMA SORULARI

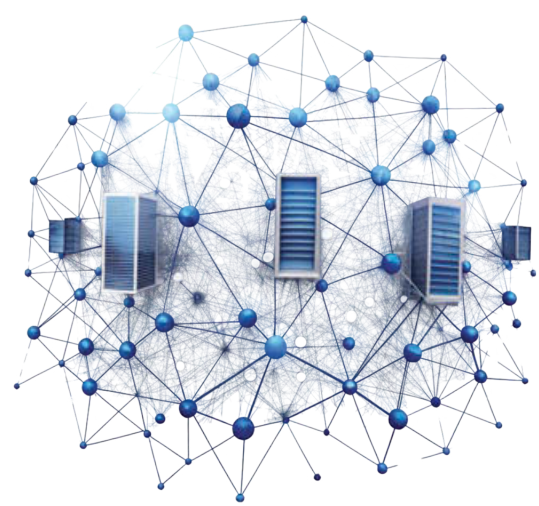
1. Veri türü nedir?
2. En sık kullanılan veri türleri nelerdir?
3. Metin veri türü ne zaman kullanılır?
4. Sayısal veri türü ne zaman kullanılır?
5. Veri doğrulama nedir?
6. Alan özellikleri neden kullanılır?
7. Varsayılan değer nedir?

PRATİK ALIŞTIRMALAR

Ödev 1

Her alan için uygun veri türünü belirleyiniz:

Alan
Ad
Doğum tarihi
Devamsızlık sayısı
Aktif öğrenci
Not



Ödev 2

Zorunlu giriş yapılmasını isteyeceğiniz üç alan belirtiniz.

Ödev 3

Varsayılan değere sahip olabilecek bir alan için örnek veriniz.

Ödev 4

Yaşın neden metin yerine sayısal veri türünde olması gerektiğini açıklayınız.

ARAŞTIRMA GÖREVİ

Microsoft Access ve LibreOffice Base programlarının kullandığı veri türlerini araştırınız. Karşılaştırma yapınız ve her iki programda ortak olan veri türlerini bulunuz.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Her alanın belirli bir veri türü vardır.

Veri türü, hangi değerlerin girilebileceğini belirler.

Alan özellikleri, veriler üzerinde kontrol sağlar.

Veri doğrulama, yanlış değerlerin girilmesini önler.

Veri türlerinin doğru seçilmesi, veritabanının verimliliğini artırır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Profesyonel sistemlerde düzinelerce farklı veri türü bulunmaktadır. Bunlardan bazıları fotoğrafları, videoları, coğrafi koordinatları veya karmaşık belgeleri depolamak için kullanılır.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Doğum tarihi için en uygun veri türü hangisidir?

a) Metin
b) Sayı
c) Tarih/Saat
d) Mantıksal

2. Veri doğrulama ne işe yarar?

a) Kayıtları silmek
b) Girilen verileri kontrol etmek
c) Raporları yazdırmak
d) Tablolar oluşturmak

Doğru veya yanlış

1. Yaş alanına herhangi bir metin girilebilir.
2. Mantıksal veri türünün genellikle iki olası değeri vardır.
3. Veri doğrulama, verilerin kalitesini artırır.

3.4

BİRİNCİL ANAHTAR VE TABLOLARIN BİRBİRİNE BAĞLANMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturursun;
- Farklı veri türlerini kullanırsın;
- Alan özelliklerini uygularsın;
- Doğrulama kurallarını kullanırsın;
- Bilgileri tablo biçiminde organize edersin.

DÜŞÜNME SORULARI

- İki kayıtta aynı bilgiler bulunursa ne olur?
- Adı ve soyadı aynı olan iki öğrenci nasıl kesin olarak birbirinden ayırt edilebilir?
- Her kaydın benzersiz bir tanımlayıcıya sahip olması neden gerekir?
- Bilgisayar gerekli kaydı nasıl hızlı bir şekilde bulur?
- Farklı tablolarda aynı bilgiler tekrarlanmadan nasıl kullanılabilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Birincil anahtarın ne olduğunu açıklarsın;
- Birincil anahtar için uygun alanları belirleyebilirsin;
- Benzersiz tanımlamanın önemini analiz edersin;
- Birincil anahtar kullanmanın avantajlarını açıklarsın;
- Tablo oluştururken birincil anahtar uygularsın;
- İlişkisel veritabanında verilerin organizasyonunu analiz edersin.

Çok sayıda kayıtla çalışırken, her kaydın kesin ve benzersiz bir şekilde tanımlanabilmesi gerekir. Bir tabloda öğrencilerle ilgili bilgilerin saklandığını varsayalım. İki öğrencinin aynı ad ve soyada sahip olması mümkündür. Bu nedenle yalnızca ad ve soyada bakarak hangi öğrenciden söz edildiği kesin olarak belirlenemez. Bu yüzden veritabanlarında her kaydı benzersiz şekilde tanımlamaya yarayan özel bir alan kullanılır.

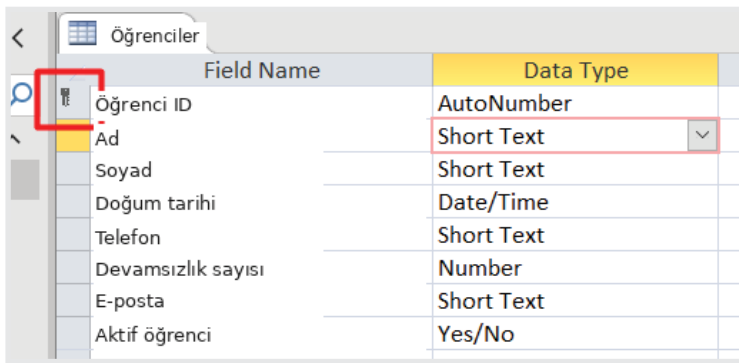
Her kayıt için benzersiz bir değer içeren alana birincil anahtar denir. Temel görevi, tablodaki her kaydın benzersiz ve kolayca tanınabilir olmasını sağlamaktır. Birincil anahtarın değeri iki farklı kayıtta tekrarlanmamalı ve boş bırakılamamalıdır.

Birincil anahtar, veritabanının doğru çalışması açısından büyük önem taşır. Kayıtların benzersiz şekilde tanımlanmasını sağlamasının yanı sıra, tabloların birbirine bağlanmasının ve karmaşık bilgi sistemlerinin organize edilmesinin temelini oluşturur. Bu nedenle tablo oluşturulurken hangi alanın bu görevi üstleneceği dikkatlice seçilmelidir.

3.4.1 BİRİNCİL ANAHTARIN ÖZELLİKLERİ

Bir alanın birincil anahtar olarak kullanılabilmesi için bazı koşulları karşılaması gerekir.

İlk olarak, değer her kayıt için benzersiz olmalıdır. İki kişinin aynı kimlik numarasına sahip olması durumunda sistem onları doğru şekilde ayırt edemez. İkinci olarak, değer boş olmamalıdır, çünkü her kaydın kendine ait benzersiz bir tanımlayıcısı olmalıdır. Üçüncü olarak, değerin sabit olması ve nadiren değiştirilmesi tercih edilir. Çünkü sık yapılan değişiklikler verilerin düzenlenmesi sırasında sorunlara yol açabilir.



Resim 1: Birincil anahtarın belirlenmesi

Birçok durumda birincil anahtar olarak, her yeni kayıt eklendiğinde otomatik olarak yeni bir değer alan özel bir alan kullanılır. Böylece kullanıcının yanlışlıkla daha önce kullanılmış bir değeri girme ihtimali ortadan kaldırılır.

Örnek

Aşağıdaki tabloyu inceleyelim.

Öğrenci ID	Ad	Soyad
1	Ana	Petrova
2	Ayet	Recepi
3	Ana	Petrova
4	John	Wayne

Tablodan "Ana Petrova" ad ve soyadının iki kez bulunduğu görülmektedir. Yalnızca ad ve soyad kullanılarak kimlik belirlenmeye çalışılsaydı, bu iki kaydı birbirinden ayırt etmek zor olurdu.

Öğrenci ID alanı her öğrenci için farklı bir değer içerdiğinden birincil anahtar olarak kullanılabilir. Kaç öğrencinin aynı ada veya soyada sahip olduğunun önemi yoktur, kimlik numaraları farklı olacaktır. Bu durumda öğrenciyi tanımlamak için, örneğin sınıf listesindeki numarası gibi benzersiz bir bilgi kullanılabilir. Bir sınıfta 3 numaralı yalnızca bir öğrenci olabilir. Eğitim sisteminde EİBU (Öğrencinin Benzersiz Kimlik Numarası), İçişleri Bakanlığında kimlik kartı numarası, Dışişleri Bakanlığında pasaport numarası, Sağlık Bakanlığında ise sağlık karnesi/sağlık kimlik numarası kullanılır. Tüm bu kayıtlar EMBG (vatandaşın benzersiz kimlik numarası) ile ilişkilendirilir. Yasal düzenlemeler nedeniyle herkesin bir kişinin EMBG bilgisine erişmesi mümkün değildir. Bu nedenle her kurum, aynı ad ve soyada sahip kişileri birbirinden ayırt etmek için kendine özgü bir yöntem oluşturur. Bir kuruluş veya şirket müşterilerine kart veriyor ve bu kart numaralarının ardışık olmasını istiyorsa, veri türü olarak Otomatik Sayı (Autonumber) kullanılır. Bu seçenek, mağazalardaki yazar kasalar için de oldukça kullanışlıdır, çünkü bu cihazlar günlük olarak yüzlerce mali fiş düzenleyebilir.

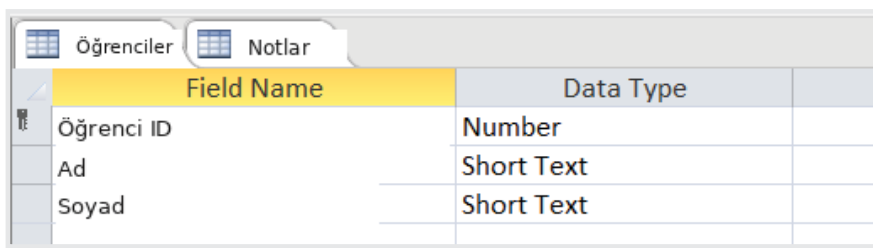
Örneğin Analizi

Önceki örnekten, birincil anahtarın kullanıcı açısından anlamlı olmak zorunda olmadığı görülebilir. Birincil anahtarın görevi, kayıtların benzersiz olmasını ve verilerin doğru şekilde organize edilmesini sağlamaktır. Uygulamada kullanıcılar çoğunlukla ad, adres veya diğer bilgileri görürken, sistem arka planda verileri güvenli bir şekilde yönetmek için birincil anahtarları kullanır.

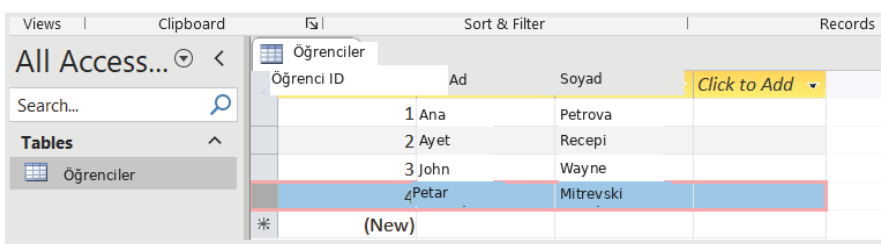
Birincil anahtar kullanılması, bilgilerin aranmasını, güncellenmesini ve yönetilmesini büyük ölçüde kolaylaştırır. Bu sayede her kayıt hızlı bir şekilde bulunabilir ve benzer bilgiler içeren diğer kayıtlarla karıştırılma riski olmadan işlenebilir.

3.4.2 TABLOLARIN BİRBİRİNE BAĞLANMASI

Daha büyük veritabanlarında bilgiler genellikle birden fazla tabloda organize edilir. Bu tabloların birlikte kullanılabilmesi için aralarında bir bağlantı bulunması gerekir. Tabloların birbirine bağlanması, bilgilerin daha verimli bir şekilde organize edilmesini ve gereksiz veri tekrarının önlenmesini sağlar.

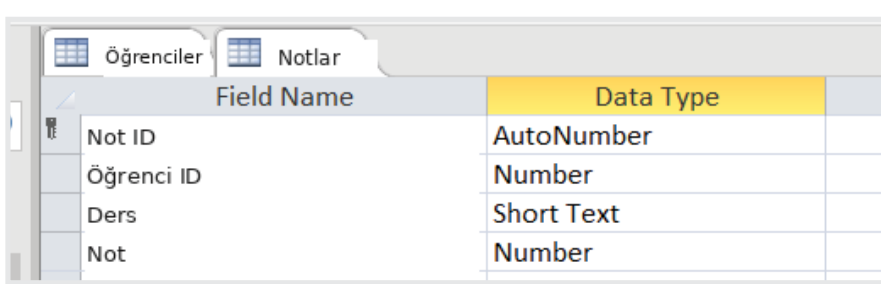


Resim 2: Öğrenci tablosunun düzenlenmesi

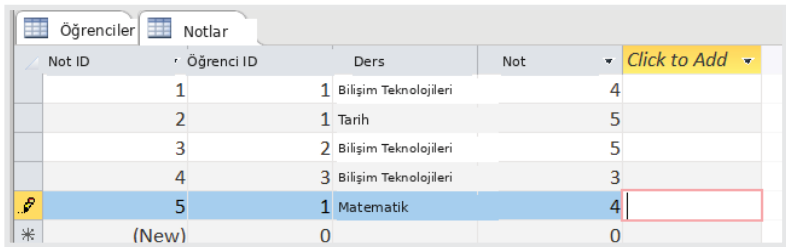


Resim 3: Öğrenci tablosu

Öğrencilerin bulunduğu bir tablo ve notların bulunduğu ayrı bir tablo olduğunu varsayalım. Her not kaydında öğrencinin adı ve soyadını tekrar yazmak yerine, öğrenciler tablosundaki kimlik numarası (ID) kullanılabilir. Böylece bilgiler yalnızca bir kez saklanır ve gerektiğinde kolayca birbirleriyle ilişkilendirilebilir.

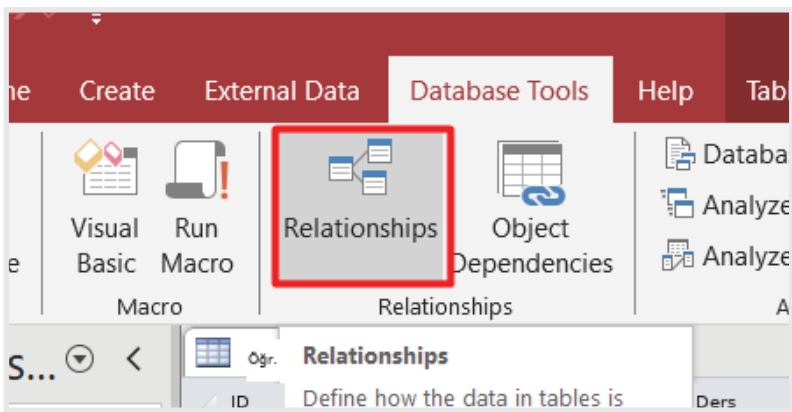


Resim 4: Not tablosunun düzenlenmesi



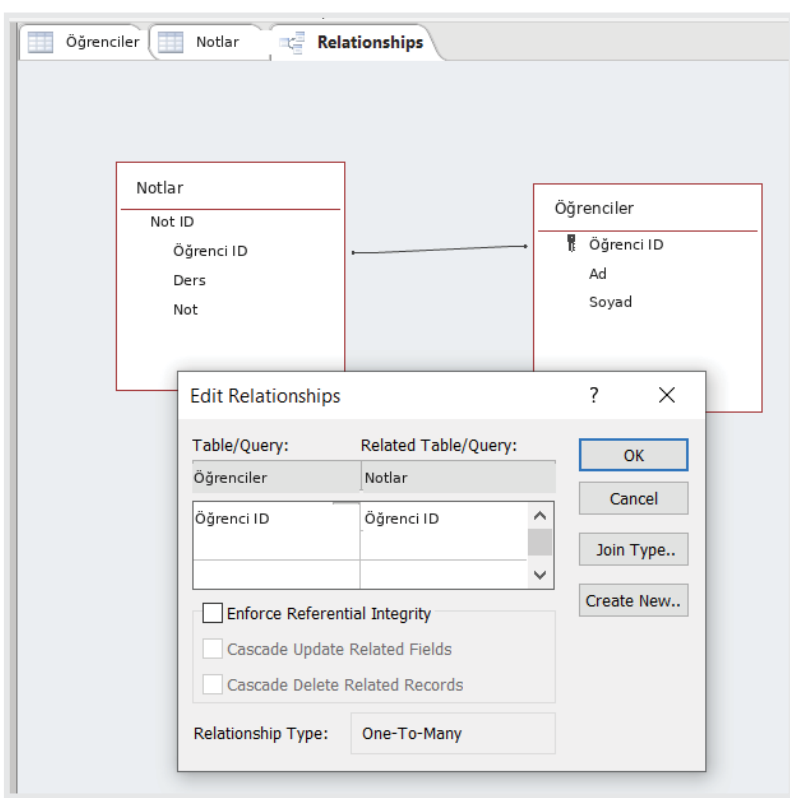
Resim 5: Not tablosunun doldurulması

Tabloların birbirine bağlanması, ilişkisel veritabanlarının en önemli özelliklerinden biridir. Bu sayede veriler düzenli kalır ve sistem büyük miktarda bilgiyle verimli bir şekilde çalışabilir.



Resim 6: Veritabanında ilişkilerin oluşturulması

İlişkiler oluşturulurken, birbirine bağlanacak alanların aynı veri türünde olması gerekir. Bu durumda iki tablo, Öğrenci ID alanı üzerinden birbirine bağlanacaktır.



Resim 7: Tablolar arasında ilişkilerin oluşturulması

Tabloların Düzenlenmesi ve Güncellenmesi

Veritabanları nadiren değişmeden kalır. Zamanla yeni alanlar ekleme, mevcut alanların özelliklerini değiştirme veya yeni kayıtlar ekleme ihtiyacı ortaya çıkar. Bu nedenle modern sistemler, tabloların kolayca düzenlenmesine ve güncellenmesine olanak sağlar.

Tablonun düzenlenmesi, yeni bir alan eklemeyi, mevcut bir alanın adını değiştirmeyi veya veri türünü değiştirmeyi içerebilir. Bu tür değişiklikler genellikle yeni gereksinimler ortaya çıktığında veya bilgilerin daha iyi organize edilmesi gerektiğinde yapılır. Tabloların güncellenmesi ise verilerin değiştirilmesi anlamına gelir. Örneğin veritabanında bulunan bir telefon numarası, adres veya başka bir bilgi değiştirilebilir. Verilerin doğru ve kullanılabilir kalması için düzenli olarak güncellenmesi gerekir.

İyi organize edilmiş tablolar, doğru seçilmiş alanlar ve verilerin dikkatli bir şekilde güncellenmesi, güvenilir ve verimli bir veritabanının temelini oluşturur. Bu işlemler sayesinde bilgiler doğru, düzenli ve kullanıcılar tarafından kolayca erişilebilir durumda kalır.

SONUÇ:

Birincil anahtar, ilişkisel veritabanlarının en önemli öğelerinden biridir. Temel görevi, her kaydın benzersiz ve kolayca tanınabilir olmasını sağlamaktır. Birincil anahtar sayesinde veriler güvenli bir şekilde organize edilebilir ve yönetilebilir.

Birincil anahtar kullanılması, bilgilerin aranmasını, güncellenmesini ve yönetilmesini daha verimli hâle getirir. Ayrıca daha büyük ve karmaşık veritabanlarının oluşturulmasının temelini oluşturur.

BİLİYOR MUSUNUZ?

İki kişinin aynı ad ve soyada sahip olabileceğini, ancak aynı kimlik numarasına sahip olamayacağını biliyor musun? Veritabanlarında da benzer bir prensip birincil anahtar aracılığıyla kullanılır.

TEKRARLAMA SORULARI

1. Birincil anahtar nedir?
2. Birincil anahtar hangi koşulları karşılamalıdır?
3. Birincil anahtar değerleri neden tekrarlanmamalıdır?
4. Birincil anahtar boş olabilir mi?
5. Birincil anahtar kullanmanın avantajları nelerdir?
6. Birincil anahtar veri arama işlemine nasıl yardımcı olur?

PRATİK ALIŞTIRMALAR

Ödev 1

Hangi alanın birincil anahtar için en uygun olacağını belirleyiniz:

| Şifre | Ürün | Miktar |

Ödev 2

Birincil anahtar olarak kullanılabilecek üç alan örneği veriniz.

Ödev 3

Ad ve soyadın neden her zaman birincil anahtar için iyi bir seçim olmadığını açıklayınız.

Ödev 4

İki kaydın birincil anahtar değerlerinin aynı olması durumunda ne olacağını düşününüz.

ARAŞTIRMA GÖREVİ

Farklı kurumların kişilerin, ürünlerin veya belgelerin benzersiz şekilde tanımlanmasını nasıl sağladığını araştırınız. En az üç örnek bulunuz ve bunların nasıl çalıştığını açıklayınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Birincil anahtar, kayıtların benzersiz şekilde tanımlanmasını sağlar.
- Birincil anahtar değerleri tekrarlanmamalıdır.
- Birincil anahtar boş değer içeremez.
- Verilerin aranmasını ve güncellenmesini kolaylaştırır.
- İlişkisel veritabanlarının düzenlenmesinde önemli bir unsurdur.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Büyük veri tabanlarında birincil anahtarlar genellikle otomatik olarak oluşturulur. Bu sayede sistem, her yeni kaydın benzersiz bir kimlik numarası almasını garanti eder.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Birincil anahtar:

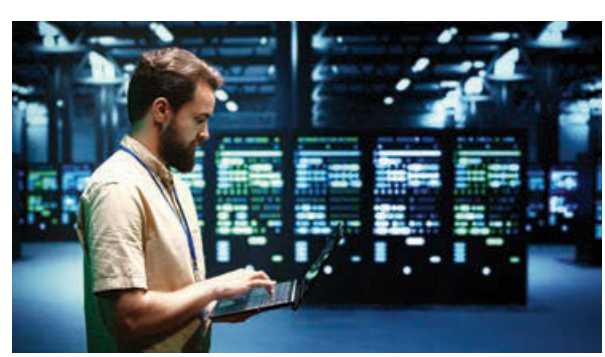
a) Tekrarlanabilir
b) Benzersiz olmalıdır
c) Önemli değildir
ç) Biçimlendirme amacıyla kullanılır

2. Birincil anahtar çoğunlukla şu amaçla kullanılır:

a) Kayıtların tanımlanması
b) Raporların yazdırılması
c) Tabloların silinmesi
ç) Metin girilmesi

Doğru mu, Yanlış mı?

1. İki farklı kayıt aynı birincil anahtara sahip olabilir.
2. Birincil anahtar boş olabilir.
3. Birincil anahtar, verilerin düzenlenmesine yardımcı olur.



3.5

VERİ GİRİŞİ VE VERİLERİ GÖRÜNTÜLEME FORMLARI

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturabileceksiniz;
- Farklı veri türlerini kullanabileceksiniz;
- Alanların özelliklerini belirleyebileceksiniz;
- Birincil anahtar kullanabileceksiniz;
- Tablolara veri girebilecek ve verileri güncelleyebileceksiniz.

DÜŞÜNME SORULARI

1. Kullanıcının verileri her zaman doğrudan tablolara girmesi pratik midir?
2. Bilgi girişi nasıl daha kolay hâle getirilebilir?
3. Tüm kullanıcıların tablonun yapısını görmesi gerekir mi?
4. Veriler daha düzenli ve anlaşılır bir şekilde nasıl görüntülenebilir?
5. Veri girişinin elektronik bir form doldurmaya benzemesi mümkün müdür?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Formun ne olduğunu açıklayabileceksiniz;
- Form kullanmanın avantajlarını sıralayabileceksiniz;
- Basit bir form oluşturabileceksiniz;
- Form aracılığıyla veri girebileceksiniz;
- Mevcut kayıtları düzenleyebileceksiniz;
- Formların bilgi sistemlerindeki kullanımını analiz edebileceksiniz.

Önceki derslerde veriler doğrudan tablolara giriliyordu. Bu yaklaşım basit olmakla birlikte her zaman en pratik yöntem değildir. Büyük veritabanlarında kullanıcıların çoğu zaman tablonun tamamını görmesine veya tüm alanlarla aynı anda çalışmasına gerek yoktur. Bunun yerine, yalnızca gerekli bilgileri görüntüleyen özel olarak hazırlanmış bir ekran üzerinden veri girmek çok daha pratiktir.

İşte bu nedenle veritabanlarında formlar kullanılır. Formlar, verilerin daha düzenli ve kontrollü bir şekilde girilmesini, görüntülenmesini ve güncellenmesini sağlar. Kullanıcı, satır ve sütunlarla çalışmak yerine elektronik bir formu andıran alanlar, düğmeler, listeler ve diğer öğelerle çalışır.

Günlük yaşamda farklı türlerde formlarla sık sık karşılaşırız. Çevrim içi başvuru yapmak, kullanıcı hesabı oluşturmak, sipariş bilgilerini girmek veya bir sınava elektronik olarak başvurmak, arka planda bir veritabanıyla iletişim kuran formlara örnek olarak verilebilir.

Tablodan Forma

Bir öğrenci tablosunu inceleyelim.

Öğrenci ID	Ad	Soyad	Telefon	E-posta
1	Ana	Petrova	071123456	ana@email.com
2	Marko	Stoyanov	075654321	marko@email.com

İlk bakışta tablo gerekli tüm bilgileri içermektedir. Ancak kullanıcının her gün yeni öğrenciler eklemesi gerekiyorsa, bu görünüm her zaman en pratik yöntem olmayabilir. Özellikle tablonun on veya yirmi alan içerdiği durumlarda bu durum daha belirgin hâle gelir.

Bu nedenle verileri daha basit bir şekilde görüntüleyecek bir form oluşturulabilir.

Form Kullanmanın Avantajları

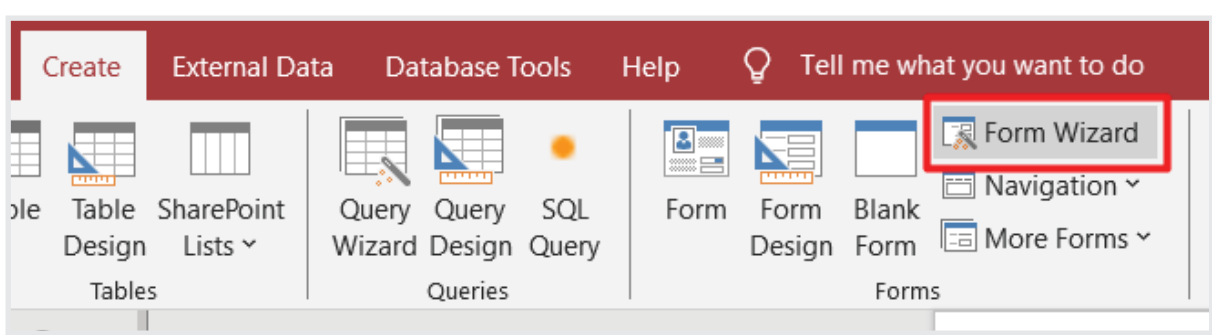
Formlar, kullanıcı ile veritabanı arasındaki etkileşimi önemli ölçüde kolaylaştırır. Bilgi girişini basitleştirir ve hata yapma olasılığını azaltır. Kullanıcının büyük bir tablo içerisinde hareket etmesi yerine, dikkati yalnızca girilmesi gereken verilere yönlendirilir.

Bir diğer avantaj ise öğelerin daha iyi düzenlenebilmesidir. Birbiriyle ilişkili bilgiler gruplandırılabilir, seçim düğmeleri, açılır listeler veya işaretleme kutuları kullanılabilir. Böylece form daha düzenli, anlaşılır ve kullanımı kolay hâle gelir.

Formlar ayrıca belirli bilgilere erişimin sınırlandırılmasına da olanak sağlar. Kullanıcı, tabloyu doğrudan açmadan form üzerinden çalışabilir. Bu durum, verilerin daha güvenli olmasına ve daha iyi kontrol edilmesine katkı sağlar.

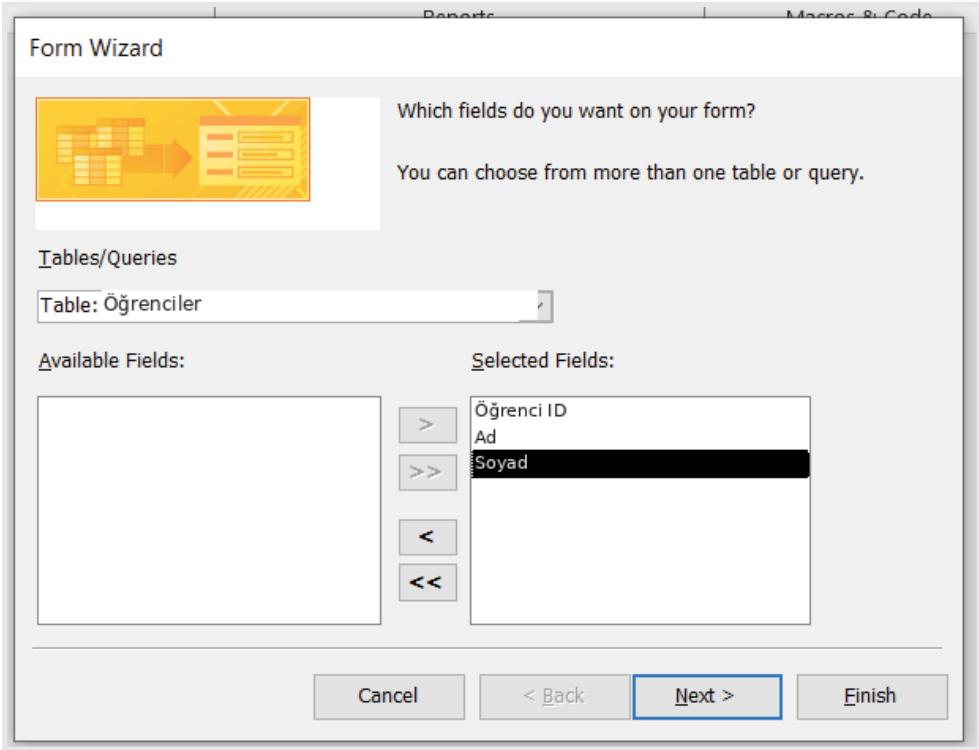
Form Oluşturma

Microsoft Access ve LibreOffice Base programlarında, mevcut bir tablo temel alınarak otomatik olarak form oluşturulmasını sağlayan araçlar bulunmaktadır. Kullanıcı öncelikle veri kaynağı olarak kullanılacak tabloyu seçer. Daha sonra program, tablodaki alanları otomatik olarak forma yerleştirir.



Resim 1 Formun etkinleştirilmesi

Form oluşturulduktan sonra, görünümü üzerinde ek düzenlemeler yapılabilir. Başlıklar, öğelerin boyutları, konumları ve verilerin görüntülenme biçimi değiştirilebilir. Böylece form, kullanıcıların belirli ihtiyaçlarına göre uyarlanabilir.



Resim 2: Tablo ve tablodaki alanların seçilmesi

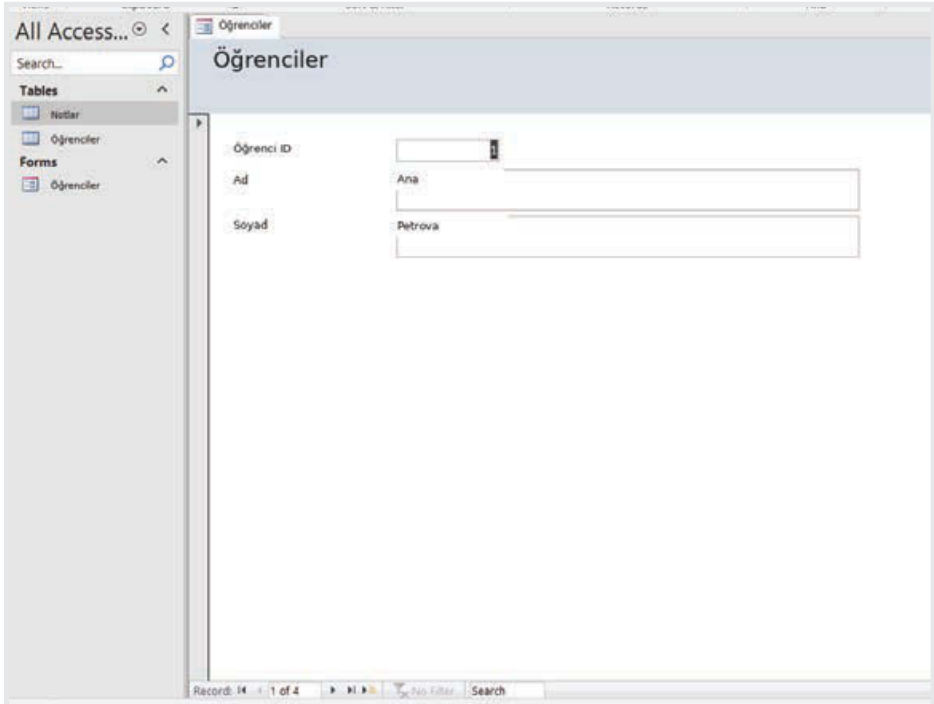
Gerekli tüm alanlar, ilgili tablodan forma aktarılır. Veritabanında birden fazla tablo varsa, öncelikle tablo seçilir, ardından formda görüntülenmesi gereken alanlar seçilir.

Form Aracılığıyla Veri Girme

Form oluşturulduktan sonra, veritabanına yeni veriler girmek için kullanılabilir. Kullanıcının artık tabloyu doğrudan açmasına ve bilgileri girmek için uygun satırı bulmasına gerek yoktur. Bunun yerine, formda görüntülenen alanları doldurması yeterlidir.

Yeni bir kayıt girilirken her bilgi, ilgili alana girilir. Herhangi bir alan için doğrulama veya kısıtlama belirlenmişse program, girilen değerin belirlenen koşulları karşılayıp karşılamadığını otomatik olarak kontrol eder. Böylece yanlış veya eksik veri girme olasılığı azaltılır.

Formlar, bilgi giriş sürecini klasik bir tabloyla çalışmaktan ziyade elektronik bir form doldurmaya daha yakın hâle getirir. Bu özellik, özellikle veritabanını ileri düzeyde bilişim bilgisine sahip olmayan kişilerin kullandığı durumlarda oldukça yararlıdır.



Resim 3: MS Access'te Form

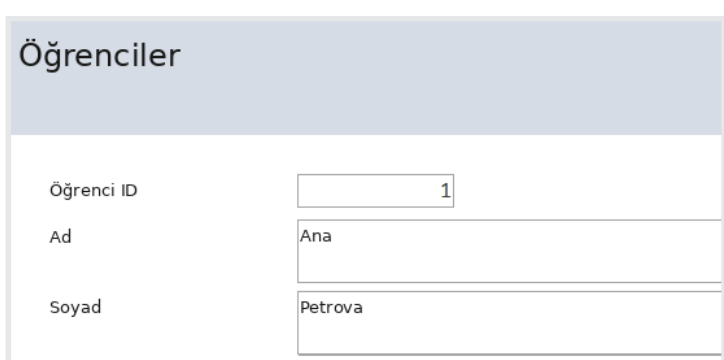
Form, Form Sihirbazı kullanılarak oluşturulduktan sonra, varsa tüm kayıtlar görüntülenir.

Önceki örnekte de görüldüğü gibi kullanıcı artık satır ve sütunları görmez. Bunun yerine her veri, ayrı bir alana girilir. Bu çalışma biçimi, internette günlük olarak kullandığımız elektronik formların doldurulmasına daha yakındır.

Mevcut Kayıtların Düzenlenmesi

Veritabanına girilen veriler her zaman değişmeden kalmaz. Telefon numaraları değişebilir, kullanıcılar yeni e-posta adresleri alabilir ve bazı bilgilerin düzenli olarak güncellenmesi gerekebilir. Formlar, bu tür değişikliklerin kolay ve hızlı bir şekilde yapılmasını sağlar.

Kullanıcı form üzerinden mevcut bir kaydı açtığında, tüm bilgiler doğrudan değiştirilebilen alanlarda görüntülenir. Değişiklik yapıldıktan sonra yeni bilgiler otomatik olarak veritabanına kaydedilir. Böylece kullanıcının tabloda ilgili bilgiyi aramasına gerek kalmaz.



Resim 4: Mevcut kaydın değiştirilmesi

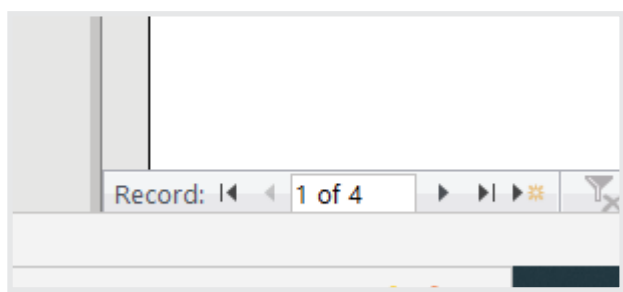
Form üzerinden düzenleme yapmak, bilgilerin daha düzenli görüntülenmesini ve daha iyi kontrol edilmesini sağlar. Kullanıcı, tabloda bulunan diğer verilerle uğraşmak zorunda kalmadan yalnızca üzerinde çalıştığı kayda odaklanır.

Kayıtlar Arasında Gezinme

Bir form genellikle yalnızca tek bir kaydı görüntülemez. Form, tabloda bulunan tüm kayıtlar arasında gezinmeye olanak sağlar. Bunun için kullanıcıya önceki, sonraki, ilk veya son kayda geçme imkânı veren gezinme düğmeleri kullanılır.

Bu özellik, özellikle tablonun çok sayıda kayıt içerdiği durumlarda oldukça kullanışlıdır. Kullanıcının verileri tek tek aramasına gerek kalmaz, form üzerindeki gezinme araçlarını kullanması yeterlidir. Böylece bilgilerin görüntülenmesi daha hızlı ve kolay hâle gelir.

Birçok modern bilgi sisteminde formlar, veri arama ve filtreleme gibi ek özellikler de içerir. Bu işlevler sonraki derslerde daha ayrıntılı olarak ele alınacak olsa da formların yalnızca bilgi girmek için değil, aynı zamanda bilgileri etkili bir şekilde kullanmak için de hizmet ettiğini bilmek önemlidir.



Resim 5: Kayıtlar Arasında Gezinme

Uygulamalı Örnek: Öğrenci Kayıtlarının Takibi

Bir okulda öğrencilerin kayıtlarını tutmak için basit bir sistem oluşturulması gerektiğini varsayalım. Veritabanında öğrencilerle ilgili gerekli bilgileri içeren bir tablo zaten bulunmaktadır. Öğretmenin veya idari personelin doğrudan tabloyla çalışması yerine, verilerin daha kolay girilmesini ve görüntülenmesini sağlayacak bir form oluşturulabilir.

Formda ad, soyad, telefon numarası, e-posta adresi ve sınıf/şube gibi alanlar bulunabilir. Yeni bir öğrenci eklenirken kullanıcı alanları doldurur ve sistem kaydı otomatik olarak veritabanına kaydeder. Daha sonra herhangi bir bilginin değiştirilmesi gerektiğinde kullanıcı, ilgili kaydı form üzerinden açarak gerekli değişikliği kolayca yapabilir.

Bu yaklaşım, çalışma verimliliğini önemli ölçüde artırır. Kullanıcı, çok sayıda satır ve sütundan oluşan karmaşık bir tablo yerine yalnızca ihtiyaç duyduğu bilgileri gösteren basit ve düzenli bir ekranla çalışır. Bu nedenle formlar, modern veritabanlarında en sık kullanılan öğelerden biri hâline gelmiştir.

Örneğin Analizi

Önceki örnekten de görülebileceği gibi formlar, kullanıcı ile veritabanı arasında bir aracı görevi görür. Formlar tabloların yerini almaz, tablolarla çalışmayı kolaylaştırır. Tüm veriler yine tablolarda saklanır, ancak kullanıcı veritabanıyla çoğunlukla formlar aracılığıyla etkileşim kurar.

Formların kullanılması, verilerin daha düzenli ve anlaşılır görüntülenmesini sağlar, hata yapma olasılığını azaltır ve verilerin girilmesini ve güncellenmesini kolaylaştırır. Bu avantajlar sayesinde formlar, veritabanlarına dayalı bilgi sistemlerinin geliştirilmesinde önemli bir role sahiptir.

SONUÇ:

Formlar, veritabanlarının önemli bir unsuru olup kullanıcı ile veriler arasındaki etkileşimi kolaylaştırır. Kullanıcı, tablolarla doğrudan çalışmak yerine, bilgileri girebileceği, görüntüleyebileceği ve güncelleyebileceği düzenli bir arayüz kullanır.

Formların kullanılması, veri girişi sırasında hata yapma olasılığını azaltırken aynı zamanda verilerin daha düzenli ve anlaşılır görüntülenmesini ve çalışma verimliliğinin artmasını sağlar. Bu nedenlerle formlar, modern veritabanlarında en sık kullanılan unsurlardan biridir.

BİLİYOR MUSUNUZ?

Kayıt olma, giriş yapma veya çevrim içi sipariş verme işlemlerinde kullanılan her web sayfası aslında bir form kullanır. Kullanıcı yalnızca veri giriş alanlarını görse de girilen bu bilgiler, arka planda bir veritabanına kaydedilir.

TEKRARLAMA SORULARI

1. Veritabanında form nedir?
2. Formların tablolarla doğrudan çalışmaya göre avantajları nelerdir?
3. Formlar en çok hangi amaçlarla kullanılır?
4. Form aracılığıyla veri nasıl girilir?
5. Mevcut kayıtlar nasıl güncellenir?
6. Kayıtlar arasında gezinme nedir?
7. Formlar tabloların yerini alır mı?

PRATİK ALIŞTIRMALAR

Ödev 1
Mevcut bir tabloyu açınız ve tablodaki tüm alanları görüntüleyecek bir form oluşturunuz.

Ödev 2
Form aracılığıyla en az beş yeni kayıt giriniz.

Ödev 3
Formu kullanarak mevcut iki kaydı değiştiriniz.

Ödev 4
Gezinme düğmelerini kullanarak tüm kayıtları inceleyiniz.

Ödev 5
Formu kullanıcılar için daha düzenli ve anlaşılır hâle getirecek üç iyileştirme öneriniz.

ARAŞTIRMA GÖREVİ

İnternette her gün kullanılan elektronik form örneklerini araştırınız.

En az üç örnek bulunuz ve analiz ediniz:

- Bunların amacı nedir;
- Hangi veriler girilir;
- Alanlar nasıl düzenlenmiştir;
- Hangi veri giriş kuralları uygulanır.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Form, verilerle çalışmak için kullanılan bir kullanıcı arayüzüdür.
- Formlar, kayıtların girilmesini, görüntülenmesini ve güncellenmesini sağlar.
- Veriler yine tablolarda saklanır.
- Formlar, verilerin daha düzenli görüntülenmesini sağlar ve çalışmayı kolaylaştırır.
- Gezinme öğeleri, kayıtlar arasında hareket etmeyi sağlar.
- Formlar, modern bilgi sistemlerinde yaygın olarak kullanılmaktadır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

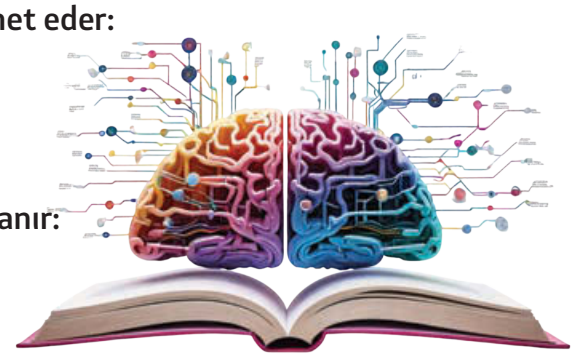
Basit formların yanı sıra, modern sistemler düğmeler, seçim listeleri, resimler, otomatik hesaplamalar ve birden fazla tablodan alınan ilişkili veriler içeren gelişmiş formların oluşturulmasına da olanak sağlar. Profesyonel uygulamalarda formlar, çoğu zaman kullanıcı ile veritabanı arasındaki temel iletişim aracıdır.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Form aşağıdaki amaçlardan hangisine hizmet eder:

a) Veritabanını silmek
b) Veri girmek ve görüntülemek
c) İşletim sistemini biçimlendirmek
ç) Dosyaları sıkıştırmak



2. Form aracılığıyla girilen veriler nerede saklanır:

a) Ana bellekte
b) Formda
c) Tabloda
ç) Raporda

Doğru mu, Yanlış mı?

1. Formlar verilerle daha kolay çalışılmasını sağlar.
2. Veriler formun kendisinde saklanır.
3. Formlar kayıtları güncellemek için kullanılabilir.
4. Gezinme düğmeleri kayıtlar arasında hareket etmeyi sağlar.

ANALİZ

Bir okulun veritabanında 500 öğrencinin bulunduğu bir tablo olduğunu düşününüz.

Açıklayınız:

- Form kullanmanın tabloyla doğrudan çalışmaktan neden daha pratik olduğunu;
- Kullanıcının hangi avantajları elde ettiğini;
- Formun yeni bir öğrenci eklenmesine nasıl yardımcı olabileceğini.

3.6

SORGULAR (QUERIES) – VERİLERİ ARAMA VE SEÇME

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturabileceksiniz;
- Farklı veri türlerini kullanabileceksiniz;
- Birincil anahtar uygulayabileceksiniz;
- Formlar aracılığıyla veri girebileceksiniz;
- Kayıtları görüntüleyebilecek ve güncelleyebileceksiniz.

DÜŞÜNME SORULARI

1. Veritabanında binlerce kayıt varsa ve bunlardan yalnızca birine ihtiyacınız varsa ne yaparsınız?
2. Bir sınıftaki tüm öğrenciler nasıl bulunabilir?
3. Yalnızca başarı düzeyi çok iyi olan öğrenciler nasıl görüntülenebilir?
4. Her zaman tüm kayıtları incelemek gerekli midir?
5. Bilgisayar gerekli bilgileri nasıl hızlı bir şekilde bulabilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Sorgunun ne olduğunu açıklayabileceksiniz;
- Verileri aramak için sorguları kullanabileceksiniz;
- Kayıtları seçmek için koşullar uygulayabileceksiniz;
- Bilgileri belirlenen ölçütlere göre sıralayabileceksiniz;
- Sorgudan elde edilen sonuçları analiz edebileceksiniz;
- Basit sorgular oluşturabileceksiniz.

Giriş

Gerçek veritabanlarında çoğu zaman yüzlerce, binlerce hatta milyonlarca kayıt saklanır. Tüm veriler önemli olmakla birlikte kullanıcıların uygulamada çoğunlukla yalnızca belirli bir bilgi bölümüne ihtiyacı vardır. Örneğin bir öğretmen yalnızca belirli bir sınıftaki öğrencileri, bir kütüphaneci yalnızca o anda ödünç verilmiş kitapları, okul yönetimi ise gezi ücretini henüz ödememiş öğrencileri görmek isteyebilir.

Belirli bir bilgi gerektiğinde tablonun tamamını incelemek yavaş ve verimsiz olacaktır. Bu nedenle veritabanı yönetim sistemleri, bilgileri aramak ve seçmek için özel mekanizmaların kullanılmasına olanak sağlar. Bu mekanizmalara sorgular (queries) denir.

Sorgular, veritabanından yalnızca belirlenen koşulları karşılayan bilgilerin seçilmesini sağlayan bir araçtır. Kullanıcının tüm kayıtları tek tek aramasına gerek kalmaz, sistem gerekli kayıtları otomatik olarak bulur ve yalnızca ihtiyaç duyulan sonuçları görüntüler.

Sorgu Nedir?

Sorgu, veritabanı yönetim sistemine yöneltilen bir istektir. Kullanıcı sorgu aracılığıyla hangi verileri almak istediğini ve bu verilerin hangi koşullara göre seçileceğini belirler.

Tüm verileri içeren tablonun aksine, sorgu çoğunlukla verilerin yalnızca bir bölümünü görüntüler. Örneğin yalnızca belirli bir sınıftaki öğrenciler, not ortalaması 4,50'nin üzerinde olan öğrenciler veya yarışmalara katılan öğrenciler görüntülenebilir.

Sorgunun tablodaki verileri değiştirmediğinin vurgulanması önemlidir. Sorgu yalnızca belirlenen koşulları karşılayan bilgileri görüntüler. Sorgu kapatıldıktan sonra tablodaki veriler değişmeden kalır.

Uygulamalı Örnek

Aşağıdaki tabloyu inceleyelim.

Ad	Soyad	Sınıf	Not Ortalaması
Ana	Petrova	I-1	5.00
Marko	Stoyanov	I-2	4.20
Elena	Nikolovska	I-1	4.80
Merita	Recepi	I-3	3.90

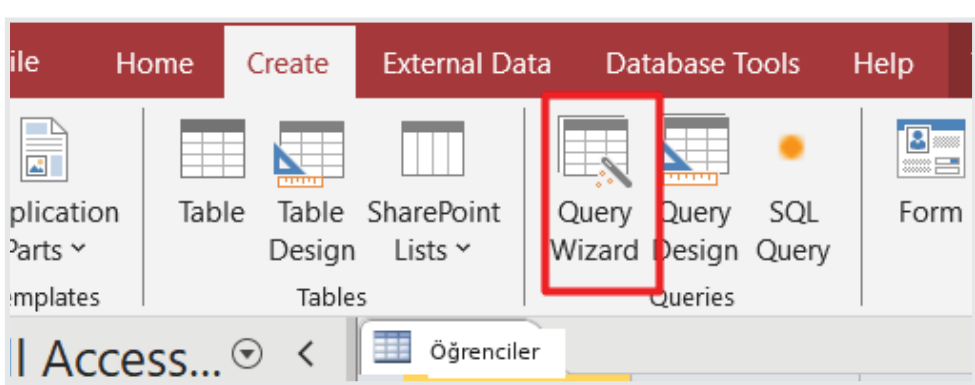
Yalnızca I-1 sınıfındaki öğrencilerin görüntülenmesi gerektiğini varsayalım.

Tabloyu manuel olarak aramak yerine, şu koşulu belirleyen bir sorgu oluşturulabilir:

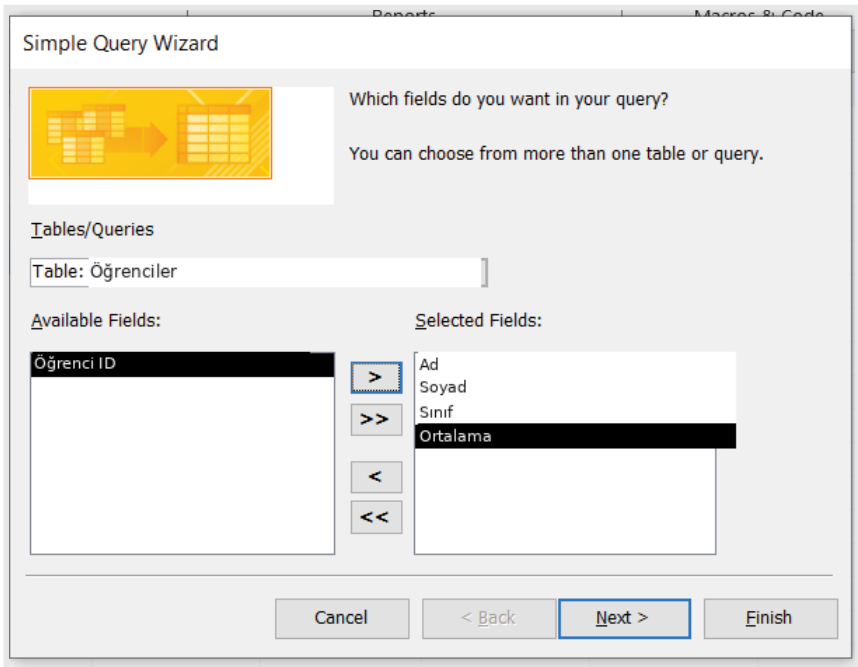
Sınıf = I-1

Sonuç olarak şu kayıtlar elde edilir:

Ad	Soyad	Sınıf	Not Ortalaması
Ana	Petrova	I-1	5.00
Elena	Nikolovska	I-1	4.80

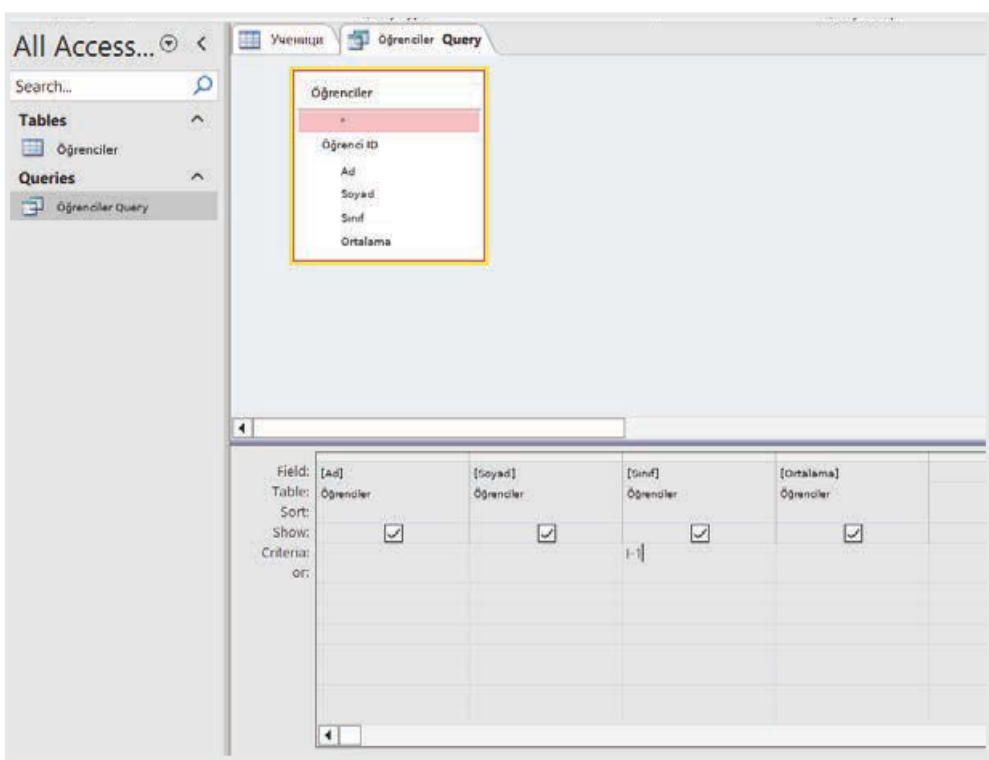


Resim 1: MS Access'te Sorgu



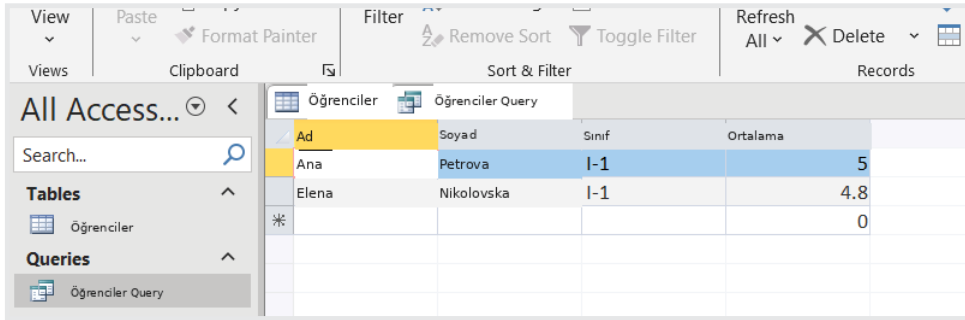
Resim 2 Sorguda parametrelerin belirlenmesi

Tüm öğeleri içeren daha önce oluşturulmuş sorguda, Tasarım Görünümü üzerinden gerekli ölçütler belirlenebilir.



Resim 3: Sorguda ölçüt belirleme

Bu durumda I-1 sınıfındaki öğrenciler seçilir ve "Sınıf" alanının Ölçüt bölümüne I-1 yazılır. Sorgu "Çalıştır (Run)" komutu seçilerek çalıştırıldığında, yalnızca belirtilen sınıftaki öğrenciler görüntülenir.



Resim 4: Sorguda Belirlenen Ölçütün Etkisi

Hazırlanan bu sorgu, veritabanından istenen verileri getirir. Bir veritabanında, veritabanının farklı özelliklerine yönelik çok sayıda sorgu oluşturulabilir.

Sorgular Neden Kullanılır?

Sorgular, veritabanlarının kullanımını önemli ölçüde kolaylaştırır. Kullanıcının çok sayıda kayıtla tek tek çalışması yerine sistem, gerekli bilgileri otomatik olarak seçer. Bu yalnızca zaman kazandırmakla kalmaz, aynı zamanda hata yapma olasılığını da azaltır.

Bunun yanı sıra sorgular, verilerin farklı şekillerde analiz edilmesine olanak sağlar. Aynı tablo, belirlenen koşullara bağlı olarak farklı bilgiler elde etmek için kullanılabilir. Örneğin bir sorguda öğrenciler sınıflarına göre, başka bir sorguda başarı durumlarına göre, üçüncü bir sorguda ise devamsızlık sayılarına göre aranabilir.

Bu özellikleri sayesinde sorgular, veritabanlarıyla çalışırken en yararlı araçlardan biridir ve çeşitli bilgi sistemlerinde yaygın olarak kullanılmaktadır.

Sorgularda Koşullar

Sorguların en önemli avantajlarından biri, verilerin hangi koşullara göre seçileceğinin belirlenebilmesidir. Kullanıcı, tablodaki tüm kayıtları görüntülemek yerine yalnızca ihtiyaç duyduğu bilgileri belirleyebilir. Bu, veritabanında çok sayıda kayıt bulunduğunda bile gerekli bilgilerin hızlı bir şekilde bulunmasını sağlar.

Koşul, bir kaydın sonuçlarda görüntülenebilmesi için karşılaması gereken kuraldır. Örneğin belirli bir sınıftaki tüm öğrenciler, belirli bir yazara ait tüm kitaplar veya miktarı belirli bir değerden az olan tüm ürünler istenebilir. Sistem kayıtları otomatik olarak kontrol eder ve yalnızca koşulu karşılayan kayıtları görüntüler.

Uygulamada farklı türlerde koşullar kullanılabilir. Bazı koşullar bir değerin belirli bir veriyle eşit olup olmadığını, bazıları ise değerin daha büyük, daha küçük olup olmadığını veya belirli bir aralıkta bulunup bulunmadığını kontrol eder. Bu esneklik sayesinde sorgular, kullanıcıların farklı ihtiyaçlarına göre uyarlanabilir.

Koşul Kullanımına Örnek

Aşağıdaki tabloyu inceleyelim.

Ad	Soyad	Ortalama
Ana	Petrova	5.00
Marko	Stoyanov	4.20
Elena	Nikolovska	4.80
Merita	Recepi	3.90

Yalnızca not ortalaması 4.50'den yüksek olan öğrencilerin görüntülenmesi gerektiğini varsayalım.

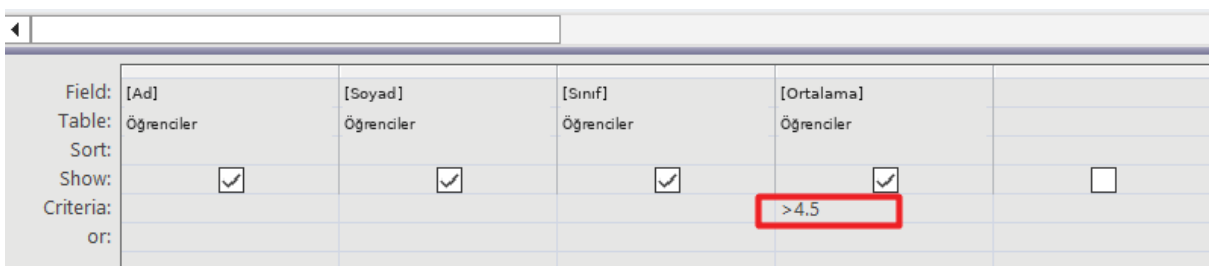
Koşul şu şekilde olacaktır:

Not Ortalaması > 4.50

Sonuç olarak aşağıdaki kayıtlar görüntülenecektir:

Ad	Soyad	Not Ortalaması
Ana	Petrova	5.00
Elena	Nikolovska	4.80

Örnekten de görülebileceği gibi sistem, belirlenen koşulu karşılayan kayıtları otomatik olarak seçmiştir.



Resim 5: Not Ortalaması İçin Koşul Belirleme

Sonuçların Sıralanması

Verileri aramanın yanı sıra sorgular, verilerin sıralanmasına da olanak sağlar. Sıralama, sonuçların belirli bir düzene göre düzenlenmesidir. Genellikle bilgilerin daha kolay incelenmesi ve analiz edilmesi amacıyla kullanılır.

Veriler alfabetik sıraya, sayısal değerlere veya tarihlere göre sıralanabilir. Örneğin öğrenci listesi soyadına göre, ürün listesi fiyatına göre, kitap listesi ise yayın yılına göre sıralanabilir.

Sıralama, tablodaki verileri değiştirmez. Yalnızca sorgu sonuçlarının görüntülenme biçimini etkiler. Böylece kullanıcı, bilgileri daha hızlı ve daha anlaşılır bir şekilde inceleyebilir.

Sıralama Örneği

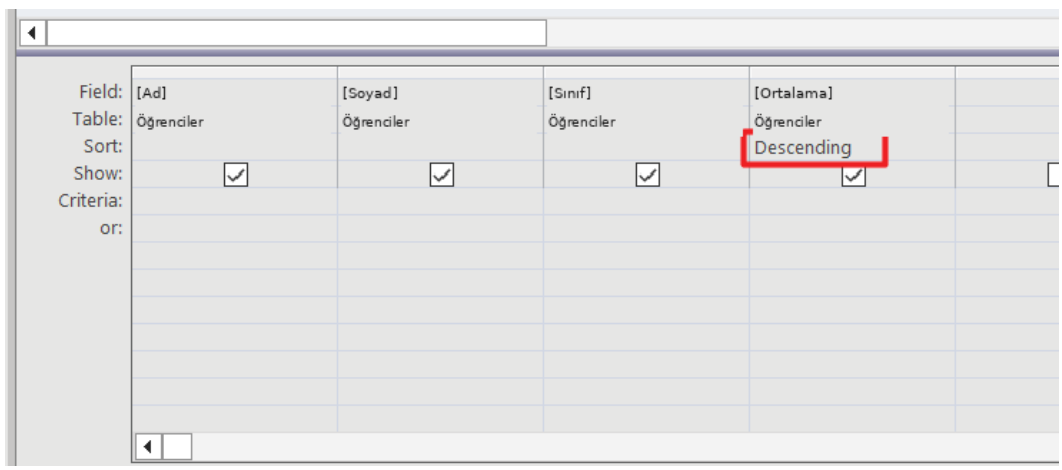
Aşağıdaki sonucu inceleyelim.

Ad	Not Ortalaması
Marko	4.20
Ana	5.00
Merita	3.90
Elena	4.80

Not Ortalamasına göre azalan sırada sıralama uygulanırsa sonuç şu şekilde olacaktır:

Ad	Not Ortalaması
Ana	5.00
Elena	4.80
Marko	4.20
Merita	3.90

Bu şekilde en yüksek değerler listenin başında görüntülenir.



Resim 6: Sonuçların sıralanması

Sorgunun Tasarım Görünümünde yapılan her değişiklikten sonra, sorgunun güncel durumunu görmek için "Çalıştır (Run)" komutunun etkinleştirilmesi gerekir.

Birden fazla koşulun birleştirilmesi.

Birçok durumda, gerekli bilgileri elde etmek için tek bir koşul yeterli değildir. Bu nedenle sorgular, aynı anda birden fazla koşulun kullanılmasına olanak sağlar. Örneğin belirli bir sınıfa ait olan ve aynı zamanda belirli bir değerin üzerinde not ortalamasına sahip öğrenciler aranabilir.

I-1 sınıfındaki not ortalaması 4,50'den yüksek olan öğrencilerin bulunması gerektiğini varsayalım.

İki koşul gereklidir:

- Sınıf = I-1
- Not Ortalaması > 4.50

Yalnızca her iki koşulu da karşılayan kayıtlar sonuçlarda görüntülenir.

Koşulların birleştirilmesi, bilgilerin çok daha kesin ve doğru bir şekilde aranmasını sağlar. Koşullar ne kadar doğru ve ayrıntılı belirlenirse elde edilen sonuçlar da o kadar ilgili olacaktır.

Uygulamalı Örnek: Okul Veritabanı

Bir okulun birinci sınıftaki tüm öğrencilerin kayıtlarını tuttuğunu varsayalım. Okul müdürü, not ortalaması 4,50'den yüksek olan ve aynı zamanda üçten fazla mazeretsiz devamsızlığı bulunmayan öğrencilerin listesini almak istemektedir.

üzlerce kaydı tek tek incelemek yerine, her iki koşulu da karşılayan öğrencileri otomatik olarak seçen bir sorgu oluşturulabilir. Sonuç, herhangi bir ek işlem yapmaya gerek kalmadan birkaç saniye içinde elde edilir.

Bu tür durumlarla eğitim, sağlık, bankacılık ve kamu yönetimi alanlarında günlük olarak karşılaşılır. Sorgular sayesinde kullanıcılar, büyük miktardaki veriler arasından doğru bilgilere hızlı bir şekilde ulaşabilir. Bu nedenle sorgular, veritabanlarıyla çalışmada kullanılan en güçlü mekanizmalardan biridir.

Örneğin Analizi

Önceki örneklerden, sorguların yalnızca veri bulmak için değil, aynı zamanda verileri analiz etmek için de kullanıldığı sonucuna varılabilir. Sorgular, gerekli bilgilerin seçilmesine, farklı ölçütlere göre sıralanmasına ve çeşitli koşulların birleştirilmesine olanak sağlar. Böylece kullanıcı, tablonun tamamıyla çalışmak yerine yalnızca ilgili verileri elde eder.

Koşullar belirleme ve sonuçları sıralama imkânı sayesinde sorgular, veritabanlarıyla çalışmanın verimliliğini önemli ölçüde artırır. Sorgular, modern bilgi sistemlerinde vazgeçilmez bir araçtır ve bilgilerin aranması ve analiz edilmesi amacıyla günlük olarak kullanılır.

Screenshot 5 – Birden Fazla Ko□ul □çeren Sorgu

Screenshot 6 – Sorgu Sonuçları

SONUÇ:

Sorgular, veritabanlarıyla çalışmada önemli bir araçtır, çünkü bilgilerin hızlı bir şekilde bulunmasını, seçilmesini ve analiz edilmesini sağlar. Kullanıcının tablodaki tüm kayıtları tek tek araması yerine sistem, belirlenen koşulları karşılayan verileri otomatik olarak seçer.

Sorgular kullanılarak farklı arama ölçütleri uygulanabilir, birden fazla koşul birleştirilebilir ve sonuçlar kullanıcının ihtiyaçlarına göre sıralanabilir. Böylece çalışma verimliliği artırılır ve doğru bilgilere dayanarak karar verme süreci kolaylaştırılır.

Pratik kullanım alanları sayesinde sorgular, modern veritabanlarının en önemli mekanizmalarından biri olup hemen hemen tüm bilgi sistemlerinde kullanılmaktadır.

BİLİYOR MUSUNUZ?

İnternet arama motorları da sorgulara benzer bir çalışma prensibi kullanır. Bir arama terimi girdiğinizde sistem, çok büyük miktardaki veri içinde otomatik olarak arama yapar ve yalnızca girdiğiniz arama isteğine uygun sonuçları görüntüler.

TEKRARLAMA SORULARI

1. Veritabanında sorgu nedir?
2. Tablo ile sorgu arasındaki fark nedir?
3. Sorguda koşul neyi ifade eder?
4. Sonuçlar neden sıralanır?
5. Sorgu, tablodaki verileri değiştirir mi?
6. Birden fazla koşulun birleştirilmesi neden yararlıdır?
7. Sorgular en çok hangi durumlarda kullanılır?

PRATİK ALIŞTIRMALAR

Ödev 1

Seçilen bir sınıftaki tüm öğrencileri görüntüleyecek bir sorgu oluşturunuz.

Ödev 2

Not ortalaması 4,50'den yüksek olan öğrencileri görüntüleyecek bir sorgu oluşturunuz.

Ödev 3

Bir önceki sorgunun sonuçlarını not ortalamasına göre azalan sırada sıralayınız.

Ödev 4

I-1 sınıfındaki ve not ortalaması 4,50'den yüksek olan öğrencileri görüntüleyecek bir sorgu oluşturunuz.

Ödev 5

Beşten fazla devamsızlığı olan öğrencileri görüntüleyecek bir sorgu oluşturunuz.

ARAŞTIRMA ÖDEVİ

Okulda öğrenciler, öğretmenler, dersler ve notlarla ilgili verilerin bulunduğunu düşününüz.

Müdürün sorguları kullanarak yanıtlayabileceği en az beş soru öneriniz.

Her soru için aşağıdakileri belirtiniz:

- Hangi verilere ihtiyaç duyulduğu;
- Hangi koşulların belirlenmesi gerektiği,
- Nasıl bir sonuç beklendiği.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Sorgu, verileri aramak ve seçmek için kullanılan bir araçtır.
- Sorgular, yalnızca ihtiyaç duyulan bilgilerin görüntülenmesini sağlar.
- Koşullar, hangi kayıtların görüntüleneceğini belirler.
- Aynı anda birden fazla koşul kullanılabilir.
- Sonuçlar farklı ölçütlere göre sıralanabilir.
- Sorgular, tablodaki verileri değiştirmez.
- Sorgular, veritabanlarıyla çalışmanın verimliliğini önemli ölçüde artırır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern veritabanı yönetim sistemlerinde istatistiksel hesaplamalar yapabilen, verileri gruplandırabilen ve birden fazla ilişkili tablodaki bilgileri analiz edebilen gelişmiş sorgular bulunmaktadır. Büyük bilgi sistemlerinde bu sorgular, raporların hazırlanması, satışların analiz edilmesi, finansal verilerin izlenmesi ve iş kararlarının alınmasına destek sağlanması amacıyla sıklıkla kullanılır.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Sorgu şu amaçla kullanılır:

a) Tabloları biçimlendirmek
b) Verileri aramak ve seçmek
c) Programları yüklemek
ç) İşletim sistemi oluşturmak

2. Sorgudaki koşul:

a) Kayıtları siler
b) Hangi kayıtların görüntüleneceğini belirler
c) Tablonun yapısını değiştirir
ç) Veritabanını kapatır

3. Sıralama şu amaçla kullanılır:

a) Sonuçları sıralamak
b) Sonuçları silmek
c) Formlar oluşturmak
ç) Yeni alanlar eklemek

Doğru mu, Yanlış mı?

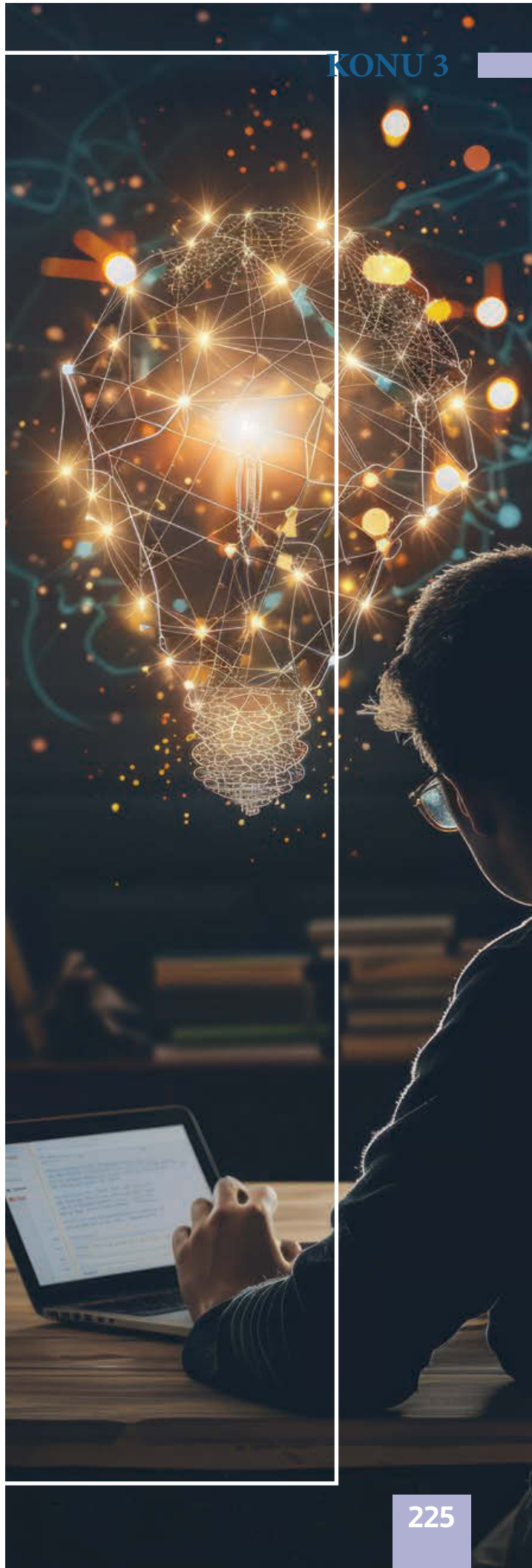
1. Sorgular aynı anda birden fazla koşul kullanabilir.
2. Sorgu sonuçları sıralanabilir.
3. Sorgu her zaman tablodaki tüm kayıtları görüntüler.
4. Sorgu tablodaki verileri değiştirir.

ANALİZ

Bir kütüphane veritabanında kitaplar, yazarlar ve kullanıcılarla ilgili bilgilerin saklandığını düşününüz.

Açıklayınız:

- Belirli bir yazara ait tüm kitapları görüntülemek için nasıl bir sorgu oluştururdunuz;
- Hangi koşulu belirlerdiniz;
- Sonuçları daha düzenli ve anlaşılır hâle getirmek için nasıl sıralardınız.



3.7

RAPORLAR (REPORTS) – VERİLERİN GÖRÜNTÜLENMESİ VE YAZDIRILMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturabileceksiniz;
- Farklı veri türlerini kullanabileceksiniz;
- Formlar aracılığıyla veri girebileceksiniz;
- Bilgi aramak için sorgular oluşturabileceksiniz;
- Sonuçlara koşullar uygulayabilecek ve sonuçları sıralayabileceksiniz;
- Veritabanından elde edilen verileri analiz edebileceksiniz.

DÜŞÜNME SORULARI

1. Veritabanındaki veriler yazdırılmak üzere nasıl hazırlanabilir?
2. Bilgileri görüntülemek için tablo her zaman en iyi yöntem midir?
3. Yazdırılmaya hazır bir öğrenci listesi nasıl oluşturulabilir?
4. Kurumlar yaptıkları çalışmalarla ilgili raporları nasıl hazırlar?
5. Sorgu sonuçları nasıl daha düzenli ve anlaşılır bir şekilde sunulabilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Raporun ne olduğunu açıklayabileceksiniz;
- Rapor kullanmanın avantajlarını sıralayabileceksiniz;
- Basit bir rapor oluşturabileceksiniz;
- Raporlarda tablo ve sorgulardan elde edilen verileri kullanabileceksiniz;
- Raporun görünümünü düzenleyebileceksiniz;
- Raporların bilgi sistemlerindeki kullanımını analiz edebileceksiniz.

Verilerle çalışırken bilgiler tablolarda düzenlenir, formlar aracılığıyla veri girişi yapılabilir ve sorgular yardımıyla veriler aranabilir. Ancak birçok durumda elde edilen bilgilerin daha düzenli bir şekilde görüntülenmesi veya yazdırılmak üzere hazırlanması gerekir. Örneğin bir öğretmen öğrenci listesini yazdırmak, bir kütüphaneci ödünç verilen kitaplarla ilgili bir rapor hazırlamak, okul müdürü ise sınıflara göre başarı durumunu incelemek isteyebilir.

Tablolar ve sorgular gerekli verileri içerse de görünümleri her zaman sunum veya yazdırma için uygun değildir. Bu nedenle veritabanı yönetim sistemleri raporların kullanılmasına olanak sağlar. Raporlar, bilgilerin düzenli ve profesyonel bir şekilde sunulması amacıyla kullanılır.

Rapor, veritabanındaki verileri görüntüleme, yazdırma veya paylaşma için hazırlanmış biçimde sunan özel bir veritabanı nesnesidir. Bir rapor, başlıklar, alt başlıklar, gruplandırılmış bilgiler, sayfa numaraları ve verilerin daha anlaşılır olmasını sağlayan diğer öğeleri içerebilir.

Rapor Nedir?

Rapor, veritabanında bulunan verilerin düzenli bir şekilde sunulmasıdır. Bilgi girişi ve düzenleme amacıyla kullanılan formlardan farklı olarak raporlar, verilerin görüntülenmesi ve sunulması amacıyla hazırlanır.

Raporlar, bilgileri doğrudan tablolardan veya daha önce oluşturulmuş sorgulardan alabilir. Böylece kullanıcı, belirli bir analiz veya sunum için gerekli olan yalnızca belirli verileri görüntüleyebilir.

Uygulamada raporlar, listeler, istatistiksel özetler, analiz sonuçları, finansal raporlar ve açık ve düzenli bir şekilde sunulması gereken diğer belgelerin hazırlanmasında sıklıkla kullanılır.

Uygulamalı Örnek

Okulun I-1 sınıfındaki öğrencilerin bir listesini hazırlamak istediğini varsayalım.

Veriler öncelikle bir tabloda saklanır. Daha sonra bir sorgu aracılığıyla yalnızca seçilen sınıftaki öğrenciler belirlenir.

Elde edilen sonuçlar şu şekilde görünebilir:

Ad	Soyad	Sınıf
Ana	Petrova	I-1
Elena	Nikolovska	I-1
Merita	Recepi	I-1

Bilgiler doğru olsa da bu görünüm yazdırma ve arşivleme için en uygun değildir. Bu nedenle başlık, hazırlanma tarihi ve düzenli bir şekilde sıralanmış bilgiler içeren bir rapor oluşturulur.

Öğrenciler

Öğrenci ID	Ad	Soyad	Sınıf	Ortalama	Click to Add
1	Ana	Petrova	I-1	5	
2	Marko	Stoyanov	I-2	4.2	
3	Elena	Nikolovska	I-1	4.8	
4	Merita	Recepi	I-1	5	
(New)				0	

Resim 1: Raporun Kaynak Verileri

Raporların Avantajları

Raporlar, bilgilerin daha düzenli ve profesyonel bir şekilde sunulmasını sağlar. Başlıklar, logolar, sayfa numaraları ve gruplandırılmış veriler gibi okunabilirliği artıran çeşitli öğeler içerebilir.

Bir diğer avantajı ise bilgilerin yazdırılabilmesidir. Kullanıcının tabloları veya sorgu sonuçlarını doğrudan yazdırması yerine, bu amaç için özel olarak hazırlanmış bir rapor oluşturması mümkündür.

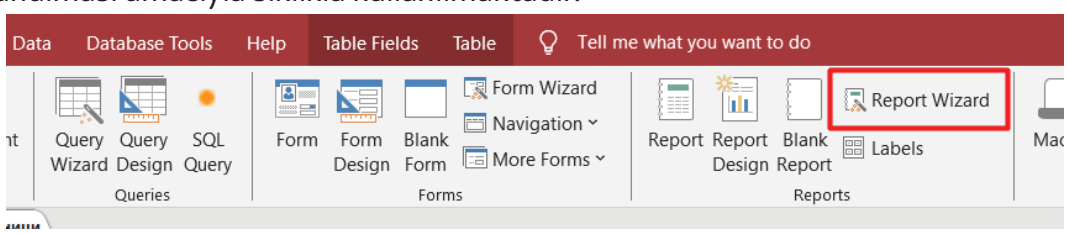
Raporlar ayrıca bilgilerin daha etkili bir şekilde sunulmasını sağlar. Veriler farklı ölçütlere göre düzenlenebilir ve kullanıcı ihtiyaç duyduğu bilgileri kolayca bulabilir.

Rapor Oluşturma

Microsoft Access ve LibreOffice Base programlarında raporların otomatik olarak oluşturulmasını sağlayan araçlar bulunmaktadır. Kullanıcı öncelikle tabloyu veya sorguyu veri kaynağı olarak seçer. Daha sonra program, seçilen verilere dayanarak temel bir rapor oluşturur.

Rapor oluşturulduktan sonra üzerinde çeşitli düzenlemeler yapılabilir. Verilerin yerleşimi değiştirilebilir, başlıklar eklenebilir ve bilgilerin görüntülenme biçimi belirlenebilir. Böylece rapor, kullanıcının ihtiyaçlarına uygun şekilde düzenlenebilir.

Eğitim kurumlarında raporlar, öğrenci listelerinin hazırlanması, başarı durumlarının raporlanması, devamsızlık istatistiklerinin oluşturulması ve çalışma sürecinde düzenli olarak kullanılan diğer verilerin sunulması amacıyla sıklıkla kullanılmaktadır.



Resim 2: Report Wizard ile rapor oluşturma

Raporun Görünümünün Düzenlenmesi

Rapor oluşturulduktan sonra görünümü daha da geliştirilebilir. Kullanıcı, alanların yerleşimini değiştirebilir, başlık ve alt başlıklar ekleyebilir, sayfa numaraları ekleyebilir ve yazdırmaya uygun bir düzen seçebilir.

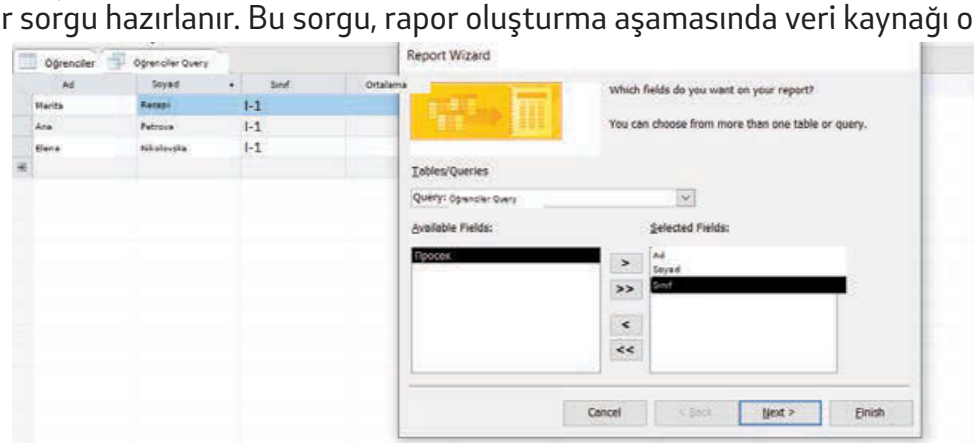
İyi düzenlenmiş bir rapor, verilerin daha kolay okunmasını ve analiz edilmesini sağlar. Özellikle çok sayıda kayıt içeren raporlarda, doğru bir düzenleme yapılması raporun kullanılabilirliği açısından büyük önem taşır.

Birçok durumda raporlarda, hazırlanma tarihi, kurumun adı veya içeriğin kısa bir açıklaması gibi ek bilgiler de bulunur. Bu unsurlar, belgenin daha düzenli ve profesyonel bir görünüme sahip olmasına katkı sağlar.

Uygulamalı Örnek: Öğrencilerin Başarı Durumlarına İlişkin Rapor

Öğrenciler arasından not ortalaması 4,50'den yüksek olanların bir raporunun hazırlanması gerektiğini varsayalım.

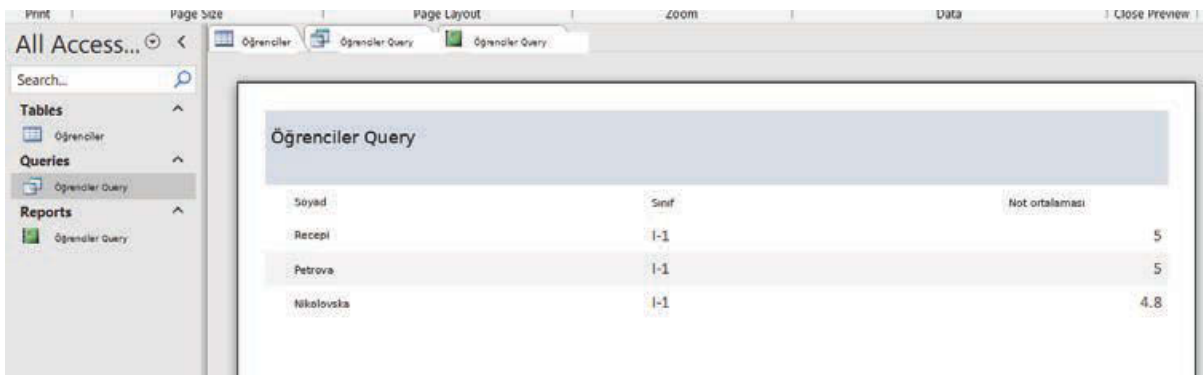
Öncelikle, belirtilen koşulu sağlayan öğrencileri seçen bir sorgu oluşturulur. Daha sonra elde edilen sonuçlar raporun temelini oluşturur. İlk olarak, ölçütün not ortalamasının 4,50 olması şeklinde belirlendiği bir sorgu hazırlanır. Bu sorgu, rapor oluşturma aşamasında veri kaynağı olarak seçilir.



Resim 2: Raporun hazırlanması

Raporda her öğrencinin adı, soyadı, sınıfı ve not ortalaması gösterilebilir. Ayrıca rapora "Üstün Başarı Gösteren Öğrenciler" başlığı ile belgenin hazırlanma tarihi de eklenebilir.

Bu şekilde elde edilen bilgiler daha düzenli ve anlaşılır hâle gelir ve yazdırılmaya veya öğretmenler kuruluna, sınıf öğretmenlerine ya da okul yönetimine sunulmaya hazır duruma getirilir.



Resim 3 Yazdırılmaya hazır rapor

Örneğin Analizi

Önceki örnekten de görülebileceği gibi rapor, yeni bir veri kaynağı oluşturmaz, var olan bilgilerin düzenli bir şekilde sunulmasını sağlar. Veriler tablolarda ve sorgularda bulunmaya devam ederken rapor, bu verilerin sunulmasına hizmet eder.

Raporların kullanılması, bilgilerin okunabilirliğini artırır ve günlük çalışmalarda daha kolay kullanılmasını sağlar. Kullanıcının karmaşık tablolar ve sorgularla çalışması yerine, incelemeye ve yazdırmaya hazır bir belge elde etmesine olanak tanır.

Bu nedenlerle raporlar, modern veritabanlarının önemli bir parçasıdır ve eğitim, yönetim, sağlık, finans kurumları ve daha birçok alanda geniş bir kullanım alanına sahiptir.

SONUÇ:

Raporlar, veritabanlarının önemli bir unsurunu oluşturur ve bilgilerin düzenli ve anlaşılır bir biçimde sunulmasını sağlar. Verilerin saklanması ve aranması amacıyla kullanılan tablolardan ve sorgulardan farklı olarak raporlar, bilgilerin sunulması ve yazdırılması amacıyla kullanılır.

Raporlar kullanılarak kurum ve kuruluşların günlük çalışmalarında yararlanılan çeşitli listeler, analizler ve bilgi özetleri hazırlanabilir. Görünümün düzenlenebilmesi sayesinde raporlar, bilgilerin açık ve profesyonel bir biçimde sunulmasını sağlar.

Pratik kullanım alanlarının geniş olması nedeniyle raporlar, her veritabanının önemli bir parçasıdır. Ayrıca incelenmesi, arşivlenmesi veya yazdırılması gereken belgelerin hazırlanmasında sıklıkla kullanılır.

BİLİYOR MUSUNUZ?

Biliyor muydun? Farklı kurumlarda müdürlere, yöneticilere ve birim sorumlularına sunulan belgelerin büyük bir bölümü, veritabanları tarafından otomatik olarak oluşturulmaktadır. Raporlar elle hazırlanmak yerine, sistemde zaten bulunan veriler temel alınarak birkaç saniye içinde oluşturulabilmektedir.

TEKRARLAMA SORULARI

1. Veritabanında rapor neyi ifade eder?
2. Form ile rapor arasındaki fark nedir?
3. Bir rapor için veriler hangi kaynaklardan elde edilebilir?
4. Raporlar ne için kullanılır?
5. Verilerin yazdırılmasında raporların avantajları nelerdir?
6. Raporun görünümü nasıl geliştirilebilir?
7. Raporlar en çok hangi alanlarda kullanılır?

PRATİK ALIŞTIRMALAR

Ödev 1

Öğrenciler tablosunu temel alarak bir rapor oluşturunuz.

Raporda aşağıdaki bilgiler gösterilmelidir:

- Ad;
- Soyad;
- Sınıf.

Ödev 2

Ortalaması 4,50'den yüksek olan öğrencileri gösterecek bir sorgu oluşturunuz ve bu sorguyu temel alarak bir rapor hazırlayınız.

Ödev 3

Rapora aşağıdaki başlığı ekleyiniz:

"Üstün Başarı Gösteren Öğrencilerin Listesi"

Ödev 4

Rapora hazırlanma tarihini ve sayfa numaralarını ekleyiniz.

Ödev 5

Raporu yazdırınız veya PDF formatında dışa aktarınız.

ARAŞTIRMA GÖREVİ

Okulların raporları hangi durumlarda kullandığını araştırınız.

En az beş örnek veriniz ve aşağıdaki noktaları açıklayınız:

- Raporun amacı nedir?
- Hangi veriler kullanılır?
- Bu raporları kimler kullanır?

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Rapor, verilerin düzenli bir şekilde sunulmasını sağlar.
- Raporlar, tablolardan ve sorgulardan elde edilen verileri kullanabilir.
- Raporlar, bilgilerin incelenmesi, yazdırılması ve sunulması amacıyla kullanılır.
- Raporun görünümü daha sonra düzenlenebilir.
- Raporlar yeni veriler oluşturmaz, mevcut bilgilerin bir sunumudur.
- Raporlar, modern bilişim sistemlerinde yaygın olarak kullanılmaktadır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Profesyonel bilgi sistemlerinde raporlar genellikle grafikler, istatistiksel analizler ve otomatik hesaplamalar içerir. Örneğin, bir finansal rapor gelir ve giderleri otomatik olarak hesaplayabilirken, bir satış raporu farklı dönemler arasındaki eğilimleri ve karşılaştırmaları gösterebilir.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Rapor en çok hangi amaçla kullanılır?
a) Veri girmek
b) Verileri görüntülemek ve yazdırmak
c) İşletim sistemi oluşturmak
ç) Tabloları silmek

2. Rapor aşağıdaki kaynaklardan hangilerinden veri kul
a) Tablo
b) Sorgu
c) Tablo ve sorgu
ç) Yukarıdakilerin hiçbiri

3. Raporlar hangi durumda kullanılır?
a) Verilerin sunum için düzenlenmesi gerektiğinde
b) Bir programın yüklenmesi gerektiğinde
c) Bir işletim sistemi oluşturulması gerektiğinde
ç) Bir diskin biçimlendirilmesi gerektiğinde

Doğru veya Yanlış
1. Raporlar yazdırılabilir.
2. Raporlar yeni kayıtların girilmesi için kullanılır.
3. Raporlar başlıklar ve sayfa numaraları içerebilir.
4. Raporlar sorgulardan elde edilen verileri kullanabilir.

ANALİZ

Bir okul veritabanında üstün başarı gösteren öğrencilerin bir listesinin hazırlanması gerekmektedir.

Aşağıdaki soruları açıklayınız:

- Hangi verilere ihtiyaç duyulacaktır?
- Öncelikle bir sorgu oluşturulması gerekir mi?
- Listenin yazdırılması için rapor neden sıradan bir tabloya göre daha uygun bir çözümdür.

3.8

VERİTABANINA DAYALI TOPLU MEKTUPLARIN HAZIRLANMASI

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturmayı;
- Verileri girmeyi ve güncellemeyi;
- Verilerle çalışmak için formları kullanmayı;
- Bilgi aramak için sorgular oluşturmayı;
- Raporlar hazırlamayı;
- Veritabanındaki verileri düzenlemeyi öğreneceksiniz.

DÜŞÜNME SORULARI

1. Aynı belge çok sayıda farklı kişiye nasıl gönderilebilir?
2. Her davetiyeye ad ve soyadı elle yazmak pratik midir?
3. Bir okul, öğrenciler için yüzlerce belgeyi nasıl yazdırabilir?
4. Şirketler müşterilerine kişiselleştirilmiş mektupları nasıl gönderir?
5. Bir belgenin her alıcı için otomatik olarak uyarlanması mümkün müdür?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

1. Toplu mektubun ne olduğunu açıklamayı;
2. Bir belgede veritabanındaki verileri kullanmayı;
3. Bir belgeyi veri kaynağına bağlamayı;
4. Birleştirme alanlarını eklemeyi;
5. Tek bir şablondan birden fazla belge oluşturmayı;
6. Toplu mektupların uygulamadaki kullanımını analiz etmeyi.

Birçok durumda aynı belgenin çok sayıda alıcıya gönderilmesi gerekir. Örneğin okul, veli toplantısı davetleri, öğrenciler için belgeler veya yarışmalara katılanlara teşekkür belgeleri gönderebilir. Belgenin içeriği büyük ölçüde aynı olsa da ad, soyadı veya adres gibi bazı bilgiler her alıcı için farklıdır.

Her belge ayrı ayrı hazırlanacak olursa bu işlem çok zaman alır ve hata yapma olasılığını artırır. Bu nedenle modern programlar, bir belgenin veritabanındaki veya tablodaki verilerle otomatik olarak birleştirilmesine olanak sağlar. Böylece tek bir şablon kullanılarak çok sayıda kişiselleştirilmiş belge oluşturulabilir.

Bu işlem toplu mektup (adres mektup birleştirme) olarak adlandırılır. Toplu mektup işlemi, veritabanındaki bilgilerin önceden hazırlanmış bir belgeye otomatik olarak eklenmesini sağlar. Bunun sonucunda aynı içeriğe sahip, ancak farklı kişisel bilgiler içeren birden fazla belge elde edilir.

Toplu Mektup Nedir?

Toplu mektup, bir veri kaynağına bağlı olarak hazırlanan belgedir. Veri kaynağı genellikle veritabanı, tablo veya alıcılara ilişkin bilgilerin bulunduğu bir listedir. Belgede, daha sonra gerçek verilerle otomatik olarak değiştirilecek özel alanlar bulunur.

Örneğin, belgeye belirli bir kişinin adı doğrudan yazılmak yerine "Ad" alanı eklenebilir. Belgeler oluşturulurken program, bu alanı otomatik olarak her alıcının adıyla değiştirir.

Bu yöntem sayesinde tek bir belge, onlarca, yüzlerce hatta binlerce kişiselleştirilmiş mektup oluşturmak için kullanılabilir. Bu durum çalışma verimliliğini önemli ölçüde artırır ve çok sayıda belgenin hızlı bir şekilde hazırlanmasını sağlar.

Uygulamalı Örnek

Okulun bir bilişim yarışması düzenlediğini ve öğrencilere davetiye göndermesi gerektiğini varsayalım.

Veritabanında aşağıdaki tablo bulunmaktadır:

Ad	Soyad	Sınıf
Ana	Petrova	II-1
Marko	Stoyanov	II-2
Elena	Nikolovska	II-1

Metin işleme programında aşağıdaki şablon hazırlanır:

Sayın <<Ad>> <<Soyad>>,
Sizi okulda düzenlenecek bilişim yarışmasına katılmaya davet ediyoruz.
Saygılarımızla,
Organizasyon Komitesi

Belgenin veritabanındaki verilerle birleştirilmesinden sonra, her öğrenci için otomatik olarak ayrı bir davetiye oluşturulur.

Öğrenciler

Öğrenci ID	Ad	Soyad	Sınıf	Click to Add
1	Ana	Petrova	II-1	
2	Marko	Stoyanov	II-2	
3	Elena	Nikolovska	II-1	
(New)				

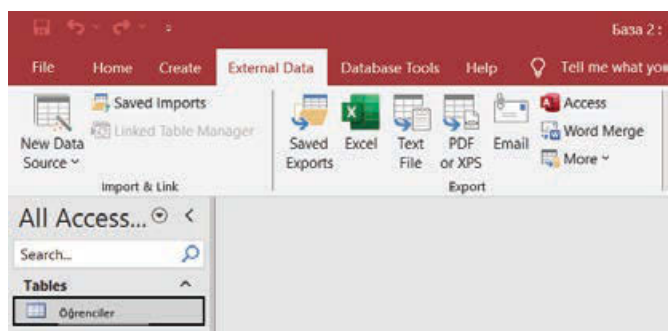
Resim 1: Alıcı bilgilerinin bulunduğu tablo

Belgenin Veritabanına Bağlanması

Toplu mektup oluşturmak için öncelikle belgenin veri kaynağına bağlanması gerekir. Veri kaynağı, veritabanı, tablo veya alıcılara ilişkin bilgileri içeren başka bir veri yapısı olabilir.

Bağlantı kurulduktan sonra program, veritabanında bulunan tüm kayıtlara erişim sağlar. Böylece veriler, belgelerin oluşturulması sırasında otomatik olarak kullanılabilir.

Modern programlarda bu işlem genellikle, kullanıcıyı bağlantı kurma ve gerekli alanları seçme aşamalarında yönlendiren özel bir Sihirbaz (Wizard) yardımıyla gerçekleştirilir. İşleme başlamadan önce tabloların kapalı olması gerekir. Veritabanının bu özel (exclusive) çalışma modu, diğer bileşenlerin ve pasif belgelerin oluşturulmasına olanak sağlar. O anda üzerinde çalışılan bir veritabanından mektup oluşturulmamalıdır. Tablo etkin (açık) durumdaysa Sihirbaz mektubu oluşturamaz. Tablonun yalnızca seçilmiş (tıklanmış) olması, ancak açık olmaması gerekir. Bazı yazılım sürümlerinde ise metin düzenleyici ile birleştirme işleminin gerçekleştirilebilmesi için öncelikle veritabanının tamamının kapatılması gerekebilir.



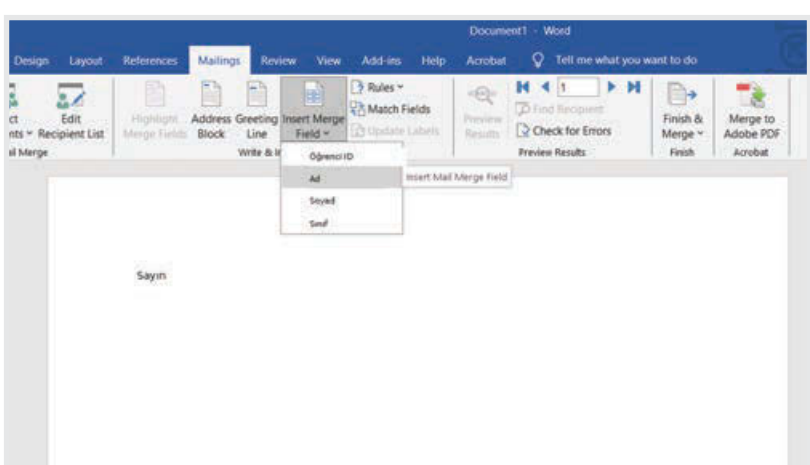
Resim 2: Metin belgesiyle bağlantı kurma

Birleştirme Alanları

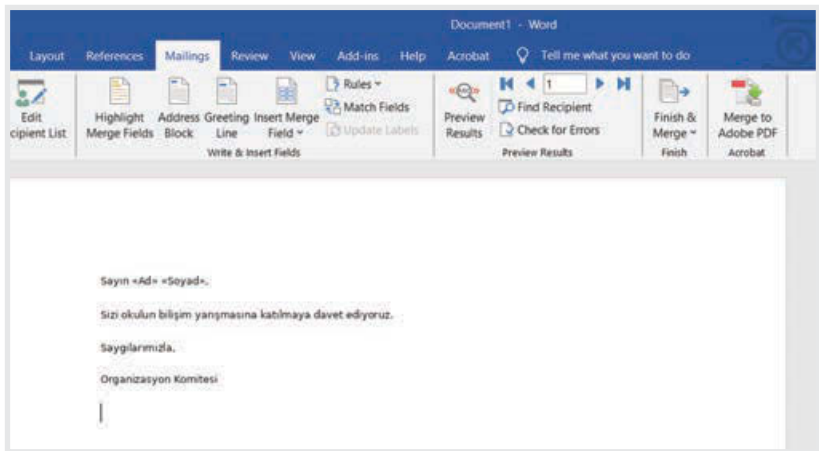
Belge veritabanına bağlandıktan sonra, kayıtlardaki verilerin belgenin hangi bölümlerinde görüntüleneceğinin belirlenmesi gerekir. Bu amaçla birleştirme alanları kullanılır.

Birleştirme alanları, programın veritabanındaki bilgileri yerleştireceği konumu belirten özel alanlardır. Her alan, tablodaki belirli bir sütunla ilişkilendirilir.

Örneğin, <<Ad>> alanı Ad sütunuyla, <<Soyad>> alanı ise Soyad sütunuyla ilişkilendirilir. Belgeler oluşturulurken program, bu alanları her kayda ait uygun bilgilerle otomatik olarak değiştirir.



Resim 3: Birleştirme Alanlarının (Merge Fields) Eklenmesi



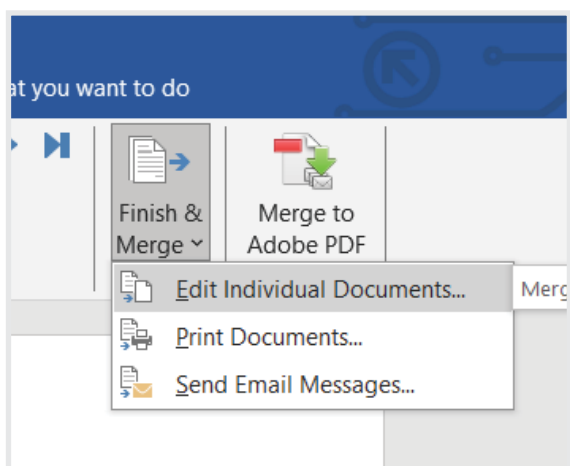
Resim 4: Tamamlanmış mektup

Toplu Mektupların Oluşturulması

Tüm birleştirme alanları yerleştirildikten sonra program, belgeleri otomatik olarak oluşturabilir. Veritabanındaki her kayıt için belgenin ayrı bir sürümü oluşturulur.

Veritabanında on kayıt bulunuyorsa on belge, yüz kayıt bulunuyorsa yüz belge oluşturulur. Kullanıcının adları ve diğer bilgileri elle değiştirmesine gerek kalmaz.

Bu özellik, çalışmayı önemli ölçüde hızlandırır ve birden fazla kişiye ait bilgiler içeren kişiselleştirilmiş davetiyelerin, belgelerin, sertifikaların, duyuruların ve diğer dokümanların hızlı bir şekilde hazırlanmasını sağlar.



Resim 5: Belgelerin oluşturulması

İşlemin tamamlanabilmesi için Şekil 5'te gösterilen simge kullanılarak birleştirme işleminin yapılması ve bu işlemin gerekli olan tüm kayıtlar için uygulanması gerekir.

Uygulamalı Örnek: Yarışma Sertifikaları

Okulun bir programlama yarışması düzenlediğini ve tüm katılımcılar için sertifikalar hazırlanması gerektiğini varsayalım.

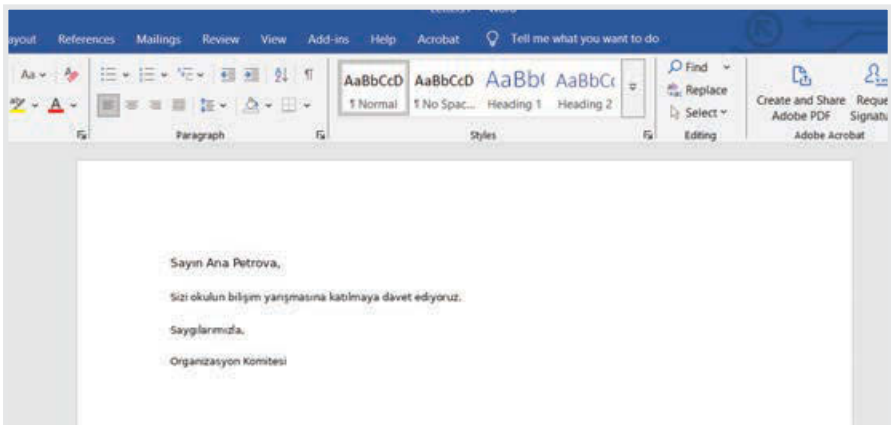
Veritabanında öğrencilerin ad ve soyad bilgileri saklanmaktadır. Her sertifikayı ayrı ayrı düzenlemek yerine, birleştirme alanlarının yerleştirildiği tek bir şablon hazırlanabilir.

Toplu mektup oluşturulurken program, her öğrenci için otomatik olarak bir sertifika oluşturacaktır. Bu sayede birkaç saniye içinde çok sayıda belge profesyonel bir görünümle hazırlanabilir.

Örneğin Analizi

Önceki örnekten de görülebileceği gibi toplu mektuplar, veritabanları ile metin işleme programları arasında bir bağlantı oluşturur. Veriler veritabanında kalırken belge, gerekli bilgilerle otomatik olarak doldurulan bir şablon görevi görür.

Toplu mektupların kullanılması, belgelerin hazırlanması için gereken zamanı azaltır, hata yapma olasılığını düşürür ve tüm belgelerin aynı biçimde hazırlanmasını sağlar. Bu nedenlerle bu teknik, eğitimde, yönetimde ve iş dünyasında geniş bir kullanım alanına sahiptir.



Resim 6: Son kişiselleştirilmiş belge

SONUÇ:

Toplu mektuplar, aynı içeriğe sahip ancak alıcılarına ait farklı bilgiler içeren çok sayıda belgenin otomatik olarak oluşturulmasını sağlayan etkili bir yöntemdir. Her belgenin ayrı ayrı düzenlenmesi yerine, kullanıcı veritabanına veya tabloya bağlanan tek bir şablon hazırlar.

Birleştirme alanlarının kullanılmasıyla program, gerekli bilgileri belgeye otomatik olarak ekler. Böylece zamandan tasarruf edilir, hata yapma olasılığı azalır ve tüm belgelerin aynı biçimde hazırlanması sağlanır.

Toplu mektuplar, çok sayıda kişiselleştirilmiş belgenin hazırlanmasının gerekli olduğu eğitim, yönetim ve iş dünyasında geniş bir kullanım alanına sahiptir.

BİLİYOR MUSUNUZ?

Biliyor muydun? Günümüzde öğrencilerin ve çalışanların aldığı sertifikaların, diplomaların, davetiyelerin ve duyuruların büyük bir bölümü, toplu mektup işlemi kullanılarak otomatik olarak oluşturulmaktadır. Her belgenin elle doldurulması yerine, veriler veritabanından alınarak önceden hazırlanmış bir şablona otomatik olarak eklenir.

TEKRARLAMA SORULARI

1. Toplu mektup nedir?
2. Toplu mektupların avantajları nelerdir?
3. Veri kaynağı nedir?
4. Belge veritabanına nasıl bağlanır?
5. Birleştirme alanları nedir?
6. Birleştirme alanları neden kullanılır?
7. Toplu mektuplar en çok hangi durumlarda kullanılır?

PRATİK ALIŞTIRMALAR

Ödev 1

Aşağıdaki alanlara sahip bir tablo oluşturunuz:

- Ad
- Soyad
- Sınıf

En az beş öğrenciye ait verileri giriniz.

Ödev 2

Bir okul yarışması için davetiye şablonu hazırlayınız.

Şablonda aşağıdaki bilgiler için alanlar oluşturunuz:

- Ad;
- Soyad;
- Sınıf.

Ödev 3

Belgeyi tabloya bağlayınız ve uygun birleştirme alanlarını ekleyiniz.

Ödev 4

Tabloda bulunan tüm öğrenciler için toplu mektup oluşturunuz.

Ödev 5

Veritabanındaki bilgileri otomatik olarak kullanacak bir sertifika şablonu hazırlayınız.

ARAŞTIRMA GÖREVİ

Toplu mektupların en çok hangi alanlarda kullanıldığını araştırınız.

Her alan için şunları belirtiniz:

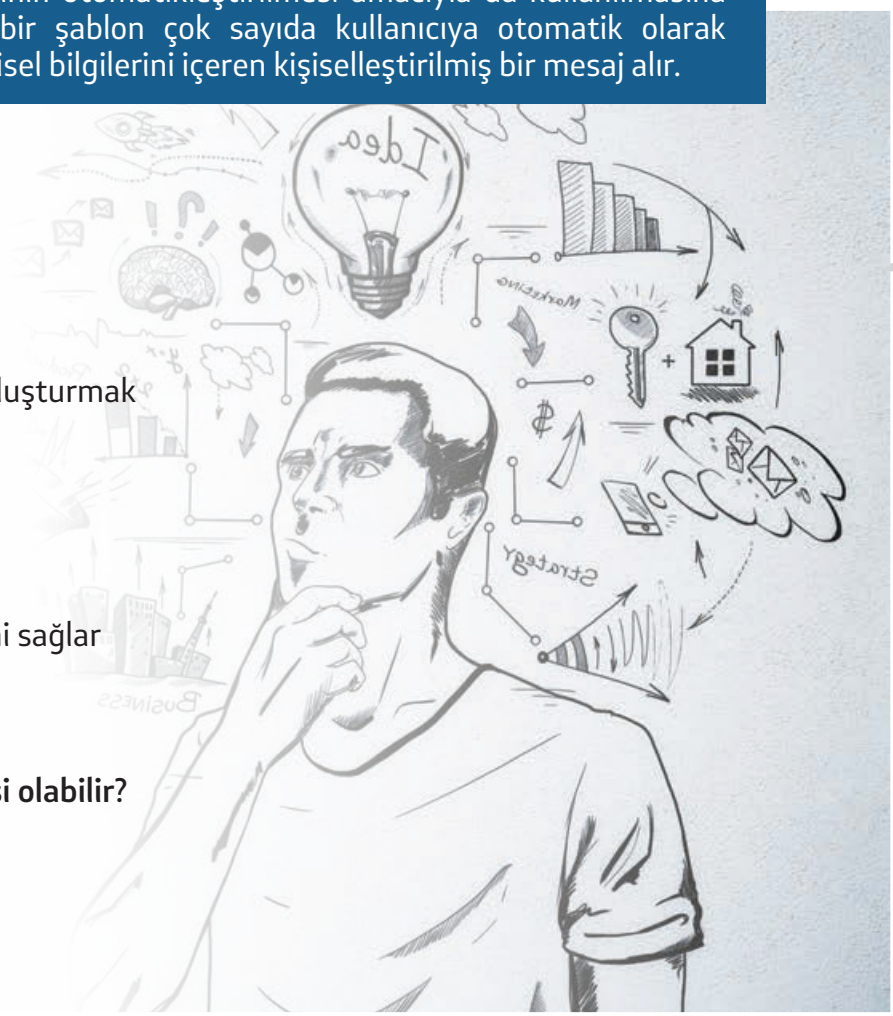
- belge türü;
- veri kaynağı;
- toplu mektupların kullanılmasının sağladığı yararlar.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- Toplu mektup, bir veri kaynağına bağlı olarak oluşturulan belgedir.
- Veri kaynağı, bir veritabanı veya tablo olabilir.
- Birleştirme alanları, verilerin otomatik olarak eklenmesini sağlar.
- Tek bir şablon kullanılarak çok sayıda farklı belge oluşturulabilir.
- Toplu mektuplar zamandan tasarruf sağlar ve hata yapma olasılığını azaltır.
- Bu teknik, davetiyeler, sertifikalar, belgeler ve duyuruların hazırlanmasında kullanılır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern sistemler, toplu mektupların yalnızca belge yazdırmak için değil, aynı zamanda e-posta gönderiminin otomatikleştirilmesi amacıyla da kullanılmasına olanak tanır. Böylece tek bir şablon çok sayıda kullanıcıya otomatik olarak gönderilebilir ve her alıcı, kişisel bilgilerini içeren kişiselleştirilmiş bir mesaj alır.



BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. Toplu mektup ne için kullanılır?

a) Program yüklemek
b) Birden fazla kişiselleştirilmiş belge oluşturmak
c) Diski biçimlendirmek
ç) Veritabanı oluşturmak

2. Birleştirme alanları:

a) Verileri siler
b) Verilerin otomatik olarak eklenmesini sağlar
c) Yeni bir tablo oluşturur
ç) Raporları yazdırır

3. Veri kaynağı aşağıdakilerden hangisi olabilir?

a) Veritabanı
b) Tablo
c) Veritabanı veya tablo
ç) İşletim sistemi

Doğru veya Yanlış

1. Toplu mektup, birden fazla belge için tek bir şablon kullanır.
2. Her belgenin elle doldurulması gerekir.
3. Veriler veritabanından otomatik olarak alınabilir.
4. Birleştirme alanları, belgeler oluşturulurken gerçek verilerle değiştirilir.

ANALİZ

Okulun bir yarışmaya katılan öğrenciler için 300 sertifika hazırlaması gerekmektedir.
Aşağıdaki soruları açıklayınız:

- Toplu mektup neden sertifikaların elle doldurulmasından daha uygun bir çözümdür?
- Veritabanında hangi bilgilere ihtiyaç duyulur?
- Sertifikada hangi birleştirme alanları kullanılır?

3.9

VERİTABANLARIYLA ÇALIŞMAK İÇİN TEMEL DİL

BU KONUYU TAMAMLADIĞINIZDA...

- Veritabanında tablolar oluşturmayı;
- Alanları ve veri türlerini belirlemeyi;
- Birincil anahtar kullanmayı;
- Formlar aracılığıyla veri girmeyi;
- Sorgular kullanarak bilgi aramayı;
- Raporlar hazırlamayı;
- Toplu mektup hazırlarken veritabanındaki verileri kullanmayı öğreneceksiniz.

DÜŞÜNME SORULARI

- Bilgisayar veritabanıyla nasıl iletişim kurar?
- Bir tablodan yalnızca belirli bilgiler nasıl istenebilir?
- Arama sonuçları nasıl sıralanabilir?
- Farklı veritabanlarında kullanılan standart bir dil var mıdır?
- Büyük veritabanlarında aramalar nasıl gerçekleştirilir?

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- SQL'in ne olduğunu açıklamayı;
- Temel SQL komutlarını kullanmayı;
- Basit sorgular gerçekleştirmeyi;
- Arama sırasında koşullar belirlemeyi;
- Sonuçları sıralamayı;
- SQL ile elde edilen sonuçları analiz etmeyi.

Önceki derslerde verilerin formlar aracılığıyla nasıl girilebileceği, sorgularla nasıl aranabileceği ve raporlar aracılığıyla nasıl görüntülenebileceği gösterilmişti. Çoğu programda bu işlemler grafiksel araçlar ve menüler kullanılarak gerçekleştirilebilir. Ancak arka planda, veritabanıyla doğrudan iletişim kurulmasını sağlayan standart bir dil bulunmaktadır.

Bu dil SQL (Structured Query Language – Yapılandırılmış Sorgu Dili) olarak adlandırılır. SQL, ilişkisel veritabanlarıyla çalışmak için kullanılan uluslararası bir standarttır ve çok sayıda veritabanı yönetim sisteminde kullanılmaktadır. SQL sayesinde kullanıcılar ve programcılar tablolar oluşturabilir, veri girebilir, bilgi arayabilir ve veritabanının içeriğini yönetebilirler.

Modern programlar veritabanlarıyla çalışmak için grafiksel araçlar sunsa da bu araçların birçok işlevi aslında SQL komutları oluşturur. Bu nedenle SQL'in temel bilgilerinin bilinmesi, veritabanlarının çalışma mantığının daha iyi anlaşılmasını sağlar.

SQL Nedir?

SQL, ilişkisel veritabanlarıyla çalışmak için kullanılan özel bir dildir. Adı, İngilizce Structured Query Language ifadesinden gelir ve Türkçeye "Yapılandırılmış Sorgu Dili" olarak çevrilir.

SQL ile tablolardaki bilgiler sorgulanabilir, yeni kayıtlar eklenebilir, mevcut veriler değiştirilebilir ve artık ihtiyaç duyulmayan bilgiler silinebilir. Bu çeşitlilik nedeniyle SQL, veritabanları alanındaki en önemli dillerden biridir.

Bu derste yalnızca verilerin aranmasını ve seçilmesini sağlayan temel SQL komutları ele alınacaktır. İşlemler sorgular üzerinde gerçekleştirilir. Aşağıdaki komutların her biri için öncelikle bir tablo oluşturulur, ardından SQL'in kullanılabilmesi için bir sorgu oluşturulur.

SELECT Komutu

En sık kullanılan SQL komutu SELECT komutudur. Bu komut, bir veya birden fazla tablodaki verilerin görüntülenmesini sağlar.

Öğrenciler tablosunu inceleyelim.

Ad	Soyad	Sınıf
Ana	Petrova	II-1
Marko	Stojanov	II-2
Elena	Nikolovska	II-1

Tüm kayıtları görüntülemek için:

```
SELECT * FROM Ogrenciler;
```

Buradaki yıldız işareti (*), tablodaki tüm alanların görüntülenmesi gerektiğini belirtir.

Yalnızca ad ve soyad bilgilerinin görüntülenmesi gerekiyorsa aşağıdaki komut yazılabilir:

```
SELECT Ad, Soyad
FROM Ogrenciler;
```

WHERE Komutu

Birçok durumda yalnızca belirli bir koşulu sağlayan kayıtların görüntülenmesi gerekir. Bu amaçla WHERE komutu kullanılır.

Örneğin, yalnızca II-1 sınıfındaki öğrencilerin görüntülenmesi gerektiğini varsayalım.

```
SELECT *
FROM Ogrenciler
WHERE Sinif = 'II-1';
```

Sistem yalnızca belirtilen koşulu sağlayan kayıtları görüntüler.

Koşullar, verilerin çok daha hızlı aranmasını sağlar ve gerçek hayattaki veritabanlarının neredeyse tamamında kullanılır.

ORDER BY Komutu

Bazen verilerin belirli bir sıraya göre görüntülenmesi gerekir. Bu amaçla ORDER BY komutu kullanılır.

Örneğin, öğrencilerin soyadlarına göre sıralanması gerekiyorsa.

```
SELECT *
FROM Ogrenciler
ORDER BY Soyad;
```

Sonuçlar alfabetik sırayla görüntülenir.

Veriler azalan sırada da sıralanabilir:

```
SELECT *
FROM Ogrenciler
ORDER BY Soyad DESC;
```

DESC anahtar sözcüğü, azalan sıralamayı belirtir.

DELETE Komutu

Bazen belirli kayıtların tablodan kaldırılması gerekir. Bu amaçla DELETE komutu kullanılır.

Örneğin:

```
DELETE FROM Ogrenciler
WHERE Sinif = 'IV-4';
```

Bu komut, belirtilen koşulu sağlayan tüm kayıtları siler.

Verilerin kaybedilmesine neden olabileceğinden DELETE komutu dikkatli kullanılmalı ve her zaman açıkça tanımlanmış bir koşulla birlikte yazılmalıdır.

Uygulamalı Örnek

Öğrenciler hakkında bilgilerin bulunduğu bir veritabanını ele alalım.

Not ortalaması 4,50'den yüksek olan tüm öğrencilerin görüntülenmesi ve sonuçların soyada göre sıralanması gerekmektedir.

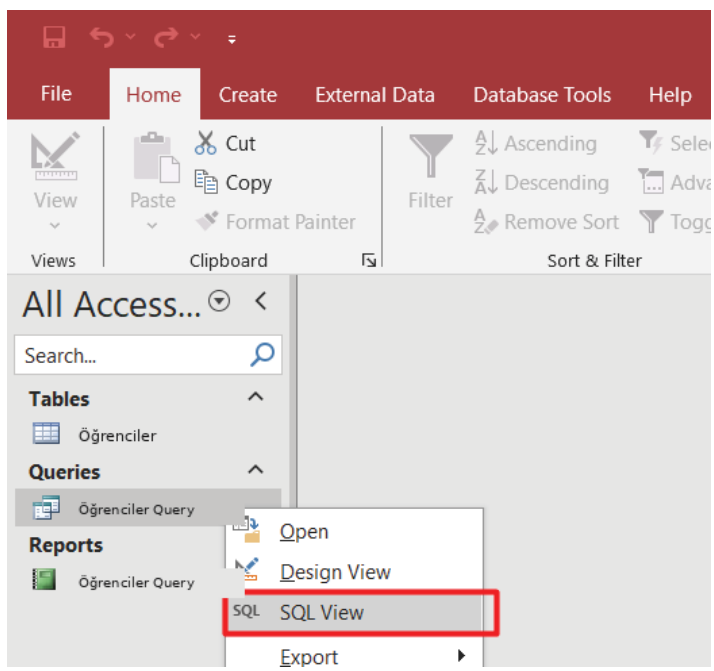
```
SELECT *
FROM Ogrenciler
WHERE Ortalama > 4.50
ORDER BY Soyad;
```

Tek bir komutta arama, koşul belirleme ve sonuçları sıralama işlemleri birleştirilmiştir. Bu, SQL'in en önemli avantajlarından biridir.

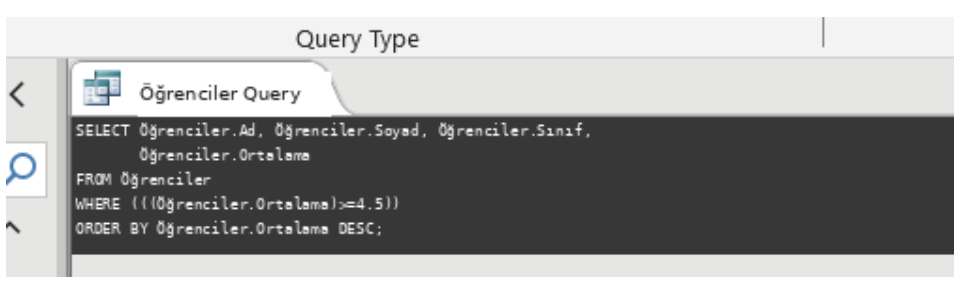
Örneğin Analizi

Önceki örnekten de görülebileceği gibi SQL, verilerin basit ve doğru bir şekilde yönetilmesini sağlar. Kullanıcının kayıtları elle araması yerine, uygun bir SQL komutu yazması yeterlidir. Sistem gerekli bilgileri otomatik olarak bulur.

Basit yapısı ve sunduğu geniş olanaklar sayesinde SQL, günümüzde ilişkisel veritabanlarıyla çalışmak için kullanılan bir standarttır ve hemen hemen tüm modern bilgi sistemlerinde yer almaktadır.



Resim 1: MS Access'te SQL görünümü



Resim 2: SQL komutları

SQL, ilişkisel veritabanlarıyla çalışmak için kullanılan standart bir dildir. SQL sayesinde tablolarda saklanan veriler aranabilir, sıralanabilir ve işlenebilir. Birçok program veritabanlarıyla çalışmak için grafiksel araçlar sunsa da bu araçların temelinde çoğunlukla SQL komutları kullanılır.

SELECT, WHERE ve ORDER BY komutları bilgilerin hızlı bir şekilde bulunmasını ve düzenlenmesini sağlarken, DELETE komutu gereksiz kayıtların kaldırılması için kullanılır. Bu özellikleri sayesinde SQL, modern bilgi sistemlerinde verilerle çalışmak için önemli bir araçtır.

SQL'in öğrenilmesiyle veritabanları konusu tamamlanmış olur. Çünkü öğrenciler yalnızca verilerin nasıl düzenlendiğini değil, aynı zamanda sistemlerin verilerle nasıl iletişim kurduğunu da öğrenmiş olurlar.

BİLİYOR MUSUNUZ?

Biliyor muydun? SQL, kırk yılı aşkın süredir kullanılmaktadır ve günümüzde dünyada en yaygın kullanılan programlama dillerinden biri olarak kabul edilmektedir. Büyük bilgi sistemlerinin, web uygulamalarının ve mobil hizmetlerin neredeyse tamamı, verilerle çalışmak için SQL kullanmaktadır.

TEKRARLAMA SORULARI

1. SQL nedir?
2. SQL kısaltması hangi kelimelerden oluşur?
3. SELECT komutu ne amaçla kullanılır?
4. WHERE komutu ne için kullanılır?
5. ORDER BY komutunun görevi nedir?
6. DELETE komutu ne zaman kullanılır?
7. SQL neden veritabanlarıyla çalışmak için standart bir dil olarak kabul edilir?

PRATİK ALIŞTIRMALAR

Ödev 1

Öğrenci tablosundaki tüm kayıtları görüntüleyecek SQL komutunu yazınız.

Ödev 2

Öğrenci tablosunda yalnızca aşağıdaki alanları görüntüleyecek SQL komutunu yazınız:

- Ad
- Soyad

Öğrenci tablosundan.

Ödev 3

Yalnızca II-1 sınıfındaki öğrencileri görüntüleyecek SQL komutunu yazınız.

Ödev 4

Tüm öğrencileri soyadlarına göre sıralayarak görüntüleyecek SQL komutunu yazınız.

Ödev 5

Not ortalaması 4,50'den yüksek olan öğrencileri görüntüleyecek SQL komutunu yazınız.

Ödev 6

Aşağıdaki SQL komutunun ne yapacağını açıklayınız:

```
SELECT *
FROM Öğrenciler
WHERE Ortalama > 4.50
ORDER BY Soyad;
```

ARAŞTIRMA GÖREVİ

SQL kullanan veritabanı yönetim sistemlerini araştırınız.

En az üç sistem için aşağıdakileri araştırınız:

- Sistem adı;
- Üretici;
- Kullanım alanı;
- Sistem hakkında ilginç bir bilgi.

Kısa bir rapor veya sunum hazırlayınız.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

- SQL, ilişkisel veritabanlarıyla çalışmak için kullanılan standart bir dildir.
- SELECT, verileri görüntülemek için kullanılır.
- WHERE, arama sırasında koşullar belirlemeyi sağlar.
- ORDER BY, sonuçları sıralamak için kullanılır.
- DELETE, kayıtları silmek için kullanılır.
- SQL, büyük miktardaki verilerle hızlı ve verimli bir şekilde çalışmayı sağlar.
- Çok sayıda modern sistem, çalışmalarında SQL kullanmaktadır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Temel komutların yanı sıra SQL, çok sayıda gelişmiş özellik de içerir. SQL ile birden fazla tablo birleştirilebilir, istatistiksel hesaplamalar gerçekleştirilebilir, veriler gruplandırılabilir ve karmaşık raporlar oluşturulabilir. Profesyonel uygulamalarda SQL, bilgi sistemlerinin ve web uygulamalarının geliştirilmesindeki en önemli teknolojilerden biridir.

BİLGİNİZİ TEST EDİN

Doğru cevabı yuvarlayınız

1. SQL nedir?

a) İşletim sistemi
b) Grafik programlama dili
c) Veritabanlarıyla çalışmak için kullanılan dil
ç) Antivirüs programı

2. SELECT komutu ne için kullanılır?

a) Kayıtları silmek
b) Verileri görüntülemek
c) Belgeleri yazdırmak
ç) Formlar oluşturmak

3. WHERE komutu ne için kullanılır?

a) Verileri sıralamak
b) Koşul belirlemek
c) Tabloları silmek
ç) Raporlar oluşturmak

4. ORDER BY ne için kullanılır?

a) Sonuçları sıralamak
b) Verileri silmek
c) Alanlar oluşturmak
ç) Tabloları birleştirmek

Doğru veya Yanlış

1. SQL, veritabanlarıyla çalışmak için kullanılır.
2. SELECT, tablodaki kayıtları siler.
3. WHERE, kayıtların bir koşula göre seçilmesini sağlar.
4. ORDER BY, sonuçların sıralanmasını sağlar.
5. DELETE, kayıtları silmek için kullanılır.

BOŞLUKLARI DOLDURUN

1. _____ komutu verileri görüntülemek için kullanılır.
2. _____ komutu sonuçları sıralamak için kullanılır.
3. _____ komutu kayıtları silmek için kullanılır.

ANALİZ

Aşağıdaki tabloyu inceleyiniz:

Ad	Soyad	Ortalama
Ana	Petrova	5.00
Marko	Stoyanov	4.20
Elena	Nikolovska	4.80

Öğrenci tablosunda ortalaması 4,50'den yüksek olan öğrencileri soyadlarına göre sıralayarak gösterecek SQL komutunu yazınız.

Sonuç olarak neyin görüntüleneceğini açıklayınız.

SON DÜŞÜNCE

Veri tabanları, modern bilgi sistemlerinin temelini oluşturur. Bu tema boyunca verilerin tablolarda nasıl düzenlendiği, formlar aracılığıyla nasıl girildiği, sorgularla nasıl arandığı, raporlar aracılığıyla nasıl görüntülendiği, adres-mektup birleştirmede nasıl kullanıldığı ve SQL ile nasıl işlendiği gösterildi. Bu bilgiler, bilişimin daha ileri düzeyde öğrenilmesi ve modern yazılım çözümlerinin geliştirilmesi için önemli bir temeldir.

KONU 4

KONU 4: MULTİMEDYA VE BİLGİSAYAR GRAFİKLERİ

Multimedya ve bilgisayar grafikleri, modern teknolojinin ve günlük yaşamın önemli bir parçasını oluşturur. Bunlar, metin, ses, görüntü, video ve animasyonun kullanılmasının yanı sıra bilgisayar yardımıyla görsel unsurların oluşturulmasını ve işlenmesini de kapsar. Bilgisayar grafikleri sayesinde gerçekçi görüntüler, oyunlar, filmler ve dijital tasarımlar oluşturabiliriz. Multimedya, bilgilerin özellikle eğitim ve eğlence alanlarında daha ilgi çekici ve anlaşılır olmasını sağlar. Günümüzde bu teknolojiler, pazarlama, tıp, mimarlık ve oyun gibi birçok alanda kullanılmaktadır. Bu nedenle multimedya ve bilgisayar grafiklerini bilmek, dijital dünyada yetişen gençler için büyük önem taşımaktadır.

YENİ KAVRAMLAR:

Multimedya, ögeler – metin, görüntü, ses, video, grafik, raster grafik, vektör grafik, piksel, çözünürlük, seçim, katmanlar, görünürlük.

ARTIK ŞUNLARI BİLECEKSİNİZ:

- Temel araçları kullanarak görüntüleri düzenleyebilirsin;
- Bir görüntünün bölümlerini kopyalayabilir ve taşıyabilir, görüntünün bölümlerini silebilir ve görüntüye efektler uygulayabilirsin.
-

TEKRARLAMA ALIŞTIRMALARI

- Görüntü düzenlemek için kullanılan en az üç aracı sıralayınız.
- "Bir görüntünün bir bölümünü kopyalamak" ne anlama gelir?
- Bir görüntüye uygulanabilecek en yaygın efektler nelerdir?
- Bir görüntünün bir bölümünü "kopyalamak" ile "taşımak" arasındaki farkı açıklayınız.
- Efektlerin görüntünün görsel görünümünü nasıl değiştirdiğini açıklayınız.
- Bir görüntüdeki nesneyi kopyalayarak başka bir görüntüye yerleştiriniz.
- Verilen bir görüntüye "siyah-beyaz" efekti uygulayınız.
- Orijinal görüntü ile düzenlenmiş görüntüyü karşılaştırınız. Aralarındaki farklar nelerdir?
- Nesnelerin taşınmasının görüntünün kompozisyonunu nasıl değiştirdiğini analiz ediniz.
- Belirli ögelerin silinmesinin görüntüyü neden iyileştirebileceğini açıklayınız.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Multimedyayı, temel ögelerini ve kullanım alanlarını, ayrıca raster ve vektör grafiklerin özelliklerini açıklamayı;
- Görüntü işleme için temel araç ve teknikleri, düzeltmeler, katmanlar ve grafik ögelerinin birleştirilmesi de dâhil olmak üzere kullanmayı;
- Görüntü oluşturmak ve dönüştürmek için yapay zekâ araçlarını kullanmayı, basit komutlar (prompt) oluşturmayı ve elde edilen sonuçları analiz etmeyi;
- Öğrenilen teknikleri ve yapay zekâ araçlarını kullanarak basit grafik tasarımlar (poster, afiş veya logo) oluşturmayı.

4.1

MULTİMEDYA

Modern dijital dünyada bilgiler nadiren yalnızca tek bir yöntemle aktarılır. Elektronik bir kitap okuduğumuzda, eğitici bir video izlediğimizde, bir sunumu takip ettiğimizde, bir uygulama kullandığımızda veya internette içerik aradığımızda genellikle aynı anda metin, görüntü, ses, video ve etkileşimli ögelerle karşılaşırız. Bu tür bir içerik sunumu insanlara daha yakındır, çünkü günlük yaşamda olayları birbirinden bağımsız olarak değil, aynı anda birden fazla duyu ve deneyim aracılığıyla algılarız. Bilgilerin bu şekilde aktarılması, multimedya kavramının anlaşılmasının temelini oluşturur.

Multimedya kavramı, metin, görüntü, ses, video ve animasyon gibi birden fazla ögenin bir araya getirilerek bilgilerin sunulması şeklinde açıklanabilir. Bu ögeler, belirli bir içeriği daha kapsamlı bir şekilde açıklamak, göstermek veya aktarmak amacıyla birlikte kullanılır.

Multimedya kelimesi iki bölümden oluşur: "multi", "çok" anlamına gelir, "media" ise bilgilerin aktarıldığı ortamları veya araçları ifade eder.

Bilişim alanında multimedya, metin, görüntü, ses, video, animasyon ve etkileşimli bölümler gibi farklı ögelerin bir arada kullanıldığı dijital içerik ve uygulamalarla ilişkilidir. Bu ögeler bir bütün hâlinde bir araya getirildiğinde multimedya içeriği veya multimedya uygulaması oluşturulur. Multimedya içeriklerine sunum, web sitesi ve elektronik ders kitabı örnek verilebilir. Multimedya uygulamalarına ise eğitim uygulamaları ve bilgisayar oyunları örnek gösterilebilir.

Bir multimedya içeriğini oluşturan bölümlere multimedya ögeleri denir. Temel multimedya ögeleri şunlardır: metin, görüntü, ses, video, animasyon ve etkileşimli ögeler. Her birinin farklı bir görevi vardır: Metin, bilgi, açıklama veya yönerge verir. Görüntü, bir kavramı, nesneyi veya olayı görsel olarak gösterir. Ses, konuşma, müzik veya sesli sinyalleri aktarır. Video, hareketi, bir işlemi veya olayı gösterir. Etkileşimli ögeler ise kullanıcının seçim yapmasına, cevap vermesine veya içeriğin bazı bölümlerini kontrol etmesine olanak sağlar.

Multimedyanın yalnızca birden fazla ögenin aynı yerde bulunması anlamına gelmediğini bilmek önemlidir. Bu ögelerin konuyla ilişkili olması ve ortak bir amaca hizmet etmesi gerekir. Örneğin, bir ders konusu hakkında metin, görüntüler ve kısa bir videodan oluşan bir sunum multimedya içeriğidir. Ancak metin, görüntü ve ses düzensiz bir şekilde yerleştirilmiş ve konuyla ilişkilendirilmemişse, bunlar bilgilerin anlaşılmasına yardımcı olmayacaktır.

Multimedya, bilgilerin anlaşılır, ilgi çekici ve kullanışlı bir şekilde sunulmasını sağladığı için günlük yaşamda geniş bir kullanım alanına sahiptir. Eğitim, eğlence, iletişim ve iş dünyası gibi birçok farklı alanda kullanılmaktadır:

- Eğitimde multimedya, ders içeriklerinin daha kolay anlaşılabilecek bir şekilde açıklanmasına ve gösterilmesine yardımcı olur. Örneğin, bir öğrenci Güneş Sistemi hakkında

yalnızca okumak yerine gezegenlerin hareketlerini gösteren bir animasyon izleyebilir, açıklamaları dinleyebilir ve etkileşimli soruları yanıtlayabilir. Böylece öğrenme daha kolay ve daha aktif hâle gelir.

- Tıpta multimedya, çeşitli sağlık durumlarının, organların işleyişinin veya belirli tıbbi işlemlerin açıklanmasına yardımcı olabilir. Örneğin, doktor, hastanın vücudunda neler olduğunu veya belirli bir işlemin nasıl gerçekleştirileceğini daha iyi anlayabilmesi için görüntüler, 3B görseller, videolar ve kısa açıklamalar kullanabilir.

- Turizmde multimedya, insanların bir yeri ziyaret etmeden önce o yer hakkında bilgi edinmelerine yardımcı olur. Örneğin, bir turizm web sitesi veya uygulaması, fotoğraflar, haritalar, videolar, sanal geziler ve turistik yerler, oteller veya müzeler hakkında kısa bilgiler içerebilir.

- Mühendislikte multimedya, karmaşık projelerin ve teknik sistemlerin daha kolay anlaşılabilecek şekilde gösterilmesi amacıyla kullanılır. Örneğin, bir makinenin çalışması, hareketlerin animasyonunu, metinsel açıklamaları, sesli yönergeleri ve kullanıcının makinenin incelemek istediği bölümünü seçebilmesini sağlayan bir dijital simülasyon aracılığıyla gösterilebilir.

- Sporda multimedya, spor etkinliklerinin analizinde kullanılabilir. Örneğin, bir spor karşılaşmasının analizi, video kayıtları, ağır çekim tekrarlar, grafikler, istatistiksel veriler ve oyuncuların hareketleriyle ilgili açıklamalar içerebilir.

Multimedyanın kullanımına bilgisayar oyunları da örnek olarak verilebilir. Bir bilgisayar oyunu, görüntü, ses, hareket, metinsel yönergeler ve oyuncunun oyunun akışını kontrol etmesini sağlayan etkileşimli ögeler içerebilir.

Multimedya, günümüzde öğrenme, çalışma, bilgi edinme ve pratik sorunları çözme biçiminin bir parçasıdır. Yalnızca bilgisayarlar ve mobil cihazlarla ilişkili değildir, aynı zamanda günümüzde farklı bilgilerin nasıl oluşturulduğu, sunulduğu ve kullanıldığıyla da yakından ilişkilidir.

SONUÇ:

Multimedya, farklı içerik türlerinin bir araya getirilmesini sağlayarak modern dijital dünyanın önemli bir parçasını oluşturur. Bilgilerin daha açık, ilgi çekici ve anlaşılır hâle gelmesini sağlar. Multimedyanın ögelerinin tanınması ve gerçek yaşam durumlarında uygulanması sayesinde yaratıcılık ve dijital beceriler geliştirilir. Bu bilgiler, modern toplumda daha etkili iletişim kurmaya ve öğrenmeye yardımcı olur.

BİLGİLERİ TEKRARLAMA SORULARI

1. Multimedya nedir?
2. Multimedyanın en az dört ögesini sıralayınız.
3. Multimedyanın günlük yaşamda neden önemli olduğunu açıklayınız.
4. Görüntü ve sesin birlikte kullanılmasının sunumu nasıl geliştirdiğini açıklayınız.
5. Multimedyanın eğitimde kullanıldığı iki somut örnek veriniz.
6. Bir YouTube videosunda hangi multimedya ögelerinin kullanıldığını belirleyiniz.
7. Bir çevrim içi oyunda hangi multimedya ögelerinin bulunduğunu analiz ediniz.
8. Multimedyanın iki farklı örneğini (örneğin film ve web sitesi) kullanılan ögeler açısından karşılaştırınız.
9. Multimedyanın en önemli ögesinin hangisi olduğunu değerlendiriniz ve nedenini açıklayınız.
10. Günlük yaşamdan, multimedyanın bir bilgiyi daha iyi anlamanıza yardımcı olduğu bir örnek veriniz. Hangi ögelerin kullanıldığını açıklayınız ve bu alandaki belirli bir sorunun çözümüne nasıl katkı sağladığını belirtiniz.
11. Tıp, turizm, spor veya mühendislik gibi bir alan seçiniz ve multimedyanın ilgili alanda nasıl kullanılabileceğini öneriniz.

4.2

VEKTÖR VE RASTER GRAFİK

Günlük yaşamda fotoğraflar, çizimler, simgeler, haritalar, şemalar, logolar ve diğer görsel gösterimlerle sık sık karşılaşırız. Bunlar bir mesaj iletebilir, bir nesneyi temsil edebilir, bir bilgiyi gösterebilir veya belirli bir yönergenin daha kolay anlaşılmasına yardımcı olabilir.

Bilgilerin, fikirlerin veya nesnelerin görüntüler, çizgiler, şekiller, renkler ve semboller gibi görsel ögeler aracılığıyla temsil edilmesine grafik denir.

Grafiklerin bilgisayar yardımıyla oluşturulması, işlenmesi ve görüntülenmesi ise bilgisayar grafikleri olarak adlandırılır.

Görüntünün hangi temel ögelerden oluştuğuna bağlı olarak bilgisayar grafiklerini genellikle iki temel türe ayırırız: raster grafik ve vektör grafik.

Vektör grafikte görüntü, çizgiler, eğriler, şekiller ve diğer grafik nesneleriyle oluşturulur. Bu ögeler bir piksel ağı şeklinde kaydedilmez, bunun yerine matematiksel kurallarla tanımlanır. Bilgisayar, görüntüyü ekranda göstermek için bu kurallardan yararlanır.

Örneğin, vektör grafikte bir çember çok sayıda küçük nokta şeklinde değil, belirli bir konumu, boyutu, çizgisi ve rengi olan bir şekil olarak tanımlanır. Bu nedenle vektör görüntü, bulanıklaşmadan veya küçük karelerden oluşmuş gibi görünmeden kolayca büyütülebilir veya küçültülebilir. Görüntü büyütülse bile çizgiler ve şekiller netliğini ve keskinliğini korur.

Vektör grafikler çoğunlukla logolar, simgeler, işaretler, semboller, teknik çizimler, illüstrasyonlar ve farklı boyutlarda kullanılması gereken tasarımlar için tercih edilir. Örneğin, bir logo kartvizit üzerinde küçük, poster veya reklam panosu üzerinde ise çok büyük kullanılabilir ve görüntü bulanıklaşmaz.

Vektör grafiklerin avantajı, çizgilerin ve şekillerin netliği kaybolmadan büyütülüp küçültülebilmesidir. Ayrıca görüntüdeki çizgiler, şekiller ve renkler gibi ayrı ögeler kolayca değiştirilebilir. Dezavantajı ise çok sayıda renk tonu, gölge ve ince ayrıntı içeren fotoğrafların ve gerçekçi görüntülerin gösterilmesi için en uygun grafik türü olmamasıdır.

Vektör grafiklerle çalışmak için Inkscape, Adobe Illustrator, CorelDRAW, Figma ve LibreOffice Draw gibi programlar kullanılır.

Raster grafikte görüntü, piksel adı verilen küçük renkli noktaların oluşturduğu bir mozaik şeklinde temsil edilir. Piksel, bir raster görüntünün oluştuğu en küçük ögedir. Pikseller çok küçük ve birbirine yakın olduğunda insan gözü bunları tek tek fark etmez, bunun yerine pikselleri bir bütün olarak algılayarak görüntüyü oluşturur.

Raster görüntülerde önemli bir kavram çözünürlüktür. Günlük kullanımda çözünürlük çoğunlukla görüntünün piksel cinsinden boyutlarıyla ilişkilendirilir, örneğin 1920 × 1080 piksel. Bu, görüntünün genişliğinde 1920 piksel, yüksekliğinde ise 1080 piksel bulunduğu anlamına gelir. Baskı ve tarama işlemlerinde çözünürlük genellikle inç başına nokta sayısı (dpi – dots per inch) olarak ifade edilir, örneğin 300 dpi. Piksel sayısının fazla olması görüntünün daha fazla ayrıntı içermesini sağlarken, baskıda daha yüksek nokta yoğunluğu daha net bir çıktı elde edilmesi açısından önemlidir.

Raster grafiklerin avantajı, çok sayıda renk, gölge ve renkler arasında yumuşak geçişler içeren karmaşık görüntüleri başarılı bir şekilde gösterebilmesidir. Bu nedenle raster grafikler fotoğraflar ve gerçekçi görüntüler için oldukça uygundur.

Raster grafiklerin dezavantajı, görüntü çok fazla büyütüldüğünde kalitesini kaybedebilmesidir. Görüntü piksellerden oluştuğu için büyütme sırasında pikseller görünür hâle gelebilir ve görüntü bulanık veya küçük karelerden oluşuyormuş gibi görünebilir. Ayrıca yüksek çözünürlüklü raster görüntüler daha fazla bellek alanı kaplayabilir.

Raster grafiklerle çalışmak için Microsoft Paint, Adobe Photoshop, GIMP, Krita ve Paint.NET gibi çizim ve görüntü işleme programları kullanılır.

Grafik dosyaları farklı formatlarda kaydedilebilir. Seçilen formata bağlı olarak dosyanın boyutu, görüntünün kalitesi, saydam arka plan desteği ve görüntünün kullanım veya düzenleme biçimi değişebilir.

Temel görüntü formatları:

- JPG veya JPEG, genellikle dijital fotoğraflar ve çok sayıda renk ve ayrıntı içeren görüntüler için kullanılan bir raster formatıdır. İnternete yüklenen veya e-posta yoluyla gönderilen fotoğraflar için uygundur, ancak saydam arka planı desteklemez.

- PNG, sıkıştırma sırasında görüntü kalitesini kaybetmeyen bir raster formatıdır. Saydam arka planı desteklediği için simgeler, web sitesi görüntüleri, ekran görüntüleri, grafik ögeleri ve kenarların ve ayrıntıların net kalmasının önemli olduğu görüntüler için sıklıkla kullanılır.

- SVG, çoğunlukla vektör grafikler için kullanılan bir formattır. Farklı boyutlarda görüntülendiğinde bulanıklaşmaması veya küçük karelerden oluşmuş gibi görünmemesi gereken logolar, simgeler, semboller ve basit illüstrasyonlar için uygundur.

Format seçerken görüntünün kullanım amacı dikkate alınmalıdır. Fotoğraflar için genellikle JPG, saydam arka plana ve net grafik ögelere sahip görüntüler için PNG, logolar, simgeler ve vektörel illüstrasyonlar için ise SVG kullanılır.

Diğer grafik dosyası formatları:

- Raster grafik için: GIF, BMP, TIFF, RAW, PCX, TGA.

- Vektör grafik için: AI, EPS, CDR, DXF, WMF, EMF.

Raster ve vektör grafik arasındaki farkın anlaşılması, görüntülerin doğru şekilde kullanılmasına, düzenlenmesine ve saklanmasına yardımcı olur. Bir görüntünün nasıl oluşturulduğu ve hangi formatta kaydedildiği bilindiğinde, görüntünün işlenmesi, yazdırılması veya paylaşılması için uygun yöntemi seçmek daha kolay olur.

SONUÇ:

Anlaşılması gereken temel nokta, raster ve vektör grafik arasındaki farktır; bu fark, dijital görüntülerin doğru şekilde kullanılabilmesi açısından büyük önem taşır. Piksel ve çözünürlük kavramlarının bilinmesi, bir görüntünün kalitesinin değerlendirilmesine yardımcı olur. Her iki grafik türünün avantaj ve dezavantajlarının bilinmesi, ihtiyaca göre en uygun yaklaşımın seçilmesini sağlar. JPG, PNG ve SVG formatlarının tanınması ise grafik içeriklerinin daha verimli kullanılmasına ve saklanmasına katkıda bulunur.

BİLGİLERİ TEKRARLAMA SORULARI

1. Raster ve vektör grafikleri, görüntünün oluşturulma şekline göre karşılaştırınız.
2. Dijital fotoğrafın neden çoğunlukla raster görüntü olarak, logonun ise vektör grafik olarak kaydedildiğini açıklayınız.
3. Düşük çözünürlüklü bir raster görüntü büyük bir poster için yazdırılmak üzere büyütülürse ne olacağını analiz ediniz.
4. JPG ve PNG formatlarını karşılaştırınız. Hangi durumda JPG, hangi durumda PNG seçerdiniz? Nedenini açıklayınız.
5. Raster ve vektör görüntüleri büyütme (zoom) sırasında gösterdikleri davranış açısından karşılaştırınız.
6. Piksellerin bir fotoğrafın kalitesini nasıl etkilediğini analiz ediniz.
7. JPG ve PNG arasındaki farkı, kalite ve dosya boyutu açısından açıklayınız.
8. Aşağıdaki durumlar için hangi formatın en uygun olduğunu belirleyiniz: web sitesi için fotoğraf, okul logosu ve saydam arka plana sahip simge. Seçiminizi gerekçelendiriniz.
9. Bir okul etkinliği için poster hazırlamanız gerektiğini düşününüz. Hangi ögeleri raster grafik, hangilerini vektör grafik olarak hazırlardınız? Seçiminizi açıklayınız.

4.3

GÖRÜNTÜ İŞLEMİNİN TEMELLERİ

Görüntü işleme, dijital bir görüntünün görünümünü, kalitesini veya kullanım amacına uygunluğunu iyileştirmek amacıyla değiştirilmesi ve düzenlenmesi sürecidir. Bu süreç, gereksiz bölümlerin kaldırılması, görüntünün boyutunun değiştirilmesi, parlaklık, kontrast ve renklerin ayarlanması, metin veya diğer unsurların eklenmesi ve çeşitli efektlerin uygulanması gibi farklı işlemleri kapsar. Bu işlemler sayesinde görüntü, yazdırma, sunum, web sitesi veya başka bir dijital içerik için uygun hâle getirilebilir. Bu nedenle, görüntü işlemenin temel tekniklerini bilmek, görsel bilgilerin doğru ve etkili bir şekilde kullanılabilmesi açısından önemlidir.

Raster grafiklerle çalışmak için kullanılan programlardan biri GIMP (GNU Image Manipulation Program)'dir. GIMP, görüntü işleme amacıyla kullanılan ve GNU Genel Kamu Lisansı (GNU GPL) kapsamında sunulan özgür bir yazılımdır. Bu lisans, kullanıcıların programı özgürce kullanmasına, kaynak kodunu inceleyip değiştirmesine ve değiştirilmiş sürümlerini aynı lisans koşulları altında paylaşmasına olanak tanır. GIMP, kullanıcının görüntüleri farklı biçimlerde açabildiği, düzenleyebildiği ve kaydedebildiği bir çalışma ortamı sunar. Programda görüntülerle çalışmak için çok sayıda araç bulunur. Bu araçlar menüler, paneller ve paletler hâlinde düzenlenmiştir. Böylece işlemlerin aşamalı ve düzenli bir şekilde gerçekleştirilmesi sağlanır. GIMP'in özellikle önemli özelliklerinden biri katmanlarla çalışma olanağıdır. Katmanlar sayesinde görüntünün farklı bölümleri birbirinden bağımsız olarak düzenlenebilir.

4.3.1. GIMP ÇALIŞMA ORTAMI

Tek Pencere Modu'nda (Single-Window Mode), GIMP çalışma ortamı genellikle üç ana bölümden oluşur: araçların bulunduğu sol panel, görüntünün gösterildiği ve düzenlendiği orta çalışma alanı ve katmanlar, kanallar, yollar ve diğer ayarlar gibi ek iletişim kutularının bulunduğu sağ panel. Panellerin genişliği, aralarındaki sınır çizgisi sürüklenerek ayarlanabilir.

Resim 4.1: GIMP çalışma ortamı

Araçlar paneli şunları içerir:

1. Görüntü işleme araçları (Tools menüsünden de seçilebilir);

2. Ön plan (ana, birincil) ve arka plan (ikincil) renklerini seçme simgesi, beyaz ve siyah dikdörtgen şeklindedir;

3. Seçilen aracın özellikleri (Tool Options yardımcı aracı).

Çalışma ortamının sağ tarafındaki yardımcı panel, üst (4 numarayla gösterilen) ve alt (5 numarayla gösterilen) yardımcı pencere grubu olmak üzere ikiye ayrılır. Bu pencereler, örneğin katmanlar, kanallar ve fırçalarla çalışma gibi görüntü işlemeye yönelik ek olanaklar sunar. Kullanılmayan yardımcı pencereler, sağ üst köşedeki küçük oka tıklandığında açılan menüden Close Tab komutuyla kapatılabilir; yeniden açmak için ise Windows -> Dockable Dialogs seçilir.

Çalışma ortamının orta bölümünü görüntü paneli kaplar. Bu panelin üst kısmında, yüklenmiş görüntülerin küçük önizlemelerinin bulunduğu sekmeler yer alır.

Görüntünün üst kısmında yatay bir cetvel, sol tarafında ise dikey bir cetvel bulunur. Görüntünün altında ve sağ tarafında yatay ve dikey kaydırma çubukları bulunur.

Görüntü panelinin alt kısmındaki durum çubuğu dört bölüme ayrılmıştır: fare işaretçisinin koordinatları (yalnızca fare işaretçisi görüntü panelindeyken gösterilir), ölçü biriminin seçildiği açılır liste, yüzde cinsinden yakınlaştırma düzeyinin seçildiği alan ve görüntünün yüklendiği dosyanın adını ve boyutunu gösteren bölüm.

Çalışma ortamının ayarlanması, Edit menüsünde bulunan Preferences penceresi üzerinden gerçekleştirilir.

4.3.2. GÖRÜNTÜ AÇMA, YENİ GÖRÜNTÜ OLUŞTURMA, KAYDETME VE DIŞA AKTARMA

Görüntü açma (Open), yeni görüntü oluşturma (New) ve kaydetme (Save, Save As) komutları File menüsünde bulunur. GIMP'te görüntüler XCF formatında kaydedilir. Görüntüler başka bir formatta Export As komutu kullanılarak kaydedilebilir.

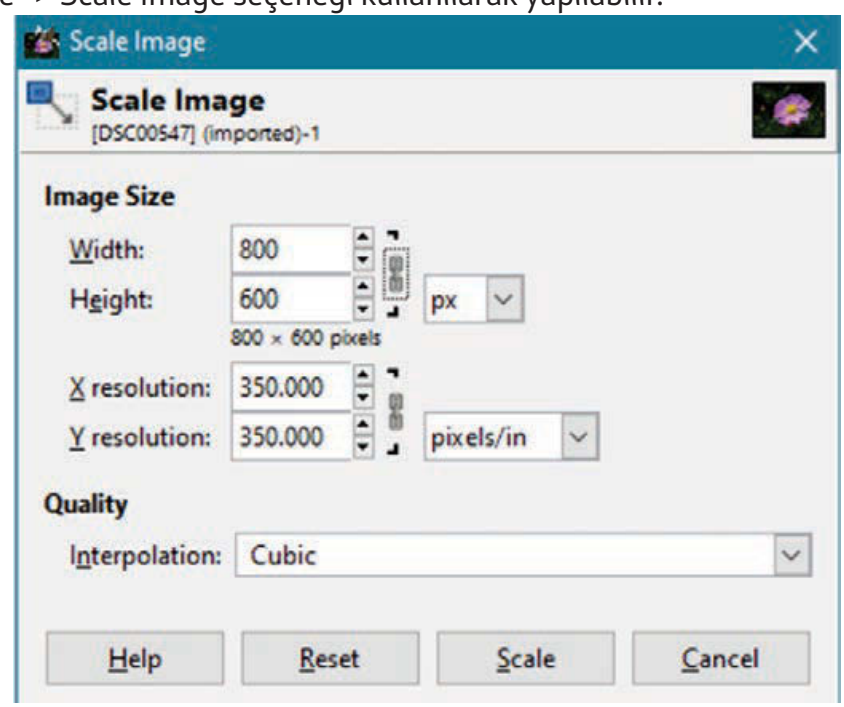
4.3.3. GÖRÜNTÜNÜN RENK MODELİNİ DEĞİŞTİRME

Bu işlem, Image menüsündeki Mode seçeneği kullanılarak yapılabilir. GIMP üç renk modeli sunar:

- RGB – 24 bit renk derinliğine sahip standart renk modelidir.
- Grayscale – Görüntünün her pikseli bir gri tonuyla gösterilir.
- Indexed – Her pikselin rengi, en fazla 256 renk içerebilen renk paletindeki rengin indeksine (sıra numarasına) göre belirlenir.

4.3.4. GÖRÜNTÜNÜN BOYUTUNU DEĞİŞTİRME

Bu işlem Image -> Scale Image seçeneği kullanılarak yapılabilir.



Resim 4.2: Image -> Scale Image

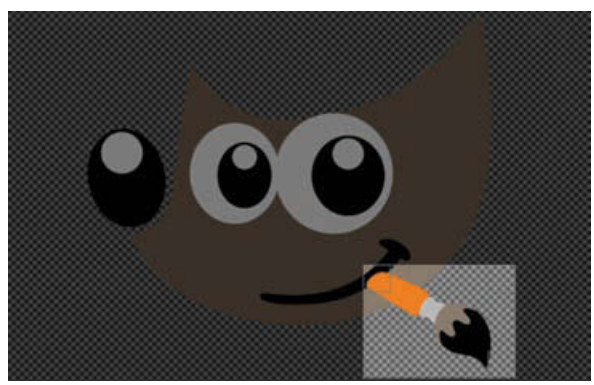
Yeni genişlik (Width) ve yükseklik (Height) birkaç farklı ölçü birimiyle belirtilebilir. Örneğin piksel (px, pixels) veya yüzde (%, percent) olarak girilebilir.

4.3.5. TUVALİN BOYUTUNU DEĞİŞTİRME

Tuval (Canvas), görüntünün görünen alanını belirler. Varsayılan olarak tuvalin boyutu görüntünün boyutuyla aynıdır. Eğer tuval görüntüden daha büyükse, görüntünün çevresinde ek bir alan oluşur. Öte yandan, tuval görüntüden daha küçükse, tuvalin dışında kalan görüntü bölümleri görünmez olur, ancak bu bölümler kaybolmaz. Tuval yeniden büyütüldüğünde bu bölümler tekrar görünür. Tuval boyutunu değiştirme penceresi Image -> Canvas Size seçilerek açılır.

4.3.6. GÖRÜNTÜNÜN BİR BÖLÜMÜNÜ KIRPMA

Bu işlem Crop ( ) aracı kullanılarak yapılır. Kırpılacak alan belirlenir ve onaylamak için Enter tuşuna basılır. Sonuç olarak yalnızca seçilen kırpma alanının iç kısmını içeren yeni bir görüntü elde edilir. Bu alanın dışında kalan her şey kaldırılır. Kırpma işlemi Esc tuşuna basılarak iptal edilebilir.

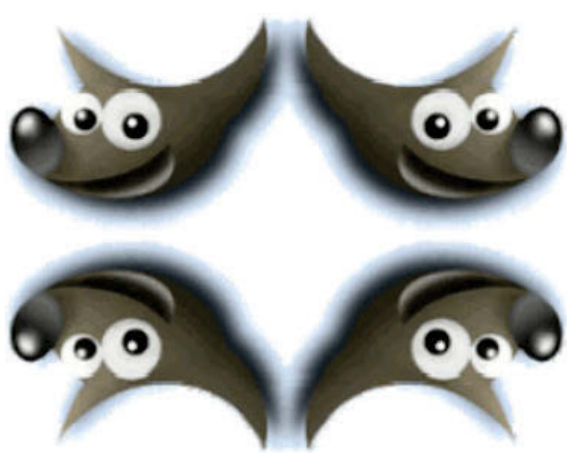


Resim 4.3: Kırpma

4.3.7. GÖRÜNTÜYÜ ÇEVİRME

Görüntünün yatay veya dikey olarak çevrilmesi, araçlar grubundaki

araçlar grubundaki Flip ( ) aracının seçilmesiyle veya Image –> Transform –> Flip Horizontally / Flip Vertically menü seçenekleri kullanılarak yapılır.

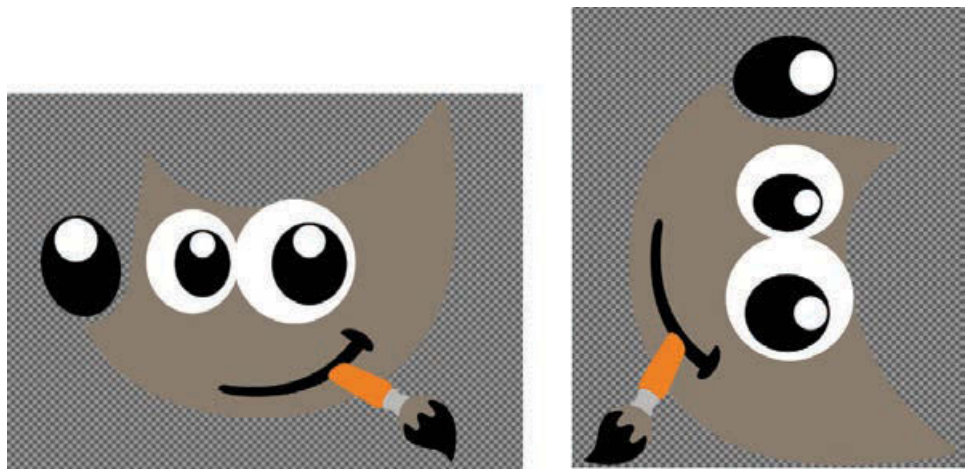


Resim 4.4: Görüntüyü çevirme

4.3.8. DÖNDÜRME

Bir görüntüyü döndürmek için araçlar grubundaki Rotate aracı

() seçilebilir veya Image –> Transform –> Rotate 90° clockwise (saat yönünde 90 derece döndürme) / Rotate 90° counter-clockwise (saat yönünün tersine 90 derece döndürme) ve Rotate 180° (180 derece döndürme) seçenekleri kullanılabilir.

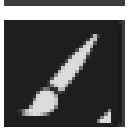


Resim 4.5: Döndürme

4.3.9. EĞRİ VE DÜZ ÇİZGİLER ÇİZME

Eğri ve düz çizgiler çizmek için kullanılan aşağıdaki üç araç aynı şekilde kullanılır:

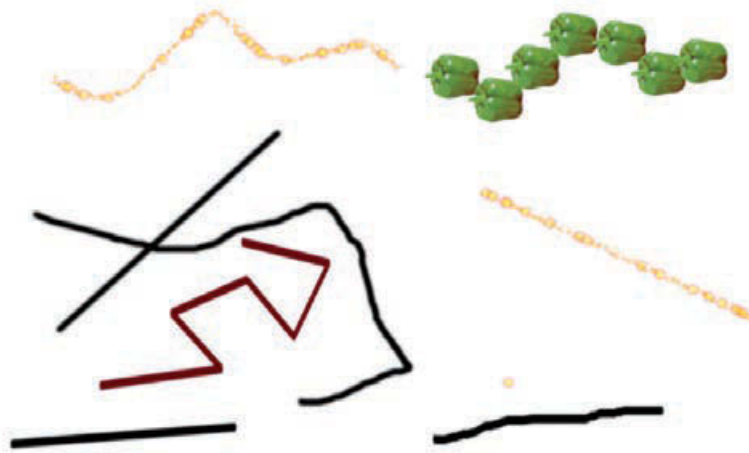
- kurşun kalem ile kaba çizgiler çizme,



- fırça ile yumuşak çizgiler çizme,

- sprey.

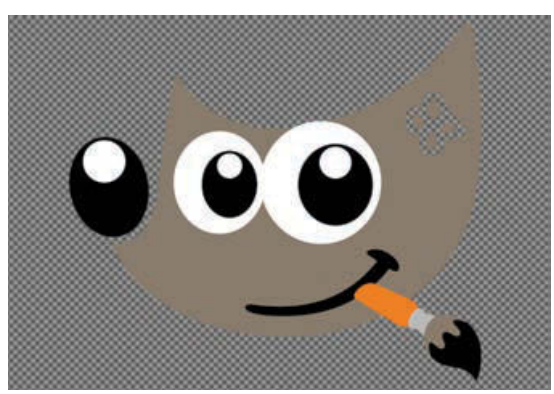
Öncelikle, aracın özellikleri penceresinde renk (Foreground Color), şekil (Brush) ve fırçanın boyutu (Size) gibi seçenekler ayarlanır. Daha sonra tuval üzerinde farenin sol tuşuna basılır ve istenilen şekil çizilene kadar tuş bırakılmadan hareket ettirilir. Düz çizgiler, çizginin başlangıç ve bitiş noktalarının belirlenmesiyle çizilir. Bunun için tuval üzerinde tıklama yapılırken Shift tuşu basılı tutulur (ilk nokta Shift tuşuna basılmadan belirlenir). Bu şekilde her tıklamada program, yeni noktayı önceki noktaya düz bir çizgiyle bağlar.



Resim 4.6: Çizgi çizme

4.3.10. GÖRÜNTÜNÜN BÖLÜMLERİNİ SİLME

Silme aracı () çizim araçlarıyla aynı şekilde kullanılır ve şekil, fırça boyutu ve opaklık gibi benzer özelliklere sahiptir. Yukarıda belirtilen ve ön plan rengini (Foreground Color) kullanan çizim araçlarının aksine, silgi arka plan rengini kullanır. Görüntünün katmanının saydam bir arka planı varsa, silgiyle silinen görüntü alanı saydam olur. Görüntünün katmanının saydam bir arka planı yoksa, silgiyle silinen alan arka planın rengiyle doldurulur.

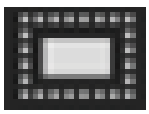


Resim 4.7: Saydam arka planlı görüntüyü silme

4.3.11. SEÇİM ARAÇLARI

Seçim komutları Select menüsü içerisinde gruplandırılmıştır. Seçimle çalışmak için temel komutlar şunlardır: All – görüntünün tamamını seçme, None – seçimi iptal etme, Invert – seçimi tersine çevirme ve Float – seçimi geçici bir katmana dönüştürme.

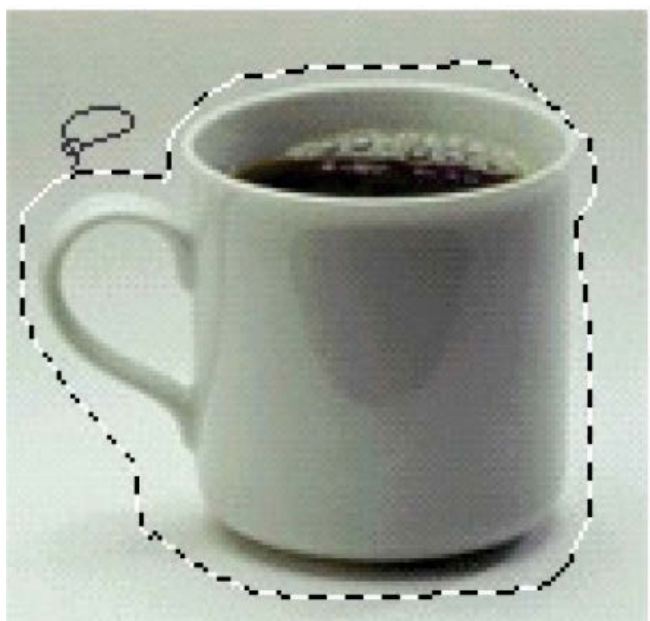
Seçim yapmak için birkaç araç vardır. Seçim yapmanın en basit yolu, ayrılmak istenen bölgenin doğrudan çizilmesidir. Bunun için araçlar grubundaki şu araçlar kullanılabilir:



- Dikdörtgen alan seçme aracı

- Elips seçme aracı

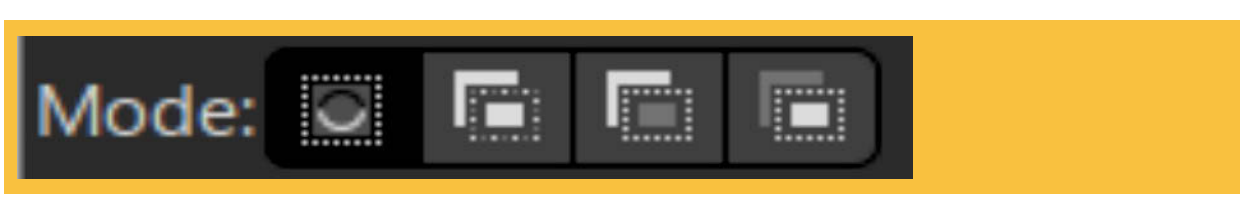
- Serbest seçim aracı



Resim 4.8: Serbest seçim aracıyla seçim

Seçim, Enter tuşuna basılarak veya başka bir görüntü düzenleme aracına tıklanarak onaylanır. Seçim yapılırken Shift tuşunun basılı tutulması, dikdörtgeni kareye, elipsi ise daireye dönüştürür. Seçim onaylanana kadar boyutu değiştirilebilir.

Seçim araçlarının seçilmesiyle, yeni seçim ile mevcut seçim arasındaki ilişkiyi belirleyen Mode seçeneği de kullanılabilir.



Resim 4.9: Mode seçeneği

Dört seçenek vardır:

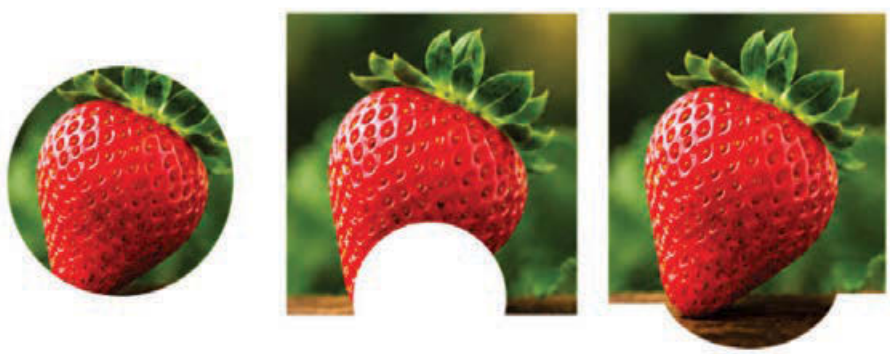
1. Replace the current selection – mevcut seçim iptal edilir ve yeni bir seçim yapılır.

2. Add to the current selection – mevcut seçim yeni seçimle birleştirilir (iki seçimin birleşimi).

3. Subtract from the current selection – mevcut seçimden yeni seçimle ortak olan bölümler silinir (fark – yeni seçimin alanı mevcut seçimden çıkarılır).

4. Intersect with the current selection – seçim, mevcut ve yeni seçimin ortak bölümlerini kapsar (mevcut ve yeni seçimin kesişimi).

Yeni seçimin mevcut seçimle birleştirilme yöntemi, yeni seçim çizilmeden önce seçilmelidir.



Resim 4.10: İki seçimin kesişimi, farkı ve birleşimi

GIMP'te renge göre seçim yapmak da mümkündür. Fuzzy Select () ve Select by Color () araçları, tıklanan noktanın rengine çok benzeyen bir alanın seçilmesini sağlar. Fuzzy Select ve Select by Color araçları arasındaki fark, ilkinin yalnızca başlangıç noktasının çevresindeki alanı seçmesi, ikincisinin ise başlangıç noktasının çevresinde olmak zorunda olmadan, görüntünün herhangi bir yerinde başlangıç noktasının rengine yeterince benzeyen tüm alanları seçmesidir.

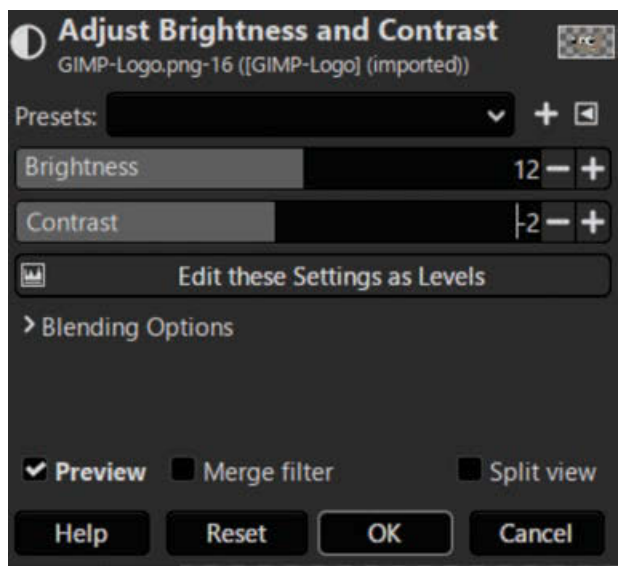
Seçim, makas (Scissors) aracı () ile de yapılabilir. Makas aracı, düzensiz şekle sahip görüntü bölümlerini seçmek için kullanılır. Kullanıcı nesnenin çevresine noktalar yerleştirir ve program renk ile kontrast farklılıklarına göre nesnenin kenarını otomatik olarak takip eder. Seçim kapatıldıktan sonra seçilen bölüm kopyalanabilir, taşınabilir, silinebilir veya ek olarak düzenlenebilir.

4.3.12. GÖRÜNTÜYÜ AYDINLATMA

Görüntü çok karanlık olduğunda veya bazı ayrıntılar yeterince görünür olmadığında kullanılır. GIMP'te bu değişiklik Colors - Brightness-Contrast menüsü üzerinden yapılabilir. Bu seçenek seçildiğinde, Brightness değerinin bir kaydırıcı yardımıyla artırılıp azaltılabildiği bir pencere açılır. Değer artırıldığında görüntü daha aydınlık, azaltıldığında ise daha karanlık olur.

4.3.13. GÖRÜNTÜ KONTRASTI

Görüntünün açık ve koyu bölümleri arasındaki farkı vurgulamak için kullanılır. Aynı pencerede, Contrast kaydırıcısı aracılığıyla kontrast değeri artırılabilir veya azaltılabilir. Kontrast artırıldığında açık bölgeler daha açık, koyu bölgeler ise daha koyu olur ve görüntü daha belirgin görünür. Kontrast azaltıldığında açık ve koyu bölgeler arasındaki fark azalır, bu nedenle görüntü daha soluk veya daha yumuşak görünebilir. Uygun görünüm elde edildikten sonra değişiklik OK düğmesine basılarak onaylanır.



Resim 4.11: Aydınlatma ve kontrastın ayarlanması

Seçim işlemleriyle ilgili çalışmalarda taşıma aracı da kullanılır.

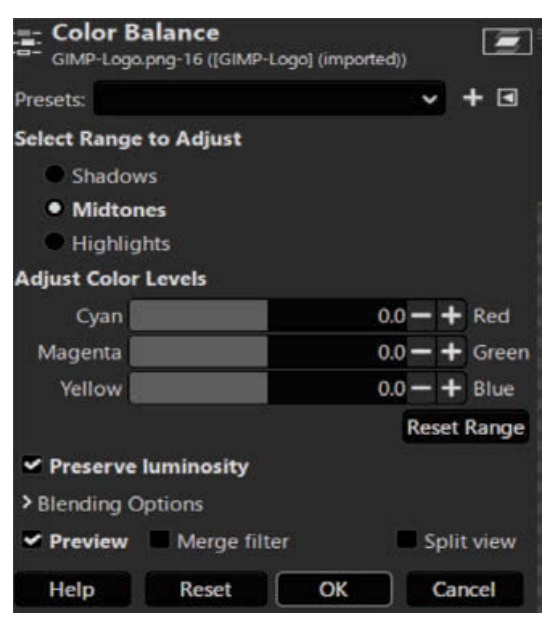
Seçilen alanın sınır çizgisinin kalınlığı da değiştirilebilir. Bunun için Select → Border seçenekleri seçilir ve ardından ayarlama penceresi açılır.

4.3.14. GÖRÜNTÜNÜN RENGİ

Colors > Color Balance komutu, renkleri dengeli olmayan görüntülerde renk düzeltmesi yapmak için kullanılır.

Koyu renkleri ayarlamak için Select Range to Adjust → Shadows seçilir. Midtones orta tonları, Highlights ise açık renkleri belirtir. Reset düğmesine basıldığında tüm ayarlar varsayılan değerlerine geri döner.

Görüntünün renkleri Colors > Hue-Saturation komutuyla da değiştirilebilir.



Resim 4.12: Color Balance penceresi

Resim 4.13: Hue-Saturation komutunun pratik uygulaması

4.3.15. GÖRÜNTÜ KLONLAMA

Bir görüntüdeki bir bölgenin kopyalanması ve başka bir bölgeye aktarılması için kullanılır.

GIMP'te bu işlem, araçlar panelinden seçilen Clone Tool ( ) aracıyla yapılır. Öncelikle görüntünün kaynak bölgesi, yani piksellerin kopyalanacağı alan seçilir. Bunun için Ctrl tuşuna basılır ve seçilen alanın üzerine tıklanır. Daha sonra fare işaretçisi kopyalamanın yapılacağı yere getirilir, farenin sol tuşuna basılır ve tuş bırakılmadan fare hareket ettirilir. Böylece seçilen içerik yeni konuma kopyalanır. Klonlama sırasında kaynak bölgenin renk, doku ve aydınlatma açısından uygun olmasına dikkat edilmelidir, böylece yapılan değişiklik doğal görünür. Bu araç genellikle küçük kusurları gidermek, istenmeyen bölümleri kapatmak veya görüntüdeki belirli öğeleri tekrarlamak için kullanılır.

4.3.16. GÖRÜNTÜ ONARMA ARACI

Healing () aracı, görüntüdeki leke, nokta veya çizik gibi küçük kusurları düzeltmek için kullanılır.

- Healing aracının simgesine tıklanır;

- Örnek alınacak yer belirlenir, Ctrl tuşu basılı tutulur ve bu noktaya tıklanır. Kaynak belirlendikten sonra Ctrl tuşu bırakılır. GIMP seçilen noktanın etrafına bir daire çizer;

- Fare işaretçisi düzeltilecek bölgenin üzerine getirilir. Alan fırçaya sığacak kadar küçükse çoğu zaman bir kez tıklamak yeterlidir. Düzeltilecek alan fırçadan daha büyükse, farenin sol tuşuna basılır ve alanın tamamı kaplanana kadar tuş bırakılmadan fare hareket ettirilir.

Gerekirse, en iyi sonucu elde etmek için aynı alan birkaç kez düzeltilebilir, düzeltilen bölge çevresine yeterince iyi uyum sağlayana kadar işlem tekrarlanabilir.

4.3.17. METİNLE ÇALIŞMA

GIMP'te görüntü düzenlemenin yanı sıra metin eklenebilir ve düzenlenebilir. Bunun için araçlar panelinden Text Tool () aracı kullanılır. Metin, yazı tipi, boyut, renk, hizalama ve harfler ile satırlar arasındaki boşluk seçenekleri kullanılarak düzenlenebilir. Metin eklenirken GIMP otomatik olarak yeni bir metin katmanı oluşturur. Bu sayede metin, görüntünün diğer bölümlerinden bağımsız olarak düzenlenebilir. Araç seçildikten sonra görüntüde metnin ekleneceği yere tıklanır ve istenen metin yazılır.

SONUÇ:

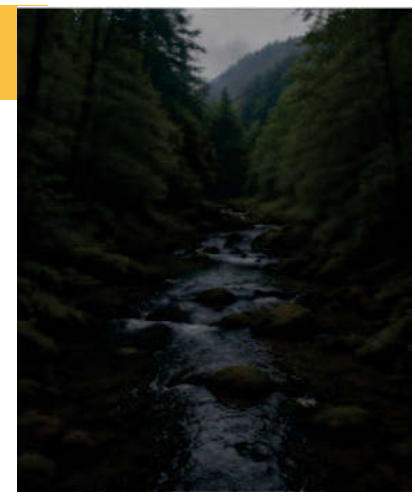
GIMP'in çalışma ortamı üç panelden oluşur: araçlar paneli, görüntü paneli ve yardımcı panel. GIMP'te oluşturulan görüntüler XCF formatında, .xcf uzantısıyla kaydedilir. Görüntüleri başka bir formatta kaydetmek için File>Export as komutu kullanılır. Image → Scale Image görüntünün boyutunu değiştirmek, Image → Canvas Size tuvali ayarlamak ve Crop Tool görüntünün bir bölümünü kırpmak için kullanılır. Image → Transform → Rotate / Flip üzerinden döndürme ve çevirme, Rectangle Select, Ellipse Select ve Free Select araçlarıyla nesne seçimi, aynı zamanda bölümlerin silinmesi veya taşınması yapılabilir. Eraser Tool, Clone Tool ve Healing Tool gibi araçlar görüntünün bölümlerini kaldırmak veya düzeltmek için kullanılır. Görüntünün görünümünü ayarlamak Colors → Brightness-Contrast, Color Balance ve Hue-Saturation komutlarıyla yapılır. Görüntüye metin ekleme Text Tool aracıyla gerçekleştirilir ve böylece düzenli ve yaratıcı grafik çözümleri oluşturulur.

BİLGİLERİ TEKRARLAMA SORULARI

1. Görüntünün boyutunu değiştirme ile tuvalin boyutunu değiştirmeyi karşılaştırınız. Hangi durumlarda birini, hangi durumlarda diğer işlemi kullanırdınız?
2. GIMP'te Save As ve Export As komutları arasındaki farkı analiz ediniz. Projenin XCF formatında kaydedilmesi, son görüntünün ise PNG veya JPG olarak dışa aktarılması neden önemlidir?
3. Görüntü RGB renk modelinden Grayscale renk modeline dönüştürülürse ne olacağını açıklayınız. Görüntüde hangi bilgiler kaybolur?
4. Fuzzy Select ve Select by Color araçlarını karşılaştırınız. Hangi durumda birinin, hangi durumda diğerinin kullanılması daha uygundur?
5. Clone Tool ile Healing aracı arasındaki farkı analiz ediniz. Küçük kusurların giderilmesinde Healing neden daha doğal bir sonuç verebilir?
6. Add, Subtract ve Intersect seçim modlarının mevcut seçimi nasıl etkilediğini açıklayınız. Her modun yararlı olabileceği birer örnek veriniz.
7. Görüntünün bir bölümünü kırpma ile görüntünün bölümlerini seçip silmeyi karşılaştırınız. Sonuç arasındaki fark nedir?
8. Aydınlatma ve kontrast değişikliğinin görüntü kalitesini nasıl iyileştirebileceğini veya kötüleştirebileceğini analiz ediniz.
9. Bir fotoğraftaki küçük bir lekeyi kaldırmak için en uygun aracın Clone Tool, Healing veya Eraser olup olmadığını değerlendiriniz. Seçiminizi açıklayınız.
10. Görüntünün kontrastını azaltmak ne zaman uygun olur? Kontrastı artırmak her zaman görüntüyü iyileştirir mi?
11. Eraser aracının görüntünün bir bölümünü her zaman aynı şekilde silip silmediğini değerlendiriniz. Katmanın saydam bir arka plana sahip olup olmaması bunu nasıl etkiler?
12. Düzensiz şekle sahip bir nesneyi seçmek için en uygun aracı seçiniz ve neden bu aracın en uygun olduğunu açıklayınız.
13. GIMP'teki çalışma ortamının önemini değerlendiriniz. Paneller, menüler, cetveller ve durum çubuğu görüntü düzenleme sırasında kullanıcıya nasıl yardımcı olur?
14. Karanlık, biraz eğik ve ana nesnenin çevresinde gereksiz boşluk bulunan bir fotoğrafı düzenlemek için bir işlem sırası oluşturunuz. Hangi araç ve komutları, hangi sırayla kullanırdınız?
15. Aşağıdaki uygulamalı alıştırmaları da yapınız.

Görüntüyü, ana nesne odakta kalacak şekilde kırpın ve ardından boyutunu değiştirin.

Koyu ve soluk bir görüntünün düzeltmesini yapın.



Çevredeki çim bölümlerini klonlayın ve boş alanı doldurun, böylece görüntü düzenli ve doğal görünsün.

Bir seçim aracı kullanarak çiçeğin rengini değiştirin.

Verilen görüntüyü oluşturun.

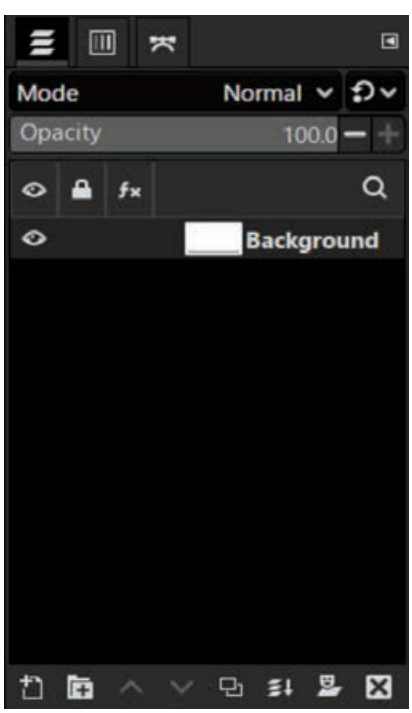
4.4

GRAFİKTE KATMANLAR (LAYERS)

GIMP'te görüntü düzenleme sırasında fotoğraf, metin, şekil, çizim veya başka bir görüntü gibi birden fazla öğe sıklıkla kullanılır. Daha kolay çalışmak için bu öğeler ayrı katmanlara yerleştirilebilir. Katmanlar, görüntünün her bölümünün diğer bölümleri doğrudan değiştirilmeden ayrı ayrı düzenlenmesini sağlar.

Katman, başka bir yüzeyin üzerine yerleştirilmiş saydam bir yüzey olarak düşünülebilir. Bir katmanda fotoğraf, diğerinde metin, üçüncüde ise şekil bulunuyorsa, bunların hepsi birlikte son görüntüyü oluşturur. Bununla birlikte, her katman ayrı ayrı taşınabilir, silinebilir, gizlenebilir, yeniden düzenlenebilir veya değiştirilebilir. İşte bu nedenle katmanlar, görüntünün düzenli ve hassas bir şekilde işlenmesi için önemlidir.

GIMP'te katmanlar **Layers** panelinde gösterilir. Panel görünür değilse Windows → Dockable Dialogs → Layers komutuyla açılabilir. Bu panelde görüntüde kullanılan tüm katmanlar, sıraları, adları ve görünürlükleri gösterilir. Aktif katman, panelde seçili olan katmandır ve araçlar ile komutlar bu katmana uygulanır.



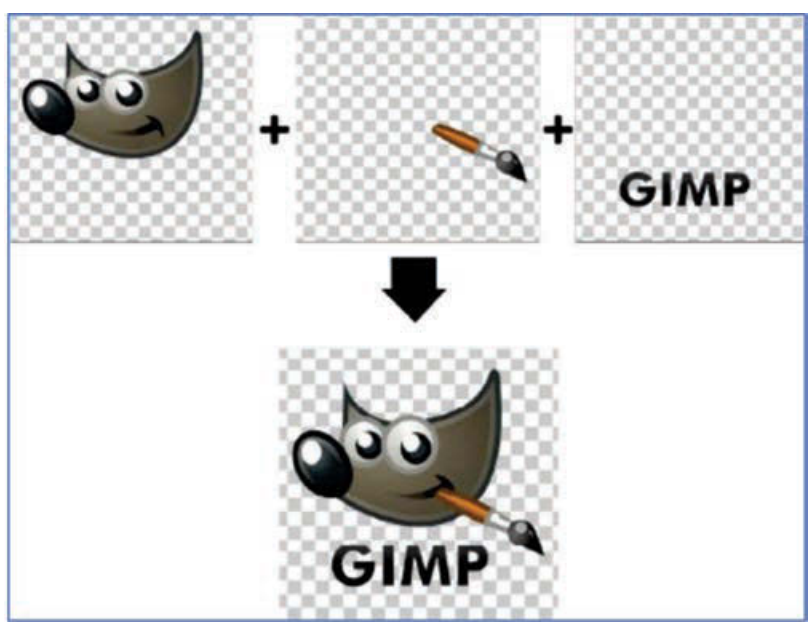
1. Yeni katman oluşturma
2. Gruplandırılmış katman oluşturma
3. Katmanın sıralama düzenindeki yerini bir konum yukarı değiştirme
4. Katmanın sıralama düzenindeki yerini bir konum aşağı değiştirme
5. Aktif katmanı çoğaltma
6. Aktif katmanı altındaki katmanla birleştirme
7. Katman maskesi oluşturma
8. Katmanı silme

Resim 4.17: Katmanlar paneli

Yeni katman, Layer → New Layer komutuyla veya Layers panelindeki yeni katman düğmesiyle eklenir. Yeni katman oluşturulurken adı, boyutu ve doldurma türü belirlenebilir, örneğin saydam arka plan, beyaz arka plan veya belirli bir arka plan rengi. Katmanlara ad vermek, özellikle görüntü birden fazla öğe içerdiğinde kullanışlıdır.

Katmanların sıralaması, hangi öğelerin diğerlerinin önünde görüneceğini etkiler. Layers panelinde daha yukarıda bulunan katman, altında bulunan katmanların önünde gösterilir. Sıralama, katmanı panelde yukarı veya aşağı sürükleyerek değiştirilebilir. Ayrıca katmanı yukarı taşımak için Layer → Stack → Raise Layer, aşağı taşımak için Layer → Stack → Lower Layer komutları kullanılabilir.

Görüntünün düzenlenmesi tamamlandıktan sonra katmanları birleştirmek en iyisidir. Layer → Merge Down komutuyla komşu katmanlar birleştirilir. Örneğin, görüntü katmanının üzerindeki metin katmanı bu komutla görüntüyle tek bir katmanda birleştirilir. Ayrıca o anda görünür olan tüm katmanlar ("göz" simgesi açık olanlar) birleştirilebilir. Bu işlem Image → Merge Visible Layers komutuyla gerçekleştirilir. Projede birden fazla katman bulunduğunda, ancak tüm görünür öğelerin tek bir katmanda bulunması gerektiğinde kullanışlıdır. Image → Flatten Image komutuyla tüm katmanlar tek bir katmanda birleştirilir ve saydamlık kaldırılır. Genellikle katmanları desteklemeyen formatlarda (örneğin JPG) kaydetmeden önce kullanılır.



Resim 4.18. Katmanlar kullanılarak oluşturulmuş görüntü

Katmanın görünürlüğü, Layers panelinde katmanın adının yanında bulunan göz simgesi aracılığıyla kontrol edilir. Göz simgesi gösterildiğinde katman görünürdür. Göz simgesi kapatıldığında, o katmanın içeriği geçici olarak görünmez, ancak silinmez. Bu, görüntünün farklı görünümlerini karşılaştırmak veya belirli bir öğenin geçici olarak gösterilmemesi gerektiğinde kullanışlıdır.

Katman, Layers panelinden seçilerek ve Layer → Delete Layer komutu kullanılarak veya paneldeki silme düğmesine tıklanarak silinebilir. Mevcut bir katmanın kopyasının oluşturulması gerekiyorsa Layer → Duplicate Layer komutu kullanılır. Bu, bir değişiklik yapmayı denemek ve aynı zamanda katmanın orijinal sürümünü korumak istediğimizde kullanışlıdır.

Resim 4.19: Yedi katmandan oluşturulmuş görüntü

Birden fazla katmanla çalışırken her öğe ayrı ayrı düzenlenebilir. Örneğin, arka plan olarak bir fotoğraf yerleştirilebilir, üzerine Text Tool aracıyla metin eklenebilir ve ardından yeni bir katmana şekil veya başka bir görüntü eklenebilir. Her öğe ayrı bir katmanda bulunduğundan, konumu, boyutu, rengi, görünürlüğü veya sıralaması kolayca değiştirilebilir.

GIMP'te katmanın opaklığı da Layers panelindeki Opacity seçeneği üzerinden değiştirilebilir. Opaklık değeri azaltıldığında katman daha saydam hâle gelir ve altındaki katmanlar kısmen görülebilir. Bu özellik, öğeler arasındaki keskin sınırları azaltmak, öğeler arasındaki renklerin yumuşak bir şekilde geçişini sağlamak, öğeleri daha düzgün bir şekilde birleştirmek veya birden fazla öğeyi tek bir kompozisyonda birleştirmek için kullanılır.

Düzenleme tamamlandıktan sonra görüntü, File → Save As aracılığıyla XCF formatında çalışma projesi olarak kaydedilebilir, böylece tüm katmanlar korunur. Görüntünün örneğin bir belgede, sunumda veya web sayfasında normal bir görüntü olarak kullanılması gerekiyorsa, File → Export As aracılığıyla PNG veya JPG gibi bir formatta dışa aktarılır. Dışa aktarma sırasında katmanlar tek bir son görüntüde birleştirilir.

SONUÇ:

Görüntülerin daha kolay işlenmesi için katmanlar, tek bir görüntüdeki ayrı öğeleri düzenlemenin bir yolu olarak kullanılır. Her katman üzerinde bağımsız olarak çalışılabilir, böylece diğer katmanlardaki görüntüler değiştirilmez. Katmanlarla çalışma, aktif katmanın mavi renkle işaretlendiği Layers paneli üzerinden gerçekleştirilir. Görüntünün işlenmesinden sonra tüm katmanlar Image → Flaten Image komutuyla birleştirilir. Bir katman şu araç ve komutlarla değiştirilebilir: ekleme (Layer → New Layer), silme (Layer → Delete Layer), çoğaltma (Layer → Duplicate Layer), sıralamayı değiştirme (Layer → Stack → Raise/Lower Layer) ve opaklığı değiştirme (Layers panelindeki Opacity).

Öğrenciler, katmanların görünürlüğünü ve sıralamasını değiştirebilecek ve böylece hangi öğenin diğerinin üzerinde veya altında görüntüleneceğini kontrol edebilecekler. Birden fazla katmanı birleştirerek daha karmaşık ve daha yaratıcı grafik çözümleri oluşturabilecekler.

BİLGİLERİ TEKRARLAMA SORULARI

1. GIMP'te katmanlar nedir?
2. Görüntü düzenleme sırasında katmanlar neden kullanılır?
3. Katmanlar GIMP'te nerede gösterilir?
4. Görünür değilse Layers paneli nasıl açılır?
5. Aktif katman nedir?
6. GIMP'te yeni bir katman nasıl eklenir?
7. Katmanlara isim vermek neden kullanışlıdır?
8. Katmanların sıralaması görüntünün görünümünü nasıl etkiler?
9. Katmanların sıralaması nasıl değiştirilebilir?
10. Katmanın yanında bulunan göz simgesi ne işe yarar?
11. Bir katmanı gizlemek ile silmek arasındaki fark nedir?
12. Bir katmanı çoğaltmak ne zaman yararlı olabilir?
13. Metnin ayrı bir katmanda bulunması neden iyidir?
14. GIMP'te görüntünün çalışma sürümü hangi formatta kaydedilir?
15. Save As ile Export As arasındaki fark nedir?
16. GIMP'te yeni bir görüntü açın. En az üç katman oluşturun: arka plan, şekil ve metin. Arka planı bir renkle doldurun, ikinci katmana bir dikdörtgen veya başka bir şekil çizin ve üçüncü katmana kısa bir metin ekleyin.
17. Katmanları kullanarak internette güvenli davranış konusunda bir poster oluşturun.

4.5

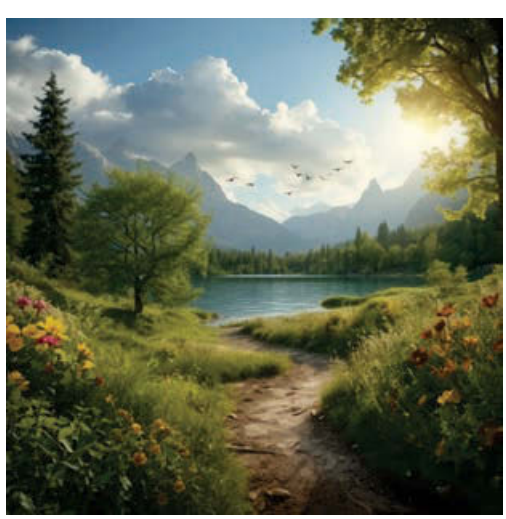
GRAFİK İÇİN ÜRETKEN YAPAY ZEKÂ

Üretken yapay zekâ, kullanıcının verdiği yönergelere dayanarak yeni içerikler oluşturabilen bir teknolojidir. Grafik alanında görüntüler, illüstrasyonlar, posterler, bannerlar, logolar, arka planlar ve diğer görsel öğeleri oluşturmak için kullanılır. Kullanıcı görüntüyü tamamen elle çizmek veya düzenlemek yerine bir açıklama girebilir ve yapay zekâ aracı bu açıklamaya dayanarak görsel bir çözüm önerebilir.

Üretken yapay zekânın grafikteki en yaygın uygulamalarından biri text-to-image'dır. Bu terim, metinsel bir açıklamadan görüntü oluşturulması anlamına gelir. Kullanıcı, görüntüde ne görmek istediğini açıklayan bir veya daha fazla cümle girer ve araç bu yönergelere göre bir görüntü oluşturur. Örneğin, "kitapların bulunduğu, açık renkli ve büyük başlıklı bir okul fuarı posteri" açıklaması girilirse, yapay zekâ aracı bu açıklamaya uygun bir görüntü oluşturabilir.

Yapay zekâ sistemine verilen metinsel yönergeye prompt denir. Prompt açık, belirli ve yeterince açıklayıcı olmalıdır. Prompt'ta görüntüde bulunması gereken nesneler, görüntünün stili, renkler, öğelerin düzeni, boyut, atmosfer ve grafiğin amacı belirtilebilir. Prompt ne kadar kesin olursa, elde edilen sonucun kullanıcının hayaline o kadar yakın olma olasılığı artar.

Örneğin, prompt "doğayla ilgili bir görüntü oluştur" çok geneldir. Daha iyi bir prompt şöyle olabilir: "Yeşil orman, mavi gökyüzü ve sol üst köşede başlık için boş alan bulunan, dağ manzaralı yatay bir banner oluştur." İkinci örnekte araç, görüntünün içeriği, renkleri, düzeni ve amacı hakkında daha fazla bilgi edinir.



Resim 4.20. Genel talimatların uygulanması, kaynak: Leonardo.ai

Resim 4.21. Belirli talimatların uygulanması, kaynak: Leonardo.ai

Grafik içerikleri yapay zekâ kullanarak oluşturmak için kullanılabilecek birçok uygulama ve çevrim içi araç vardır. Bunlara Adobe Firefly, Canva Magic Media, Microsoft Designer, Midjourney, ChatGPT (görüntü oluşturma özelliği ile), Leonardo AI, Ideogram ve Freepik AI örnek olarak verilebilir. Bu araçlar, metinsel bir prompt temelinde görüntüler, illüstrasyonlar, posterler, bannerlar ve diğer görsel içeriklerin oluşturulmasına olanak sağlar. Bazıları daha çok sanatsal görüntülere, bazıları grafik tasarıma ve sosyal medya materyallerine, bazıları ise mevcut görüntülerin düzenlenmesine ve geliştirilmesine yöneliktir.

Görüntü oluşturmak için kullanılan YZ araçları, tasarım sırasında yardımcı olarak kullanılabilir. Başlangıç fikirlerinin, görsel önerilerin ve aynı tasarımın farklı seçeneklerinin hızlı bir şekilde oluşturulmasını sağlar. Kullanıcı birden fazla sonucu karşılaştırabilir, en uygun olanı seçebilir ve daha sonra bunu bir grafik düzenleme programıyla ek olarak düzenleyebilir. Bu şekilde YZ, yaratıcı çalışmanın tamamen yerini almaz, planlama, ilham alma ve grafik çözümlerinin daha hızlı hazırlanmasına yardımcı olabilir.

Üretken yapay zekâ, grafiklerin bir türden diğerine dönüştürülmesine veya uyarlanmasına da yardımcı olabilir. Raster görüntü, piksellerden oluşur ve genellikle fotoğraflar ile çok sayıda ayrıntı içeren karmaşık görüntüler için kullanılır. Vektör görüntü ise çizgiler, şekiller ve matematiksel olarak tanımlanan öğelerden oluşur ve bu nedenle kalite kaybı olmadan büyütülebilir. Belirli YZ araçlarının veya YZ destekli araçların yardımıyla raster görüntü basitleştirilebilir ve örneğin logo, simge veya illüstrasyon gibi vektörel bir görünüme dönüştürülebilir. Öte yandan vektör grafik, bir belgede, sunumda veya web sayfasında kullanılması gerektiğinde raster görüntü olarak dışa aktarılabilir.

YZ araçlarını kullanırken sonuçların analiz edilmesi önemlidir. Oluşturulan görüntü her zaman doğru, uygun veya kaliteli olmayabilir. Bazen şekillerde, metinlerde, oranlarda, renklerde veya öğelerin düzeninde düzensizlikler olabilir. Bu nedenle kullanıcı, görüntünün belirlenen amaca uygun olup olmadığını, açık ve anlaşılır olup olmadığını, görsel olarak düzenli olup olmadığını ve amaçlanan kullanım için uygun olup olmadığını dikkatlice kontrol etmelidir.

Oluşturulan grafikler elle oluşturulan grafiklerle karşılaştırılabilir. Elle oluşturulan grafik, kullanıcı şekilleri, renkleri, metni ve düzeni kendisi belirlediği için her öğe üzerinde daha fazla kontrol sağlar.

YZ grafikleri ise fikirlerin ve görsel önerilerin daha hızlı elde edilmesini sağlar, ancak bazen ek kontrol ve düzenleme gerektirir. En iyi sonuçlar çoğu zaman YZ'nin yardımcı bir araç olarak kullanılması ve kullanıcının son tasarımı ek olarak düzenleyip şekillendirmesiyle elde edilir.

Tasarımda YZ araçları posterler, bannerlar, kapak görüntüleri, reklamlar, illüstrasyonlar, arka planlar ve sosyal medya için görsel materyaller oluşturmak amacıyla kullanılabilir. Örneğin öğrenci, bir YZ aracıyla bir arka plan oluşturabilir, ardından bunu bir görüntü işleme programında açarak metin ekleyebilir, renkleri değiştirebilir, görüntünün bazı bölümlerini kaldırabilir ve son posteri hazırlayabilir. Böylece öğrenilen grafik işleme teknikleri ile yapay zekânın olanakları birleştirilir.

YZ araçlarıyla çalışırken sorumlu kullanım kurallarına da uyulmalıdır. Hakaret eden, aldatan veya kişileri ve olayları yanlış şekilde gösteren içerikler oluşturulmamalıdır. Ayrıca görüntünün kamuya açık paylaşım, okul projesi veya ticari amaçla kullanılıp kullanılamayacağına dikkat edilmelidir. Kullanıcı her zaman elde edilen sonucu eleştirel bir şekilde değerlendirmeli ve gerekirse çalışmanın hazırlanmasında bir YZ aracının kullanıldığını belirtmelidir.

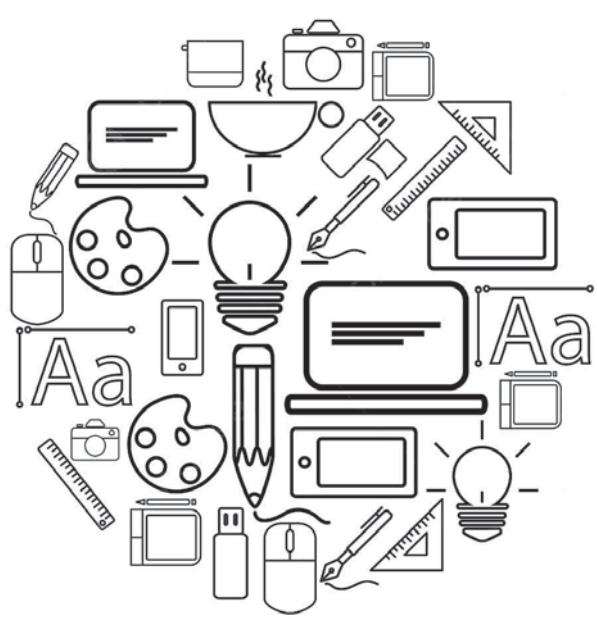
Üretken yapay zekâ, modern grafik işleme alanında yararlı bir destek sunmaktadır. Hızlı bir şekilde fikirler ve görsel çözümler oluşturulmasını sağlar, ancak başarılı bir tasarım hâlâ kullanıcının bilgisine, değerlendirmesine ve yaratıcılığına bağlıdır. Bu nedenle kullanıcının açık bir prompt oluşturmayı, elde edilen sonuçları analiz etmeyi ve YZ araçlarını görüntü işleme için kullanılan manuel tekniklerle birleştirmeyi bilmesi önemlidir.

SONUÇ:

Dijital görüntüler oluşturmak için YZ aracı kullanmak, elde edilen sonuçların kalitesini doğrudan etkileyen basit ve açık talimatlar (prompt'lar) oluşturulduğunda daha verimli olur. Oluşturulan görüntülerin analiz edilmesi, YZ tarafından oluşturulan içeriklerin doğruluğunu, yaratıcılığını ve uygunluğunu değerlendirme becerisini ve eleştirel düşünmeyi geliştirir. Yapay zekâ ile oluşturulan görüntüler ile elle hazırlanmış grafik çözümlerinin karşılaştırılması, her iki yaklaşımın avantajlarını ve sınırlamalarını ortaya koyar. YZ, insanın yaratıcı ifadesinin yerini almayan, onu tamamlayan yararlı bir araç olabilir.

BİLGİLERİ TEKRARLAMA SORULARI

1. Üretken yapay zekâ nedir?
2. Üretken yapay zekâ grafik alanında nasıl kullanılır?
3. Text-to-image terimi ne anlama gelir?
4. Prompt nedir?
5. Görüntü oluşturmak için prompt'a hangi bilgiler eklenebilir?
6. Görüntü oluşturmak için kullanılabilecek bir YZ aracı örneği veriniz.
7. Bir YZ aracı tarafından elde edilen görüntü kullanılmadan önce neden analiz edilmelidir?
8. YZ ile oluşturulan bir görüntü ile elle oluşturulan bir grafiği karşılaştırınız. Her iki yöntemin avantajları ve dezavantajları nelerdir?
9. Aşağıdaki prompt'u geliştiriniz: "Okul için bir poster oluştur." Konu, renkler, düzen ve metin hakkında ayrıntılar ekleyiniz.
10. "Dijital güvenlik" konusu için bir banner prompt'u oluşturunuz.
11. Bir YZ aracının yardımıyla "Dijital güvenlik" konusunda bir poster için başlangıç görüntüsü oluşturunuz. Öncelikle görüntünün neleri içermesi gerektiğini, hangi renklerin kullanılacağını ve başlık için nerede boş alan bulunacağını belirten bir prompt hazırlayınız. Daha sonra elde edilen sonucu analiz ederek görüntünün poster için uygun olup olmadığına karar veriniz. Görüntüyü bir görüntü işleme programında açınız, başlık ve kısa bir metin ekleyiniz ve gerekirse boyut, renk veya düzen üzerinde düzeltmeler yapınız. Son posteri çalışma projesi olarak kaydediniz ve görüntü olarak dışa aktarınız.



KONU TEKRARI İÇİN SORULAR

Doğru cevabı yuvarlayınız.

1. Multimedya nedir?

A. Fotoğrafların bir grafik programında düzenlenmesi
B. İçeriği aktarmak için birden fazla öğenin birleştirilmesi
C. Görüntülerin farklı formatlarda kaydedilmesi
Ç. Metnin tabloya dönüştürülmesi

2. Raster görüntü nedir?

A. Çizgilerden ve geometrik şekillerden oluşan görüntü
B. Piksellerden oluşan görüntü
C. Yazdırılamayan görüntü
Ç. Her zaman şeffaf arka plana sahip görüntü

3. Vektörel görüntü nedir?

A. Piksellerden oluşan görüntü
B. Ses ve videodan oluşan görüntü
C. Çizgiler, eğriler ve şekillerden oluşan görüntü
Ç. Yalnızca GIMP'te açılabilen görüntü

4. Fotoğraflar için en yaygın kullanılan format hangisidir?

A. JPG B. SVG C. AI D. DXF

5. Farklı boyutlarda kullanılacak bir logo için en uygun format hangisidir?

A. JPG B. PNG C. XCF D.GIF

6. GIMP, katmanları içeren çalışma sürümünü hangi formatta kaydeder?

A. JPG B. PNG C. XCF D. GIF

7. Text-to-image ne anlama gelir?

A. Görüntüyü metne dönüştürme
B. Metinsel bir açıklamadan görüntü oluşturma
C. Görüntüden metni silme
Ç. Metni görüntü olarak kaydetme

8. Bir görüntü hem bir web sitesinde küçük bir ikon hem de bir posterde büyük bir işaret olarak kullanılacak. Hangi grafik türü en uygundur?

A. Raster grafik, çünkü piksellerden oluşur

B. Vektör grafik, çünkü netliğini kaybetmeden büyütülebilir

C. JPG fotoğrafı, çünkü her zaman küçük boyutludur

Ç. GIF dosyası, çünkü her zaman en yüksek kalitededir

9. Düşük çözünürlüklü bir raster görüntü çok fazla büyütülürse, büyük olasılıkla ne olur?

A. Daha keskin hâle gelir

B. Vektör görüntüye dönüşür

C. Bulanık veya küçük karelerden oluşuyormuş gibi görünür

Ç. Şeffaf bir arka plan kazanır

10. GIMP'te bir öğrenci metni düzenliyor, ancak yaptığı değişiklikler yanlış öğeye uygulanıyor. En olası neden nedir?

A. Yanlış aktif katman seçilmiştir

B. Görüntü çok küçüktür

C. Yazı tipi çok büyüktür

Ç. Belge XCF olarak kaydedilmiştir

11. Metin katmanı koyu arka plan katmanının altına yerleştirilirse ne olabilir?

A. Metin görünmeyebilir

B. Metin otomatik olarak büyür

C. Arka plan silinir

Ç. Görüntü PNG olarak dışa aktarılır

12. Katmanı gizlemek ile katmanı silmek arasındaki fark nedir?

A. Aralarında fark yoktur

B. Gizlemek geçicidir, silmek ise katmanı kalıcı olarak kaldırır

C. Silmek yalnızca opaklığı azaltır

Ç. Gizlemek görüntünün boyutunu değiştirir

13. Görüntü çok karanlıksa, başlangıç düzeltmesi için hangi komut en uygundur?

A. Colors → Brightness-Contrast

B. File → Export As

C. Layer → Delete Layer

Ç. Select → None

14. Küçük bir lekeyi kaldırırken Healing aracı neden çoğu zaman Clone Tool'dan daha doğal bir sonuç verir?

A. Çünkü tüm görüntüyü otomatik olarak siler

B. Çünkü yapılan düzeltmeyi çevredeki alanla uyumlu hâle getirir

C. Çünkü görüntüyü her zaman vektörel görüntüye dönüştürür

Ç. Çünkü görüntüye metin ekler

15. Canvas Size yerine Crop Tool ne zaman kullanmak daha uygundur?

A. Ana nesnenin etrafındaki gereksiz bir bölümün kaldırılması gerektiğinde

B. Görüntünün etrafına yeni bir alan eklenmesi gerektiğinde

C. Metnin renginin değiştirilmesi gerektiğinde

Ç. Yeni bir katman oluşturulması gerektiğinde

16. Hangi prompt en iyi şekilde formüle edilmiştir?

A. Güzel bir şey yap.

B. Okul için bir görüntü.

C. Açık renkli bir arka plan, kitaplar, öğrenciler ve başlık için boş alan içeren, okul kitap fuarı için yatay bir banner oluştur.

Ç. Renk yap.

17. Projeyi XCF olarak kaydetmek ne zaman en uygundur?

A. Görüntünün daha sonra katmanlar hâlinde yeniden düzenlenmesi gerektiğinde

B. Görüntünün hemen fotoğraf olarak yazdırılması gerektiğinde

C. Görüntü hiçbir öğe içermediğinde

Ç. Metnin silinmesi gerektiğinde

18. YZ tarafından oluşturulan bir arka plan, başlık ve ek bir görüntü içeren bir poster hazırlanacaksa, en uygun çalışma sırası hangisidir?

A. Önce posteri dışa aktarmak, ardından metin eklemek

B. Arka planı oluşturmak, sonucu kontrol etmek, bir grafik programında düzenlemek ve metin eklemek

C. Arka planı silmek ve boş bir görüntü olarak kaydetmek

D. Tüm öğeler için yalnızca tek bir katman kullanmak

19. Aynı posterin biri logolu, diğeri logosuz olmak üzere iki farklı versiyonu oluşturulacaksa, GIMP'te hangi özellik en çok yardımcı olur?

A. Logonun bulunduğu katmanın görünürlüğünü açıp kapatmak

B. Ekran çözünürlüğünü değiştirmek

C. Tüm katmanları silmek

D. Hesap makinesini açmak

20. Bir öğrencinin gökyüzünde küçük bir leke bulunan bir fotoğrafı var. En uygun işlem hangisidir?

A. Tüm görüntüyü SVG'ye dönüştürmek

B. Alanı düzeltmek için Healing veya Clone Tool aracını kullanmak

C. Görüntüyü 10 piksele küçültmek

D. Ses eklemek

21. GIMP'te arka plan, şekil, görüntü ve metin içeren basit bir poster oluşturulacaksa, öğeleri düzenlemenin en iyi yolu nedir?

A. Hepsini tek bir katmana yerleştirmek

B. Her ana öğeyi ayrı bir katmana yerleştirmek

C. Metni girdikten sonra silmek

D. Görüntüyü yalnızca ses olarak kaydetmek

22. YZ aracını görüntünün manuel olarak işlenmesiyle en iyi şekilde birleştiren plan hangisidir?

A Yapay zekâ aracının başlangıç görüntüsünü oluşturması, kullanıcının ise görüntüyü analiz etmesi, düzenlemesi ve göreve uygun hâle getirmesi

B. Kullanıcının hiçbir şeyi kontrol etmeden görüntüyü hemen yayımlaması

C. Herhangi bir analiz yapmadan yalnızca manuel düzenleme kullanılması.

KONU 5

KONU 5: WEB GELİŞTİRME

Web geliştirmenin temelleri, her gün kullandığımız web sitelerinin nasıl çalıştığını öğrenmenin yaratıcı bir yoludur. Web siteleri oluşturmak yaratıcılığı ve mantıksal düşünmeyi bir araya getirir. Her web sitesi üç temel teknoloji kullanılarak oluşturulur: HTML, temel yapıyı oluşturur, CSS, görünümü ve renkleri düzenler, JavaScript ise sayfaya etkileşim ve dinamizm ekler.

Bu temelleri öğrendiğinde kendi web sitelerini oluşturabilecek ve fikirlerini gerçek dijital projelere dönüştürebileceksin. Ayrıca sitenin telefon, tablet veya bilgisayarda iyi görünmesini ve kullanımının kolay olmasını nasıl sağlayacağını anlaman da önemlidir.

Bu, web geliştirmeye başlamak isteyen ve gelecekte bu meslekle ilgilenmek isteyen herkes için ilk adımdır.

YENİ KAVRAMLAR

- web sayfasının yapısı, HTML belgesi, temel HTML etiketleri – heading (başlık), paragraph (paragraf), image (görüntü), link (bağlantı), CSS, seçiciler, renkler, yazı tipleri, arka plan, web sayfası öğelerinin temel düzeni (layout), uyarlanabilir (responsive) tasarımın temelleri, farklı cihazlara uyarlama, YZ talimatları (prompt).

ZATEN BİLİYORSUN:

- Farklı internet servislerinin sunduğu olanakları.
- WWW servisinin internetteki rolünü.
- Web 1.0, Web 2.0, Web 3.0, Web 4.0 ve Web 5.0'a örnekler vermeyi.

AŞAĞIDAKİ KONULARDA ŞUNLARI ÖĞRENECEKSİNİZ:

- Web sayfasının temel yapısını açıklamayı ve web içerikleri oluşturmak ve düzenlemek için HTML ve CSS kullanmayı;
- Farklı cihazlar için web sayfalarını uyarlarken temel düzen ve uyarlanabilir (responsive) tasarım ilkelerini uygulamayı;
- Web içerikleri oluşturmak için yapay zekâ araçlarını kullanmayı;
- Talimatlar (prompt) oluşturmayı, elde edilen sonuçları analiz etmeyi ve YZ tarafından oluşturulan kodu düzenlemeyi;
- Manuel olarak hazırlanmış ve otomatik olarak oluşturulmuş kodu bir arada kullanarak basit bir web projesi oluşturmayı.

TEKRARLAMA ALIŞTIRMALARI:

- En az üç farklı internet servisini sıralayın ve ne amaçla kullanıldıklarını açıklayın.
- WWW servisinin, e-posta veya FTP gibi diğer internet servisleriyle karşılaştırıldığında bilgiye erişim biçimini nasıl etkilediğini açıklayın.
- Kendi kelimelerinizi kullanarak Web 1.0 ve Web 2.0 arasındaki temel farkların neler olduğunu açıklayın.
- Web 3.0'a somut bir örnek verin ve bu servisin internet olanaklarını nasıl geliştirdiğini açıklayın.
- Kullanıcılarla etkileşim açısından Web 2.0 ve Web 4.0'ı karşılaştırın ve temel özelliklerinin neler olduğunu açıklayın.
- Web 5.0 uygulaması için kendi örneğinizi oluşturun ve bu uygulamanın günlük yaşamda kullanıcılara nasıl yardımcı olacağını açıklayın.

5.1

WEB SAYFASI KAVRAMI

Web sayfası, Google Chrome, Mozilla Firefox veya Microsoft Edge gibi bir web tarayıcısında (browser) görüntülenen dijital bir belgedir. Daha geniş bir sistem olan web sitesi, birbiriyle bağlantılı birden fazla web sayfasının oluşturduğu bütündür. Her web sitesi, birbirine hiper bağlantılarla bağlanmış bir veya daha fazla web sayfasından oluşur. Bir web sunucusunda barındırılır ve web sayfalarının kullanıcılara sunulmasını sağlayan yazılım kuruludur.

Her web sayfasının, kullanıcıların internet üzerinden bu sayfaya erişmesini sağlayan URL (Uniform Resource Locator) olarak bilinen benzersiz bir adresi vardır. Web sayfaları çoğunlukla HTML (HyperText Markup Language) dili kullanılarak oluşturulur. HTML bir programlama dili değildir ve sayfanın yapısını ve içeriğini tanımlamak için kullanılır. Tarayıcıya web sayfasının içeriği ve yapısı hakkında bilgi verir ve öğelerin tanımlanmasını ve nasıl görüntüleneceğini sağlar.

HTML'nin yanı sıra genellikle başka teknolojiler de kullanılır:

- CSS (Cascading Style Sheets) – stil ve tasarım için,
- JavaScript – etkileşim ve dinamiklik için.
- Web sayfaları şu şekilde olabilir:
- Statik – içerik sabittir ve sık sık değişmez.
- Dinamik – içerik, kullanıcıya veya verilere bağlı olarak değişir (örneğin sosyal ağlar veya web mağazaları).

5.1.1. WEB SAYFASININ TEMEL YAPISI

Her web sayfası, doğru şekilde görüntülenmesini ve çalışmasını sağlayan mantıksal ve teknik bir yapıya sahiptir. Temel teknik yapı HTML ile tanımlanır ve birkaç ana öğeden oluşur ve aşağıdaki şekilde oluşturulur:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Sayfanın başlığı</title>
</head>
<body>
<h1> Ana başlık</h1>
<p> Bu bir metin örneğidir.</p>
</body>
</html>
```

Bu kod, bir web sayfasının iskeletini temsil eder. Öğelerin her birinin aşağıdaki işlevleri vardır:

1. <!DOCTYPE html>

Bu ifade, tarayıcıya belgenin HTML5'in en yeni sürümünde yazılmış bir web sayfası olduğunu bildiren bir bildiridir. Sayfanın doğru görüntülenmesi için önemlidir.

2. <html>

Bu etiket, HTML belgesinin başlangıcını belirtir. İki <html> etiketi arasında bulunan her şey web sayfasının bir parçasıdır.

3. <head>

Bu bölümde sayfayla ilgili meta bilgiler yer alır. Örneğin:

- sayfanın başlığı (<title>),
- karakter kümesi bilgileri,
- CSS dosyalarıyla bağlantı.

Bu bölüm doğrudan kullanıcıya gösterilmez.

4. <meta charset="UTF-8">

Metinlerin, özellikle Kiril alfabesinin, doğru görüntülenmesini sağlamak için kullanılır. Tarayıcıya Unicode karakter tablosunun ve UTF-8 kodlama biçiminin kullanılması gerektiğini bildirir.

4. <title>

Tarayıcının sekmesinde görüntülenen sayfanın başlığını tanımlar.

5. <body>

Sayfanın en önemli bölümüdür. Kullanıcının gördüğü her şey burada bulunur:

- metin,
- görüntüler,
- videolar,
- bağlantılar.

Teknik yapının yanı sıra web sayfasının, içeriğin daha iyi düzenlenmesini sağlayan mantıksal bir yapısı da vardır. Bu yapı genellikle şunları içerir:

1. Header (başlık bölümü) – Web sayfasının üst kısmıdır. Genellikle logo, başlık ve gezinme menüsü içerir;

2. Navigation (navigasyon) – Kullanıcıların sayfalar arasında gezinmesini sağlayan menüdür. Farklı bölümlere veya sayfalara yönlendiren bağlantılardan oluşur;

3. Main content (ana içerik) – Sayfanın temel bilgilerinin gösterildiği merkezi bölümdür. Metin, görüntü, video ve diğer öğeleri içerebilir;

4. Sidebar (yan çubuk) – Sayfanın yanında bulunan ek bölümdür. Reklamlar, ek bilgiler ve bağlantılar içerebilir.

5. Footer (alt bilgi) – Sayfanın alt kısmıdır. Genellikle iletişim bilgileri, telif hakkı bilgileri ve ek bağlantılar içerir.

İyi bir web sayfası yapısının önemi:

- Daha iyi okunabilirlik – Kullanıcılar içeriği daha kolay anlar,
- Daha iyi kullanıcı deneyimi – Basit gezinme ve düzen sağlar,
- Arama motoru optimizasyonu (Search Engine Optimization) – Arama motorları sayfayı daha kolay analiz eder,
- Daha kolay bakım – Geliştiriciler sayfayı daha kolay düzenleyebilir ve geliştirebilir.

HTML etiketlerinin doğru şekilde iç içe yerleştirilmesi, web geliştirmenin temel kurallarından biridir. En basit şekilde, etiketler "kutuların içinde kutular" prensibine göre çalışır. Son açılan etiket, ilk kapatılmalıdır. Buna LIFO (Last In, First Out – Son Giren İlk Çıkar) sistemi denir. Web tarayıcıları (Chrome, Edge, Firefox gibi) HTML kodunu doğrudan görüntülemez. Öncelikle HTML kodunu DOM (Document Object Model – Belge Nesne Modeli) adı verilen bir yapıya dönüştürürler. DOM, ebeveyn ve çocuk öğelerden oluşan bir aile ağacına benzer ve bu yapı öğeler arasındaki hiyerarşiyi gösterir.

- Doğru örnekte: <html> ebeveyn, <body> ise onun çocuğudur. <body> kutusu tamamen <html> kutusunun içinde yer alır.

```
<!DOCTYPE html>
        <html>
                <head>
                        <title>Doğru yapı</title>
                </head>
                <body>
                        <h1>Merhaba!</h1>
                        <p> Bu<strong> doğru</strong> iç içe yerleştirilmiş bir metindir.</p>
                </body>
        </html>
```

Yanlış örnekte (<html><body>...</html></body>): İçteki kutu (<body>) hâlâ açıkken dıştaki kutuyu (<html>) kapatmaya çalışıyoruz.

```
<!DOCTYPE html>
        <html>
        <body>
        <head>
        <title>Yanlış yapı</title> </head>
        <h1>Yanlış!</h1>
        <p>Bu <strong>yanlış </p> iç içe yerleştirilmiş bir metindir.</strong> </body>
</html>
```

Kod kötü şekilde iç içe yerleştirilmişse, web sayfası farklı tarayıcılarda veya cihazlarda tamamen farklı görünebilir. Ayrıca öğeler üst üste binebilir, kendi konumlarından "kayabilir" veya metin okunamaz hâle gelebilir.

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Web sayfası, internetin temel unsurlarından biridir ve modern iletişim ve bilgi paylaşımında önemli bir rol oynar. Web sayfası kavramını ve yapısını anlamak, web teknolojileri alanında daha ileri düzeyde öğrenme için temel oluşturur. Web sayfasının temel teknik yapısı <html>, <head> ve <body> gibi HTML öğelerinden oluşurken, mantıksal yapısı başlık, navigasyon, ana içerik ve alt bilgi gibi bölümleri içerir. Bu bileşenler bir araya gelerek web sayfasının işlevsel, düzenli ve kullanıcılar için yararlı olmasını sağlar.

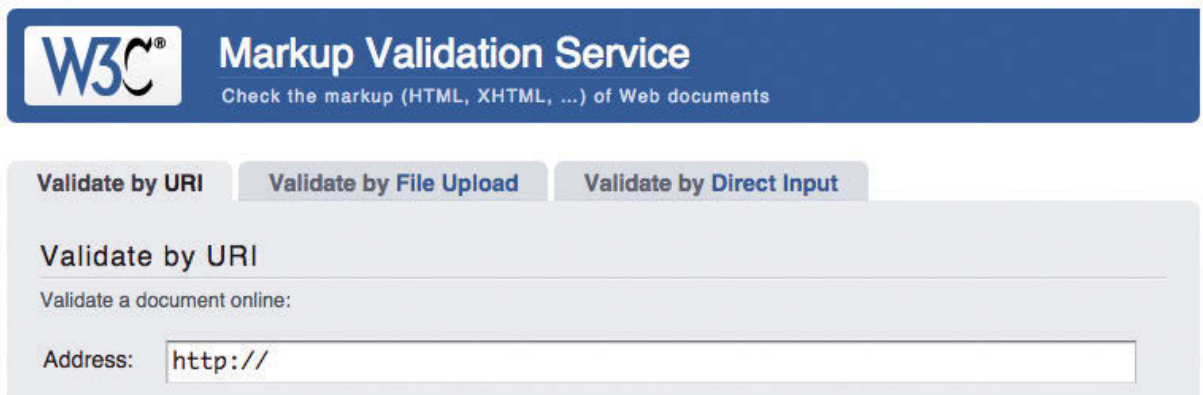
BİLGİ KONTROL SORULARI

1. Tarayıcıya belgenin HTML5 ile yazılmış bir web sayfası olduğunu bildiren başlangıç etiketi hangisidir?
2. Her doğru HTML belgesinin içermesi gereken üç temel etiketi sıralayın.
3. Sayfanın başlığı veya karakter kodlaması gibi meta bilgiler <head> bölümüne mi yoksa <body> bölümüne mi yerleştirilir?
4. <head> etiketi içinde bulunan içerik ile <body> etiketi içinde bulunan içerik arasındaki farkı açıklayın. Bunlardan hangisini web sayfasının ziyaretçisi görür?
5. HTML etiketlerinin doğru şekilde iç içe yerleştirilmesi neden önemlidir? Örneğin, neden <html><body>...</body></html> doğru, <html><body>...</html></body> ise yanlıştır?
6. Kendi kelimelerinizle <title> etiketinin görevini açıklayın. İçeriği web tarayıcısında nerede görüntülenir?
7. Zorunlu tüm yapısal öğeleri içeren boş bir HTML belgesinin tam kodunu yazın. Tarayıcının sekmesinde sayfanın başlığı "İlk Web Sayfam" olmalıdır.
8. Aşağıdaki kodda yapısal bir hata bulunmaktadır. Hatanın ne olduğunu bulun ve doğru kodu yazın.

```
<!DOCTYPE html>
<html>
 <body>
  <head>
   <title>Hoş Geldiniz</title>
  </head>
  <h1> Merhaba Dünya!</h1>
 </body>
</html>
```

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

W3C Markup Validation Service aracını araştır. İşlevlerini incele.



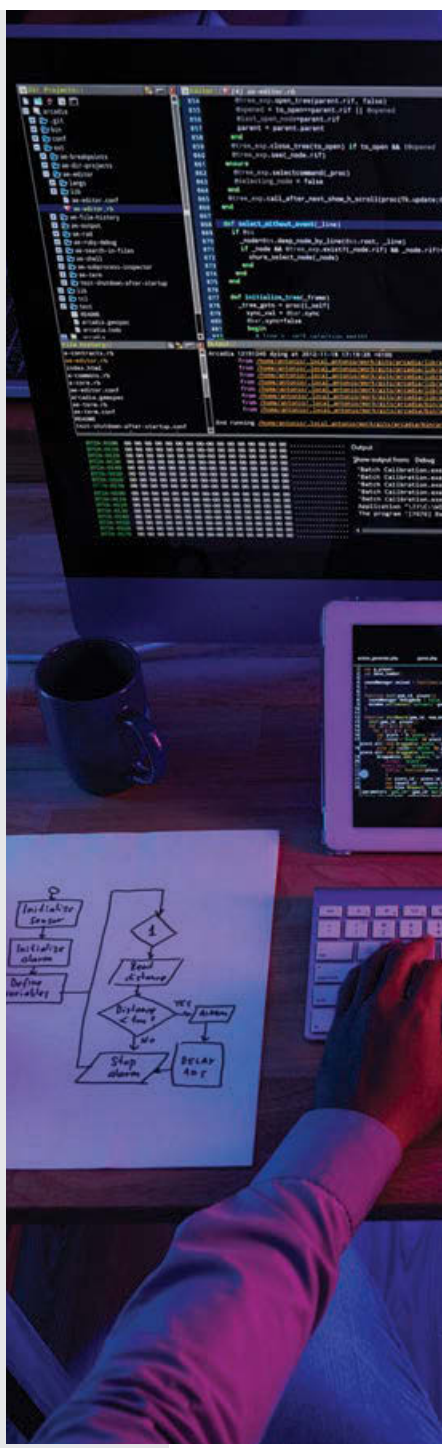
5.2

HTML'IN TEMELLERI

5.2.1. TEMELLERİ HTML ETİKETLERİ

Bir web sayfasının nasıl oluşturulduğunu daha iyi anlamak için HTML kodunu bir evin mimari planı gibi düşünebilirsin. Kod, tarayıcıya pencerenin, kapının ve çatının nereye yerleştirileceğini söyleyen etiketlerden oluşur.

Aşağıda temel HTML etiketlerini pratik örneklerle inceleyeceğiz. Böylece bu etiketleri nasıl kullanacağını öğreneceksin.



1. Başlık Etiketleri (<h1> - <h6>)

Başlıklar, içeriği düzenlemek ve önem derecesini belirlemek için kullanılır. <h1> en önemli ve en büyük başlıktır (kitabın başlığı gibi), <h6> ise en küçük başlıktır (alt başlıkların alt başlıkları için).

- **Uygulamalı Örnek:** Yemek tarifleri için bir web sayfası hazırladığını düşün. Tarifin adı <h1> etiketiyle, "Malzemeler" veya "Hazırlanışı" gibi alt başlıklar ise <h2> etiketiyle yazılır.

  <h1>Krep Tarifi</h1>

  <h2> Gerekli Malzemeler</h2>

2. Metin ve Başlık Etiketleri

- **Paragraf ( )** – Normal metinleri veya cümleleri paragraf olarak göstermek için kullanılır. Yeni bir etiketi açıldığında tarayıcı, metnin daha okunaklı olması için paragrafın öncesinde ve sonrasında otomatik olarak biraz boşluk bırakır.

Örneğin, aşağıdaki metin ayrı bir paragraf olarak görüntülenir.

Krepler her yaştan insanın sevdiği bir tatlıdır. Hızlı ve kolay hazırlanır.

- **Satır sonu
 –** Metni zorunlu olarak yeni bir satıra geçirmek için kullanılır. Bu etiket, kapanış etiketi olmayan az sayıdaki etiketlerden biridir, tek başına yazılır.
- ... etiketi – Metni kalın (Bold) yapar ve önemli kelimeleri vurgulamak için kullanılır.
- ... etiketi – Metni eğik (Italic) yapar.

3. Bağlantılar için Etiket (<a>)

<a> (anchor) etiketi, bir web sayfasını internetteki başka bir web sayfasına veya başka bir belgeye bağlamak için kullanılır. Tarayıcının bağlantının nereye yönlendirdiğini bilmesi için href özniteliği kullanılır. Örneğin, ziyaretçileri krep tarifleri hakkında daha fazla fikir aramaları için Google'a yönlendirmek istediğinde aşağıdaki kodu yazarsın:

```
<a href="https://www.google.com"> Google'da fikir arayın</a>
```

4. Görüntüler için Etiket (<img>)

Görsel öğeler olmadan web sayfaları ilgi çekici olmazdı. <img> etiketi, sayfaya bir görüntü eklemek için kullanılır. Bu etiketin kapanış etiketi olmadığını (kendi kendine kapandığını) ve src (görüntünün yolu) ile alt (görüntü yüklenmezse görüntülenen alternatif metin) özniteliklerini kullandığını bilmek önemlidir. Hazırlanmış lezzetli kreplerin görüntüsünü göstermek için aşağıdaki kod kullanılır:

```
<img src="krepler.jpg" alt="Çikolatalı kreplerin bulunduğu bir tabak">
```

Tüm bu etiketleri ana gövdenin (<body>) içinde mantıksal bir bütün hâlinde uyguladığında, aşağıdaki basit ama eksiksiz HTML kodunu elde edersin:

```
<!DOCTYPE html>
<html>
<head>
        <title>İlk Tarifim</title>
</head>
<body>
        <h1>Hızlı Krep Tarifi</h1>
        <p>Bu, sadece 15 dakikada hazırlayabileceğiniz en lezzetli ve en kolay krep tarifidir.
        </p>
        <img src=" krepler.jpg" alt=" Lezzetli krepler">
        <p> Beğendiyseniz, daha fazla fikir için partner sitemizi ziyaret edin:
        </p>
        <a href="https://www.google.com"> Daha fazla tarif burada</a>
</body>
</html>
```

5. Noktalı veya numaralı liste oluşturma etiketleri:

- — Noktalı, işaretsiz liste,
- — 1, 2, 3... şeklinde numaralandırılmış liste,
- — Liste öğesi. Her zaman <ul> veya <ol> etiketinin içinde kullanılır.

Bu etiketler, web sayfanda gösterilecek en sevdiğin derslerin bir listesini oluşturmak için kullanılabilir.

```
<h1>Okuluma hoş geldiniz</h1>
<p>Bu benim <strong> en sevdiğim </strong> derslerim:</p>
        <ul>
        <li>Matematik</li>
        <li>Web Tasarımı</li>
         </ul>
```

6. Blok tanımlama etiketi (division kelimesinin kısaltması – bölüm, kısım).

HTML'de anlamsal olmayan bir blok öğesidir ve diğer öğeleri gruplandırmak, daha kolay düzenlemek, yapılandırmak ve CSS kullanarak biçimlendirmek veya değiştirmek amacıyla kullanılan evrensel bir boş kapsayıcı (kutu) görevi görür:

```
<div class=" profil-kartı">
        <img src="avatar.jpg" alt="Profil resmi ">
        <h2>Marko Markovski</h2>
        <p>Meslek: Web tasarımcısı</p>
</div>
```

BİLİYOR MUSUNUZ?

Metinde "boşluk" bırakma ihtiyacını biliyor musun? Tarayıcı, bırakılan birden fazla boşluğu yalnızca tek bir boşluk olarak algılar. Tüm boşlukları tek bir boşlukta birleştirir. Teknik olarak art arda birden fazla boşluk bırakmak istiyorsan,   (non-breaking space – bölünemez boşluk) özel karakterini kullanmalısın.

SONUÇ:

Hangi durumlarda boşluk bırakma ihtiyacı ortaya çıkar ve birden fazla boşluk bırakmak için   etiketi nasıl kullanılır.

BİLGİ KONTROL SORULARI:

1. Bir web sayfasının mantıksal yapısı (kullanıcının gördüğü) ile teknik yapısı (tarayıcının kodda gördüğü) arasındaki farkı açıklayın.
2. HTML belgesinde <head> etiketinin görevi nedir, <body> etiketinin görevi nedir? Görüntülenmesi gereken bir metin <body> yerine <head> içine yazılırsa ne olur?
3. Bir restoran için çevrim içi menü oluşturmanız gerekiyor. Seçtiğiniz üç yemekten oluşan numaralı bir liste (<ol>) oluşturacak HTML kodunu yazın.
4. Aşağıdaki kod bölümünü inceleyin:

```
<p>Bloguma hoş geldiniz<strong> programlama blogu.</p></strong>
```

Bu kodda hatayı bulun, iç içe yerleştirme (nesting) kurallarına göre neyin yanlış olduğunu açıklayın ve doğru sözdizimini yazın.

5. Tarayıcının şu kodda yazılan metni nasıl görüntüleyeceğini analiz edin:

```
<h1>öğreniyoruzHTML Birlikte!</h1>
```

Bu örnekte, metin için kullanılan HTML etiketlerinin hangi özelliği ortaya çıkmaktadır?

6. Basit bir HTML belgesini inceliyorsunuz. Programcı içeriği şu şekilde düzenlemiştir:

En üstte, makalenin ana ve en önemli başlığını <h3> etiketiyle belirtmiştir.

Biraz aşağıda, daha küçük bir alt başlığı <h1> etiketiyle belirtmiştir.

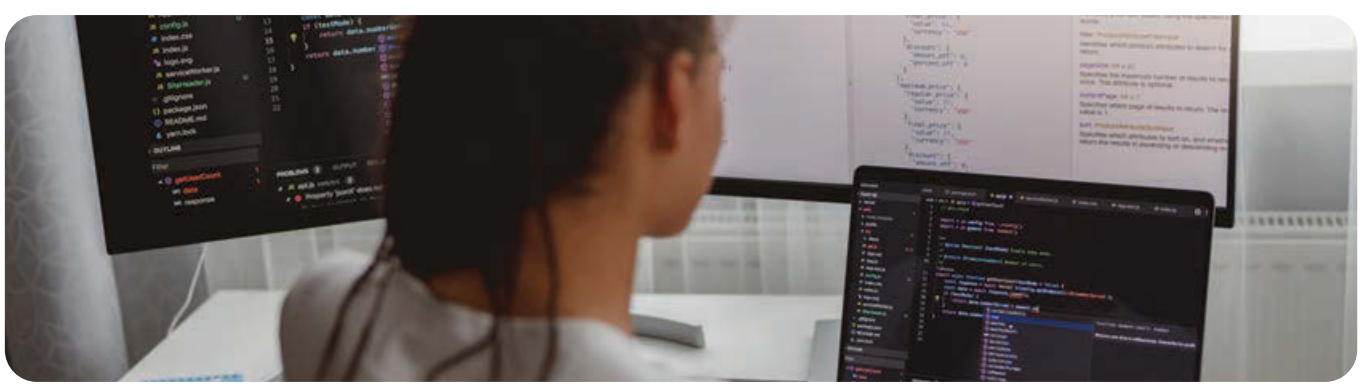
Son olarak, uzun ve normal bir metin paragrafı için, her cümleye ayrı ayrı olmak üzere beş adet <h1> etiketi kullanmıştır. Bunu, metnin çok önemli ve dikkat çekici olmasını istediği gerekçesiyle yapmıştır.

Bu temel HTML etiketlerinin kullanımını değerlendirin. Bu yapının mantıksal hiyerarşide ne gibi sorunlar oluşturduğunu ve sayfanın ana konusunu anlamaya çalışan Google gibi arama motorlarını nasıl etkileyebileceğini açıklayın. Kodda nelerin değiştirilmesi gerektiğini önerin.

7. Kodu değerlendirin ve hangi teknik ve mantıksal hataların yapıldığını belirleyin:

```
<div>
<p>En ünlü eserlerimize göz atın:</p>
<ul> <a href="https://muzej.mk/monaliza">
<li> Mona Lisa <img src="monaliza.jpg" alt=" Mona Lisa'nın resmi "></li>
</a>
<li>Davut Heykeli</li>
 <img src="david.jpg" alt="Davut'un resmi">
 <li>Yıldızlı Gece</li>
 </ul>
</div>
```

Müzenin web sayfasındaki listenin doğru şekilde görüntülenmesi için bir çözüm önerin.



SONUÇ:

Google'ın Web Crawler robotları başlık hiyerarşisini (<h1> - <h6>) nasıl tarar?

ÖĞRENDİKLERİNİZİ HATIRLAYIN

Her HTML belgesi iki bölümden oluşur: başlık (head) ve gövde (body). Belgenin başlık bölümünde, web sayfasının içeriği olarak görüntülenmeyen sayfaya ait bilgiler yazılır. İçerik ise gövde bölümüne yazılır. İlk <html> etiketi tarayıcıya bunun bir HTML belgesinin başlangıcı olduğunu bildirir, son </html> etiketi ise belgenin sona erdiğini belirtir. Başlık bölümü <head> ve </head> etiketleriyle, gövde bölümü ise <body> ve </body> etiketleriyle çevrelenir. <!DOCTYPE> bildirimi, tarayıcının web sayfasını doğru şekilde görüntülemesi için kullanılır. Temel kodda, karakter kodlama biçimini tanımlayan etiketin de bulunması gerekir. Bu etiket başlık bölümüne eklenir: <meta charset="UTF-8">. Metin düzenlemek için kullanılan temel etiketler aşağıdaki tabloda verilmiştir:

Etiket (Tag)	Adı / Anlamı	Temel İşlevi (Görevi)
<!DOCTYPE html>	Belge bildirimi	Tarayıcıya en yeni HTML5 sürümünün kullanıldığını bildirir.
<html>	Kök öğe	Tüm kodu kapsar ve HTML belgesinin başlangıcını ve sonunu belirtir.
<head>	Belge başlığı	Gizli bilgileri ve meta verileri içerir (örneğin <meta charset="UTF-8"> ve <title>).
<body>	Belge gövdesi	Sayfanın kullanıcının gördüğü tüm görünür bölümünü (metin, resimler, bağlantılar) içerir.
<h1>ile <h6>	Başlıklar	Başlıkları kesin bir hiyerarşiyle tanımlar (en önemli <h1>'den en küçük <h6>'ya kadar).
** **	Paragraf (Paragraph)	Normal blok metnini düzenlemek ve görüntülemek için kullanılır.
<a>	Köprü (Bağlantı)	href özniteliği (bağlantının hedefi) aracılığıyla farklı web sayfalarını birbirine bağlar.
<img>	Resim	src kaynağı aracılığıyla sayfaya bir resim ekler ve zorunlu alt açıklamasını içerir.
** / **	Sırasız / Sıralı liste	Noktalı listeler ( ) veya numaralı listeler ( ) oluşturur ve her öğe için kullanır.
** **	Bölüm / Kutu (Division)	Öğeleri gruplandırmak ve daha sonra biçimlendirmek için kullanılan evrensel, boş bir kapsayıcıdır.

BİLGİ KONTROL SORULARI:

1. Aşağıdaki kod satırlarından hangisi HTML etiketlerinin DOĞRU şekilde iç içe yerleştirilmesini (nesting) göstermektedir?

- A) Modern HTML kodu öğreniyoruz.
- B) Modern HTML kodu öğreniyoruz.
- C) Modern HTML kodu öğreniyoruz.
- D) Modern HTML kodu öğreniyoruz.

2. VS Code'da aşağıdaki metni yazarsanız tarayıcı ekranda ne görüntüler: Merhaba öğrenciler! (5 boşlukla)?

- A) “Merhaba öğrenciler!” (tam olarak 5 boşluk, çünkü HTML tüm boşlukları korur)
- B) “Merhabaöğrenciler!“ (kelimeleri tamamen birleştirir, çünkü boşluk silinir)
- C) “Merhaba öğrenciler!“ (yalnızca bir boşluk gösterilir, bunun nedeni White-space Collapse özelliğidir)
- D) Tarayıcı bir sözdizimi hatası bildirecek ve metni hiç göstermeyecektir.

3. HTML’de etiketinin temel görevi nedir?

- A) Metni ekranda otomatik olarak kalınlaştırmak ve büyütmek için kullanılır.
- B) Diğer öğeleri gruplandırmak için evrensel, boş bir kapsayıcı (kutu) görevi görür.
- C) Yalnızca web sayfasına resim ve bağlantı eklemek için kullanılır.
- D) Sayfanın içeriğini otomatik olarak İngilizceye çevirir.

4. Aşağıdaki etiketlerden hangisi kapanış etiketi olmayan ve kendi kendine kapanan “Boş Öğeler“ (Void Elements) grubuna girer?

- A) <p>
- B) <h1>
- C) <img>
- D) <a>

5. VS Code dosyaları zaten bu formatta kaydetmesine rağmen, kodun <head> bölümüne <meta charset=”UTF-8”> satırını yazmak neden gereklidir?

- A) Tarayıcıya (Chrome, Safari...) karakterler için hangi standardın kullanılacağını belirtmek ve Kiril alfabesinin bozuk karakterler olmadan görüntülenmesini sağlamak için.
- B) VS Code programına Makedonca kod yazma izniniz olduğunu bildirmek için.
- C) Kodun daha hızlı yazılmasını sağlayan otomatik kısaltmaları (Emmet) etkinleştirmek için.
- D) Tarayıcının en üst sekmesinde görünen ana başlığı tanımlamak için.
- 6. Okulunuzun logosunun resmini bir web sayfasına eklemek istediğinizi düşünün. Resmi eklemek için gereken tam HTML satırını yazın ve alt özniteliğini belirtmenin neden önemli olduğunu kısaca açıklayın (en az bir neden belirtin).
-

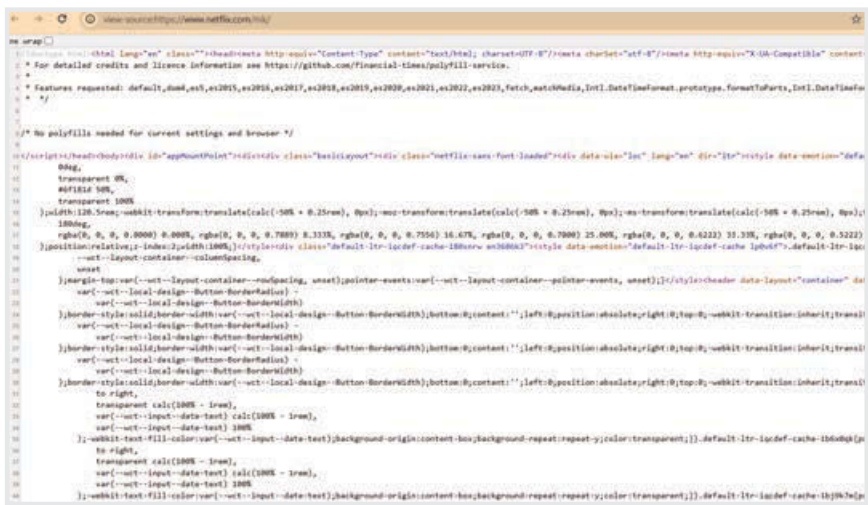
DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Modern HTML yalnızca metin ve resimleri görüntülemek için değildir. Geçmişte karmaşık programlar gerektiren, günümüzde ise birkaç satır kodla etkinleştirilebilen gelişmiş teknolojiler (API’ler) ile birlikte gelir:

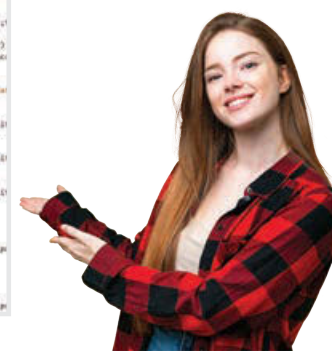
1. Coğrafi Konum (Geolocation): HTML ve biraz JavaScript kullanarak kullanıcıdan izin isteyebilir ve web sitesinin harita üzerinde tam olarak nerede bulunduğunu öğrenmesini sağlayabilirsin.
2. **Yerel Depolama (Local Storage):** HTML5, web sitenin verileri doğrudan kullanıcının tarayıcısında saklamasına olanak tanır (örneğin “Karanlık Mod” anahtarı veya alışveriş sepetindeki ürünler). Kullanıcı bilgisayarını kapatıp ertesi gün geri dönse bile site onun seçimini hatırlayabilir.
3. **Ses ve video akışı: <audio> ve <v**ideo> etiketleri, doğrudan tarayıcı içinde YouTube veya Spotify gibi kendi oynatıcını oluşturmanı sağlayan gelişmiş kontrollere sahiptir, herhangi bir harici eklentiye ihtiyaç duyulmaz.

5.2.2 HTML KODU YAZMAK İÇİN KULLANILAN EDİTÖRLER

Web sayfası, HTML kodu içeren ve .html uzantısına sahip bir belgedir. En sevdiğin web sayfasını aç, herhangi bir yere sağ tıkla ve menüden "View page source" seçeneğini seç. Sayfanın HTML kodu, Notepad veya Notepad++ gibi bir metin düzenleyicide yazılabilecek bir metin belgesi şeklinde görüntülenir. Kodu Microsoft Word, Apple Pages gibi kelime işlemcilerde yazmak önerilmez, çünkü bu programlar tarayıcının okuyamayacağı dosyalar oluşturabilir. HTML kodu, Visual Studio Code (VS Code) gibi editörlerde de yazılabilir. VS Code ücretsizdir ve kurulması gerekir. Ayrıca CodePen.io, OneCompiler gibi çevrim içi editörler de kullanılabilir. Bu ders kitabında HTML öğrenmek için Visual Studio Code (VS Code) editörü kullanılacaktır. VS Code açık kaynaklıdır ve web geliştirmede endüstri standardı olarak kabul edilir. Kod yazarken akıllı öneriler sunar, kullanılabilir dosyaları otomatik olarak gösterir ve web sayfasını tarayıcıda görüntülemeye olanak tanır.



Resim 1: HTML kodunun görünümü



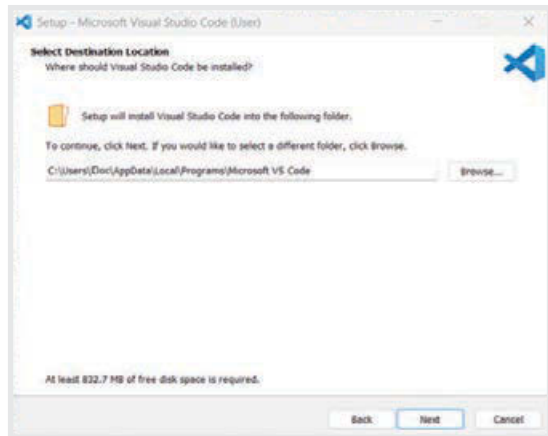
5.2.3. HTML EDİTÖRÜ VISUAL STUDIO CODE'UN KURULUMU

Visual Studio Code (VS Code) kurulumu oldukça hızlı ve basit bir işlemdir. Visual Studio Code'un resmi web sitesini aç ve işletim sistemine uygun sürümü seç. Gerekli sürüme tıkla ve kurulum dosyasının (.exe uzantılı) bilgisayarına indirilmesi için birkaç saniye bekle.

1. İndirilen dosyayı (örneğin, VSCodeUserSetup-x64.exe) bilgisayarında bul ve kurulum işlemini başlatmak için dosyaya çift tıkla (double-click).
2. Kurulum sürecinde sana yol gösteren bir kurulum sihirbazı açılır. Öncelikle Lisans Sözleşmesi (License Agreement) penceresi görüntülenir. Sözleşmeyi oku, "I accept the agreement" (Sözleşmeyi kabul ediyorum) seçeneğini işaretle ve ardından Next (İleri) düğmesine tıkla.

Resim 2: Sözleşme koşullarının kabul edilmesi

3. Daha sonra dosyayı bilgisayarında kaydedeceğin konumu seç.

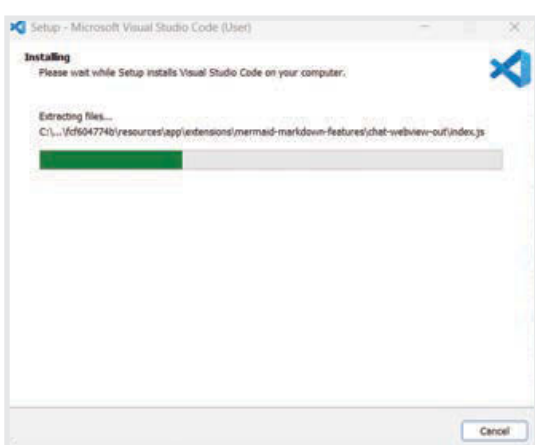


Resim 3: Kurulum konumunun belirlenmesi

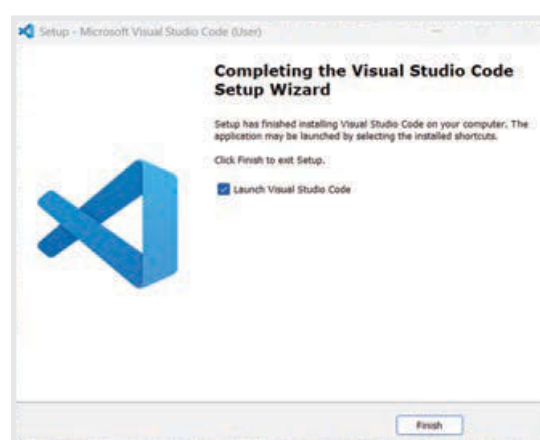
4. "Select Additional Tasks" başlıklı pencere açıldığında aşağıdaki seçenekleri belirleme olanağın vardır:

- Create a desktop icon: VS Code'u hızlı bir şekilde açabilmek için masaüstünde VS Code kısayolu (simge) oluşturur.
- Add "Open with Code" to Windows Explorer... (Bu şekilde iki seçenek vardır, ikisini de seç.) Bu kullanışlı bir seçenektir, daha sonra HTML projelerinin bulunduğu bir klasöre sağ tıklayarak "Open with Code" seçeneğini seçebilir ve tüm projeyi doğrudan editörde açabilirsin.
- Register Code as an editor for supported file types: Herhangi bir .html veya .css dosyasına çift tıklandığında dosyanın otomatik olarak VS Code'da açılmasını sağlar.
- Add to PATH: Bu seçenek varsayılan olarak işaretlidir ve programın sistemle doğru şekilde çalışmasını sağlar.

Bu seçenekleri belirledikten sonra kurulum işlemini başlatmak için Next (İleri) düğmesine tıkla.

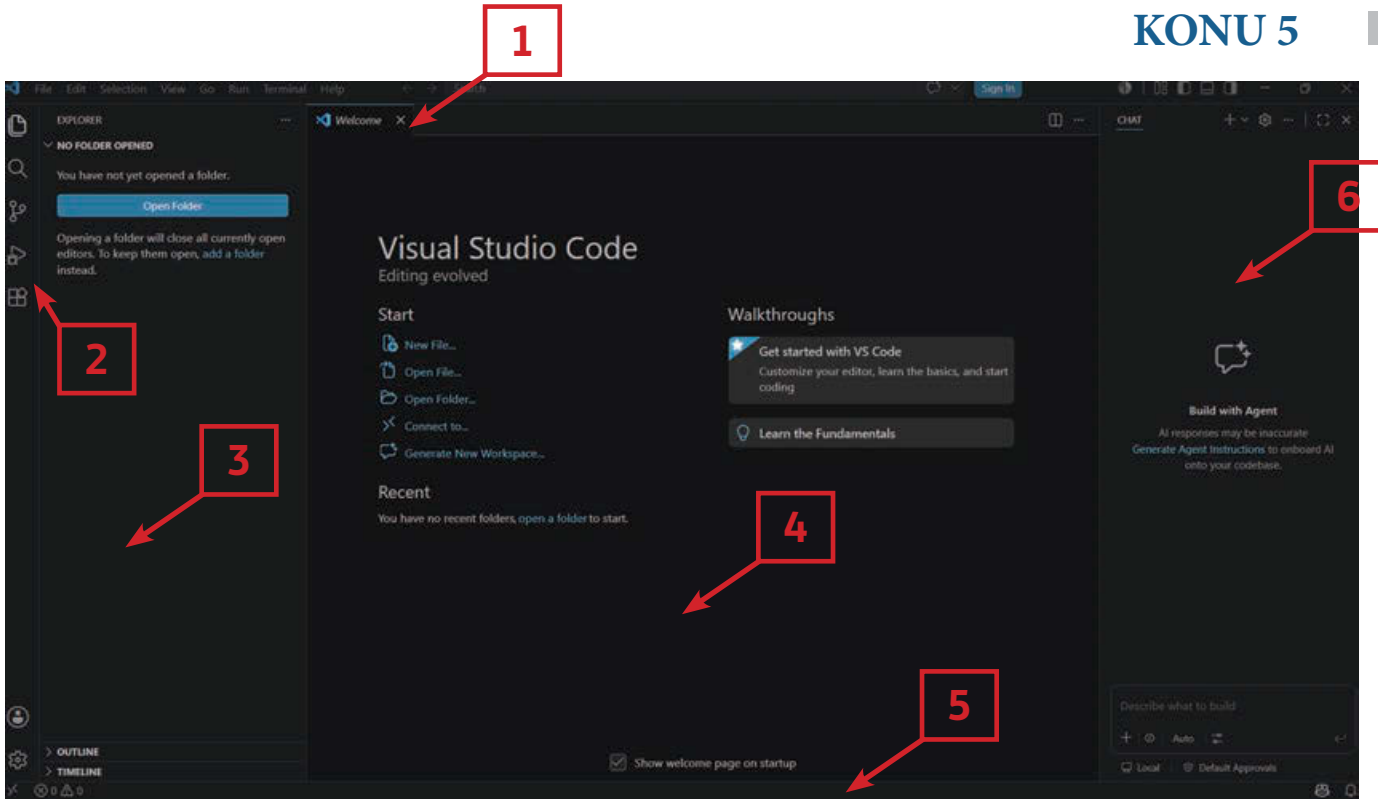


Resim 4: Kurulum işlemi



Resim 5: VS Code kurulumunun tamamlanması

5. Kurulum işlemi, yeşil çubuk tamamen dolana kadar yaklaşık 10–20 saniye sürer. Finish (Son) düğmesine tıklandığında VS Code editörü açılır ve başlangıç çalışma penceresi görüntülenir.



Resim 6: VS Code'un Başlangıç Çalışma Penceresi

1. Başlangıç Sekmesi: "Welcome" (Hoş Geldiniz)

Ekranın tam ortasında Welcome adlı büyük bir sekme açılır. Burası başlangıç noktandır ve şu kısayollar bulunur:

- **New File** – Yeni, boş bir belge oluşturmak için.
- **Open Folder** – Bilgisayardaki bir klasörü açmak için (geliştiriciler için en önemli seçenek).
- **Clone Git Repository** – İnternetten kod indiren ileri düzey kullanıcılar için.

2. En Sol Kenardaki Araç Çubuğu (Activity Bar)

Pencerenin en solunda bulunan dar, dikey simge çubuğudur. Bu simgeler temel "çalışma modlarını" temsil eder:

- **Explorer (İki klasör simgesi)**: Zamanın %95'inde bunu kullanacaksın. Buraya tıkladığında klasörlerini ve dosyalarını görebileceğin yan pencere açılır.
- **Search (Büyüteç)**: Tüm dosyalarda aynı anda kelime aramak ve değiştirmek için.
- **Source Control (Dal/düğüm):** Git ve kod sürümlerini yönetmek için.
- **Run and Debug (Üçgenli böcek):** Kodu test etmek ve hataları bulmak için.
- **Extensions (Biri dışarı doğru çıkan dört kare):** Ücretsiz eklentilerin "mağazasıdır". Buradan daha kolay kod yazmak için kullanışlı araçlar yükleyebiliriz.

3. Yan Pencere (Sidebar)

Bu alan, araç çubuğunun hemen sağında bulunur. Explorer açık olduğunda burada projenin yapısını görürsün: hangi HTML dosyalarının bulunduğu, resimler için klasörlerin olup olmadığı vb. Başlangıçta herhangi bir klasör açık değilse burada büyük, mavi "Open Folder" düğmesi bulunur.

4. Ana Düzenleme Alanı (Editor)

Ekranın en büyük ve merkezi bölümüdür. Bir dosyaya (örneğin index.html) tıkladığında içeriği burada açılır. Burası HTML kodunu yazacağın

ve değiştireceğin "çalışma alanındır". Aynı anda birden fazla dosya açabilir ve bunları Chrome'daki gibi üst kısımda sekmeler hâlinde düzenleyebilirsin.

5. Alt Durum Çubuğu (Status Bar)

Pencerenin en altında ince bir çubuk bulunur (genellikle mavi veya mor renktedir). Çalıştığın dosya hakkında anlık bilgiler verir: hangi satırda olduğun, hangi programlama dilinin etkin olduğu (sağ altta HTML yazmalıdır) ve karakter kodlaması (varsayılan olarak UTF-8). Pencere, dikkatini dağıtmayacak şekilde tasarlanmıştır, solda dosyaların organizasyonu, ortada ise kod bulunur.

6. Yapay zekâ asistanıyla çalışma alanı.

5.2.4. EDİTÖRLE ÇALIŞMA

VS Code'da çalışmaya başlama işlemi oldukça basit ve hızlıdır. Programı ilk kez açtığında hatırlaman gereken en önemli şey, HTML dosyalarının tek başına "havada duramayacağı" ve her zaman bilgisayarındaki kendi klasörlerinde (folder) düzenli bir şekilde saklanması gerektiğidir. Aşağıda adımlar tek tek açıklanmıştır:

Adım 1: Proje klasörünü oluştur

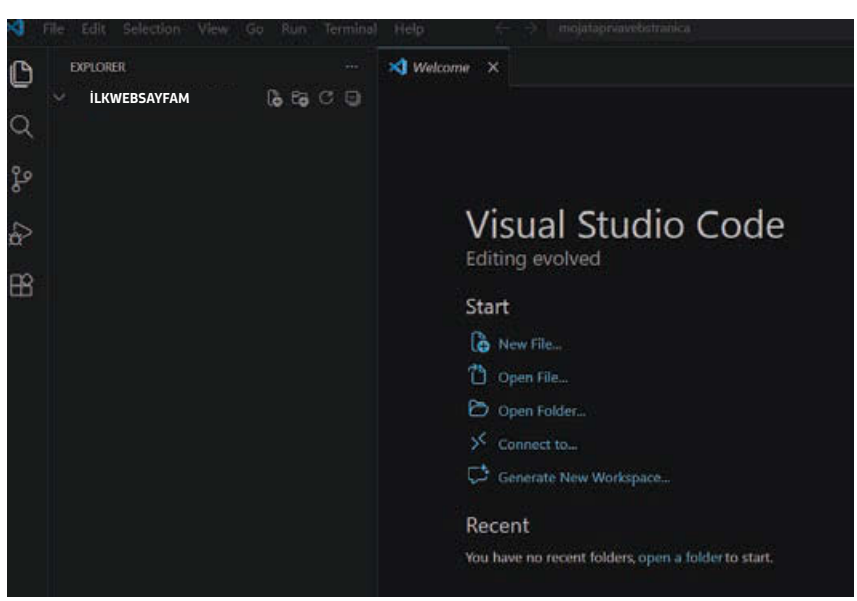
Visual Studio Code'u (VS Code) açmadan önce bilgisayarında yeni, boş bir klasör oluştur. Örneğin ilk-projem adını verebilirsin. Küçük Latin harfleri kullanman, boşluk bırakmaman ve web sayfasının içeriğini ifade eden bir isim seçmen önerilir.

Adım 2: Klasörü VS Code'da aç

Şimdi VS Code'u aç. Klasörü programa eklemenin iki yolu vardır:

1. File -> Open Folder... menüsünü kullan ve 1. adımda oluşturduğun klasörü seç.
2. Ya da klasörü farenin sürükle-bırak (Drag, Drop) yöntemiyle doğrudan VS Code'un boş alanına taşı.

Bu neden önemlidir? Bu şekilde VS Code, bundan sonra yapacağın tüm çalışmaların belirli bir projeye ait olduğunu bilir. Ekranın sol tarafı senin çalışma alanın olan Explorer bölümüne dönüşür.



Resim 7: Visual Studio Code'da oluşturulan klasör

Adım 3: Yeni bir HTML Dosyası Oluştur

Sol tarafta, klasörünün adının yazdığı bölümde New File (Yeni Dosya) simgesine tıkla (üzerinde artı işareti bulunan bir kâğıt simgesine benzer).

Dosyanın adını index.html olarak yaz ve Enter tuşuna bas.

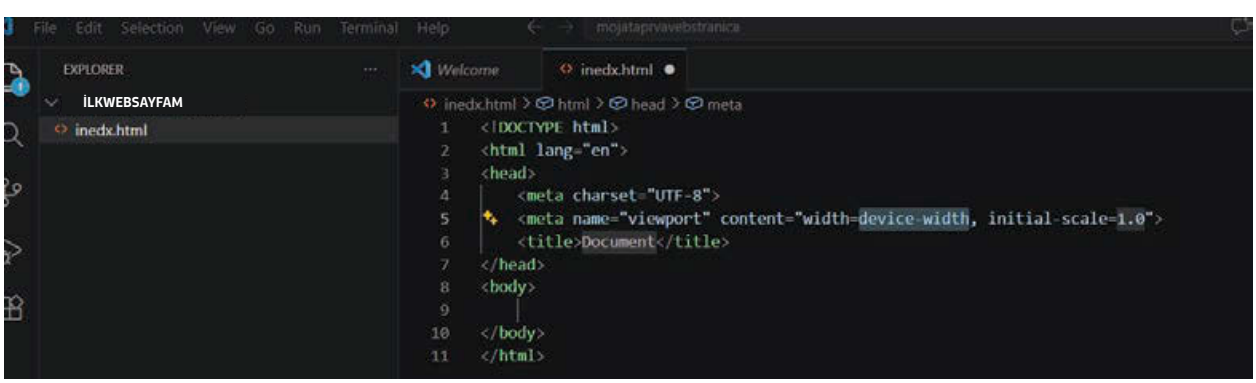
Resim 8: Visual Studio Code'da adlandırılmış dosya

Önemli: Dosya adı .html ile bitmelidir, böylece bilgisayar bunun sıradan bir metin belgesi değil, bir web sayfası olduğunu anlar. index adı, her web sitesinin ana (başlangıç) sayfası için dünya genelinde yaygın olarak kullanılan standart bir addır.

Adım 4: Başlangıç İskeletini Etkinleştir

Şimdi dosya açılmış ve boş durumdayken:

1. İlk satıra yalnızca bir ünlem işareti yaz: !
2. Küçük bir menü (Emmet kısayolu) görünecektir. Enter veya Tab tuşuna bas.
3. VS Code, gerekli başlangıç kodunun tamamını (<html>, <head>, <body> ve UTF-8 tanımlaması bulunan iskeleti) otomatik olarak oluşturacaktır.



Resim 9: Bir web sayfasının temel iskeleti oluşturuldu

Adım 5: İlk kodunu yaz

Tarayıcı, ekranda yalnızca başlangıç <body> ve kapanış </body> etiketleri arasında bulunan içeriği görüntüler.

İmleci ilk <body> etiketinin altına yerleştir ve şunları yaz:

```
<body>
  <h1>Merhaba Dünya!</h1>
  <p>Bu, VS Code'da yazdığım ilk web sayfasıdır.</p>
</body>
```

Adım 6: Kodu kaydet (Save)

Üstte, index.html yazan sekmede küçük beyaz bir daire göreceksin. Bu, "Yeni bir kod yazdın ve henüz kaydedilmedi!" anlamına gelir.

- Klavyeden Ctrl + S (Windows) veya **Cmd +** S (Mac) kısayolunu **kullan.**
- Beyaz daire kaybolacak ve X işaretine dönüşecektir. Artık kodun sabit diske güvenli bir şekilde kaydedilmiştir.

Böylece ilk kodun tamamlanmış ve görüntülenmeye hazırdır.

Web sayfasını görüntülemenin birkaç yolu vardır. Aşağıda üç yöntem açıklanacaktır.

Yöntem 1: Klasörden doğrudan açma. Bu, her bilgisayarda çalışan ve internet bağlantısı veya ek eklenti kurulumu gerektirmeyen en basit yöntemdir.

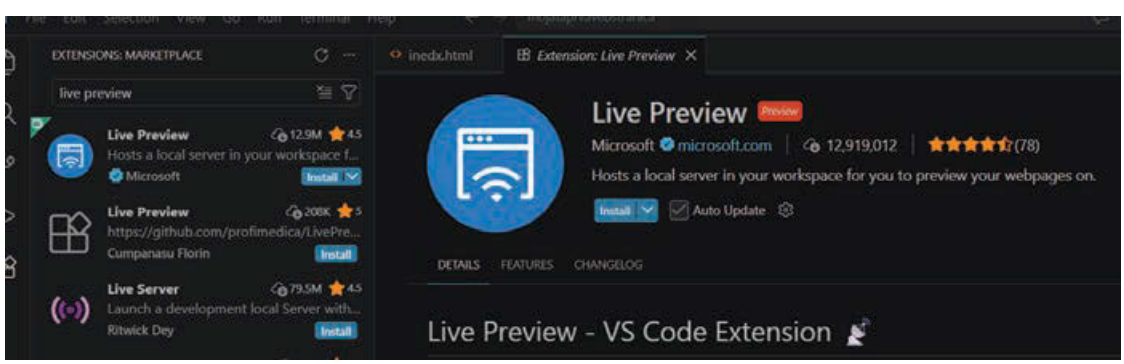
- index.html dosyasını kaydettiğin klasörü bilgisayarında aç.
- index.html dosyasına çift tıkla.
- Bilgisayar dosyayı otomatik olarak varsayılan tarayıcında (Chrome, Edge, Firefox) açacaktır.

Bu yöntem için önemli not: VS Code'daki kodda her değişiklik yaptığında, önce değişikliği Ctrl + S ile kaydetmen gerekir. Daha sonra tarayıcıyı manuel olarak açmalı ve değişiklikleri görmek için Refresh (Yenile) düğmesine tıklamalı veya klavyeden F5 tuşuna basmalısın.

Yöntem 2: "Live Preview" eklentisi ile (doğrudan VS Code içinde önizleme)

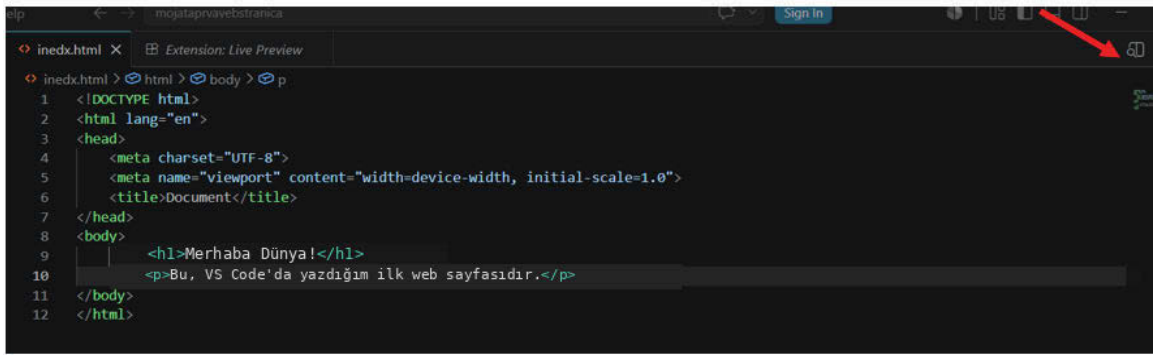
İki monitörün yoksa veya ekranı Chrome ile ikiye bölmek istemiyorsan, Microsoft doğrudan programın içinde küçük bir tarayıcı açan bir eklenti geliştirmiştir.

1. Extensi**ons bölümü**nü (soldaki dört kare simgesi) seç ve Live Preview e**klentisini a**ra (Microsoft tarafından geliştirilen resmî eklenti).
2. Install **(Yükl**e) düğmesine tıkla.



Resim 10: Live Preview kurulumu

3. Kurulum tamamlandığında Install düğmesi bir dişli simgesine dönüşür (bu, eklentinin kurulduğu anlamına gelir). Bu andan itibaren eklenti VS Code'a eklenmiş olur ve kullanıma hazırdır. Üst sağ köşedeki küçük ekran simgesine tek tıklayarak kullanabilirsin
4. index.html koduna geri döndüğünde, ekranın üst sağ köşesinde büyüteçli küçük bir bilgisayar ekranına benzeyen yeni bir simge göreceksin.



Resim 11: Live Preview yüklemek

5. Bu simgeye tıkla. VS Code ekranı iki bölüme ayrılacaktır: solda kodun, sağda ise web sayfanın canlı önizlemesi hemen görüntülenecektir. Yaptığın her kaydetme işlemi, değişiklikleri anında burada gösterir.

Resim 12: Çalışma alanı ve web sayfasının görüntülenmesi

Yöntem 3: Sağ taraftaki yapay zekâ asistanı aracılığıyla

Sağ tarafta zaten yapay zekâ penceresi bulunduğundan, hızlı bir önizleme için onun özelliklerinden de yararlanabilirsin.

1. Tüm HTML kodunu seç.

2. Sohbette yapay zekâ asistanına şu soruyu sor: "Bu kod oluşturulduğunda nasıl görünecek?" veya "Bu HTML kodunun önizlemesini göster."

3. Copilot veya Cursor gibi birçok modern yapay zekâ aracında, doğrudan kendi pencerelerinde bulunan "Preview" (Önizleme) düğmesi vardır. Bu düğmeye tıklayarak sayfanın tasarımının o anda nasıl göründüğünü görsel olarak görüntüleyebilirsin.

Başlangıç seviyesinde öğrenme için Yöntem 2 (Live Preview) en uygun yöntemdir, çünkü her şey tek bir yerde bulunur ve VS Code'un dışına çıkman gerekmez.

BİLGİ KONTROL SORULARI:

1. VS Code'da kodundaki değişiklikleri kaydetmek için kullanılan dünya standardı klavye kısayolu hangisidir?

- A) Ctrl + C
- B) Ctrl + S
- C) Ctrl + V
- D) Ctrl + Z

2. VS Code'un en soldaki dikey çubuğunda bulunan, bir tanesi ayrılmış dört kareden oluşan simge neyi ifade eder?

- A) Kod içinde kelime arama menüsü.
- B) Klasör ve dosyalarla çalışma menüsü (Explorer).
- C) Live Preview gibi eklentilerin kurulabileceği eklenti mağazası (Extensions).
- D) Yapay zekâ ile sohbet bölümünün açıldığı alan.

3. Görüntüleme eklentileri kullanmak yerine web sayfanı bilgisayarındaki klasörden çift tıklayarak doğrudan tarayıcıda açıyorsun. VS Code'da yeni HTML kodu ekledikten ve Ctrl + S ile kaydettikten sonra, değişikliği Chrome'da görmek için hangi zorunlu adımı uygulamalısın?

- A) Tarayıcıyı tamamen kapatıp klasörden tekrar çift tıklayarak açmalısın.
- B) Tarayıcıdaki Refresh (Yenile / Yeniden Yükle) düğmesine manuel olarak tıklamalı veya klavyeden F5 tuşuna basmalısın.
- C) Hiçbir şey yapmana gerek yok, Ctrl + S'ye bastığın anda sayfa otomatik olarak değişir.
- D) Eski HTML dosyasını klasörden silip yeni bir dosya oluşturmalısın.

4. HTML belgesinin başlangıç iskeletini elle yazmak yerine, VS Code'un (Emmet aracılığıyla) bu kodu Enter tuşuna bir kez basarak otomatik oluşturması için boş dosyaya hangi işareti (kısayolu) yazmalısın?

- A) Soru işareti (?)
- B) Hashtag (#)
- C) Ünlem işareti (!)
- D) Eğik çizgi (/)

5. Sağ taraftaki yerleşik yapay zekâ asistanıyla çalışırken, kodunun yalnızca belirli bir satırını açıklamasını veya düzeltmesini istemenin en etkili yolu hangisidir?

- A) Tüm kodu sohbet penceresine elle yeniden yazmalısın.
- B) Sol tarafta kodun ilgili satırını fareyle seçmen ve ardından sağdaki yapay zekâ penceresine sorunu yazman yeterlidir.
- C) Dosyayı özel bir .txt formatında kaydedip asistana göndermelisin.
- D) Yapay zekâ asistanları yazdığın kodu göremez, bu nedenle ona satırın numarasını (örneğin "14. satır") söylemelisin.

6. Boş bir VS Code'da yeni bir projeye başlamak ve programın hemen sana yardımcı olmaya başlaması (etiketleri renklendirmesi ve öneriler sunması) için izlenmesi gereken doğru adım sırası hangisidir?

- A) Önce kod yazılır, sonra yeni dosya oluşturulur ve en sonunda bir klasöre yerleştirilir.
- B) Önce masaüstüne boş bir metin dosyası kaydedilir, sonra klasör olarak yeniden adlandırılır ve Chrome açılır.
- C) Önce yeni bir dosya oluşturulur, sonra ünlem işareti yazılır ve en son bilgisayarda nereye kaydedileceği düşünülür.
- D) Önce proje klasörü açılır (Open Folder), ardından klasörün içinde yeni bir dosya oluşturulur (New File) ve dosya hemen .html uzantısıyla kaydedilir.
- E) Sıralamanın hiç önemi yoktur, VS Code bilgisayarı açmadan önce bile ne yazmak istediğini kendiliğinden bilir.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Eğer sık sık aynı yapıyı (örneğin bir kartın iskeletini, belirli sınıflara sahip bir tabloyu veya bir menüyü) yazıyorsan, bunu sürekli yeniden yazmana veya eski projelerden Copy/Paste yapmana gerek yoktur. Kendi Emmet kısayolu**nu oluşturabilirsin. Sol alttaki dişli** simgesine (Settings) tıkla ve User Snippets seçeneğini seç. Ardında**n html.json d**osyasını seçerek **kendi kı**sayolunu tanımlayabilirsin. Örneğin mojakarticka adlı bir kısayol tanımlarsan, bunu boş bir HTML dosyasına yazıp Tab tuşuna bastığında VS Code başlıklar, sınıflar ve resimler içeren önceden tanımladığın kodun tamamını otomatik olarak yükler. Ayrıca HTML dosyan 500 veya 1000 satıra ulaştığında, yukarı ve aşağı kaydırma yapmak oldukça zorlaşır. İleri düzey programcılar akıllı gezinme özelliklerini kullanır.

1. **Breadcrumbs (Ekmek kırıntıları): Ko**dun üst kısmında her zaman bir yol gösterilir (örneğin src > index.html > body > div.container > table). Bu yoldaki herhangi bir etikete tıklayarak kaydırma yapmadan doğrudan o bölüme geçebilirsin.
2. **Outline Menüsü: Sol a**lt bölümde (klasörlerin altında) Outline adlı bir menü bulunur. HTML belgesini bir aile ağacı şeklinde gösterir. Bu ağaçtaki herhangi bir etikete tıkladığında editör seni doğrudan o bölüme götürür.

5.2.5. HTML ETİKETLERİNİN KULLANIMI

HTML dilinde, etiketlere ek özellikler kazandırmak için öznitelikler (attributes) kullanılır. Etiketlerin kendisi web sayfasının iskeletiyse (başlığın nerede, paragrafın nerede olduğunu belirtir), öznitelikler bu etiketlere ek bilgiler verir. Öznitelikler her zaman başlangıç etiketinin içinde yazılır (asla kapanış etiketinin içinde yazılmaz). Şu biçimde yazılır: öznitelik_adı="değer", yani bir adı ve bir değeri vardır. Öznitelik adı küçük harflerle yazılır ve özniteliğin değeri her zaman çift tırnak ("") veya tek tırnak (') içinde olmalıdır. Çift tırnak daha sık kullanılır. Tek tırnak, değerin içinde çift tırnak bulunduğunda kullanılır, örneğin ime='bulvar "8 Eylül" '.

Ayrıca, bağlantı etiketi (<a>) nereye "götüreceğini" bilmesi için href özniteliğini kullanır.

```
<a href="https://www.google.com"> Google'a Git</a>
```

Köprüler için başka öznitelikler de kullanılır. target="_blank" bağlantıyı yeni bir sekmede açar, böylece kullanıcı senin sayfandan ayrılmaz. title ise fareyi bağlantının üzerine getirdiğinde küçük bir metin açıklaması (tooltip) gösterir.

```
<a href="https://mk.wikipedia.org" title="Vikipedi'yi ziyaret et">Vikipedi'yi ziyaret et</a>
```

(basit bir köprü)

```
<a href="https://www.google.com" target="_blank"> Google'ı Yeni Sekmede Aç</a>
```

(yeni sekmede açılan köprü)

Resim etiketi (<img>) özeldir çünkü kendi başına kullanılır (kapanış etiketi yoktur). Bir resmi görüntülemek için src (resmin kaynağı) ve alt (alternatif metin) özniteliklerini kullanır:

```
<img src="resim.jpg" alt="Görselin açıklaması">
```

Resmin genişliği ve boyutu için de öznitelikler vardır: width ve height; bunlarla resmin boyutu piksel cinsinden belirtilir.

```
<img src="avatar-kullanici.png" alt="Resmin açıklaması" width="150" height="150">
```

İnternette en sık kullanılan özelliklerden biri, resmin kendisine tık**landığında bir bağlantıya yönlendirilmesidir (örne**ğin logoya tıkladığında ana sayfaya dönmek gibi).

Bunu yapmak için <img> etiketini <a> etiketinin başlangıç ve kapanış etiketleri arasına yerleştirmen yeterlidir.

```
<img src="youtube-logo.png" alt="YouTube Logosu" width="200">
```

```
  <img src="youtube-logo.png" alt="YouTube Logosu" width="200">
</a>
```

Tüm bu etiketler tek bir kod içinde birleştirilebilir ve ilgi çekici bir sayfa oluşturulabilir. Örneğin, aşağıdaki kod resim ve resim hakkında daha fazla bilgiye yönlendiren bir köprü içeren bir sayfa oluşturur:

```
<h1> Küçük bir yavru köpek resmi</h1>

<img src="https://images.unsplash.com/photo-1543466835-00a7907e9de1" alt="Yukarı bakan köpek" width="300">

<br> <a href="https://mk.wikipedia.org/wiki/Köpek" target="_blank">Vikipedi'de köpekler hakkında daha fazla bilgi okuyun</a>
```

HTML ile tablolar oluşturmak için hücreleri birleştirmeye yarayan özel öznitelikler kullanılır. Bir tablo oluştururken, satırlar ve sütunlar için birkaç yardımcı etiketin içine yerleştirildiği bir ana etiket kullanırız:

Tablo için temel etiketler:

- **<table>** – Tablonun tamamını başlatır ve bitirir.
- **<tr>** (Table Row) – Yeni bir yatay satır oluşturur.
- **<td>** (Table Data) – O satırda verilerin bulunduğu normal bir hücre (sütun) oluşturur.
- **<th>** (Table Header) – Başlık hücresi oluşturur (metin otomatik olarak kalın ve ortalanmış olur).

Bir ders programı oluşturduğumuzu ve bir dersin iki ders saati boyunca işlendiğini düşünelim. Bu durumda iki sütunu veya iki satırı birleştirmemiz gerekir. Bunun için yalnızca <td> veya <th> etiketlerinin içinde kullanılan aşağıdaki iki özniteliği kullanırız:

colspan özniteliği (Sütunları birleştirme)

Bu öznitelik, hücrenin yatay olarak birden fazla sütuna yayılmasını sağlar.

- **Örnek:** Eğer iki ders saati boyunca Bilişim dersi varsa, iki ayrı hücre oluşturmak yerine şöyle tek bir hücre oluşturabiliriz:

```
<td colspan="2"> Bilişim (blok ders)</td>
```

rowspan özniteliği (Satırları birleştirme)

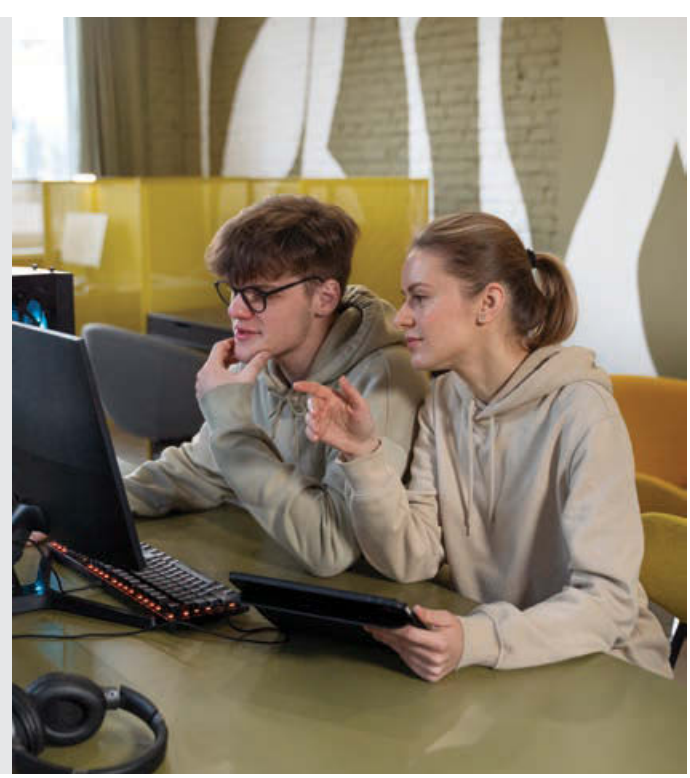
Bu öznitelik, hücrenin dikey olarak, yani aşağı doğru birden fazla satıra yayılmasını sağlar.

- **Örnek:** Her gün aynı saatte "Büyük Teneffüs" varsa, günlere ait satırları birleştirebiliriz:

```
<td rowspan="3">Büyük Teneffüs</td>
```

Tüm bu etiketleri ve öznitelikleri birleştiren tam kod:

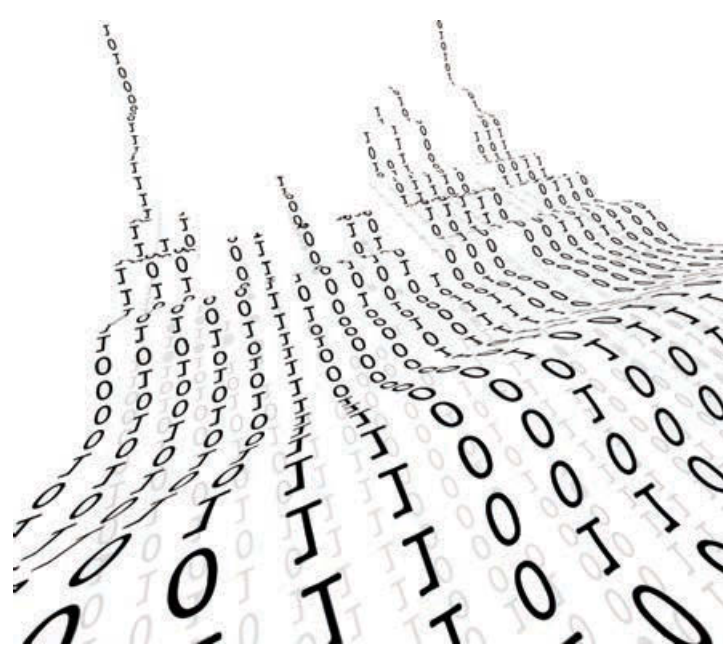
```
<table border="1"> <tr>
<th>Ad</th>
<th>Soyad</th>
 <th>Ders</th>
</tr>
<tr>
<td>Marko</td>
<td>Petrovski</td>
<td>Bilişim Teknolojileri</td>
</tr>
<tr>
<td>Ana</td>
<td>Stoyanova</td>
<td>Matematik</td>
</tr>
</table>
```



Resim 13: HTML ile oluşturulmuş ders programı

Tablo oluşturma bilgimizi, tablo, giriş alanları ve hücre birleştirme içeren bir kurs kayıt formu oluşturmak için kullanabiliriz. Bu örnek daha ileri düzeydedir, giriş alanlarını mükemmel şekilde hizalamak için kenarlıksız bir tablo (görünmez ızgara) kullanır ve en sonunda düğmeyi iki sütun boyunca genişletmek için colspan kullanır.

```
<h2>HTML Etiketleri Kursuna Kayıt</h2>
<p>Yerinizi ayırtmak için lütfen zorunlu alanları doldurun.</p>

<table>
 <tr>
  <td><b>Ad ve soyad:</b></td>
  <td><input type="text" placeholder="Adınızı girin..." required></td>
 </tr>

 <tr>
  <td><b>E-posta:</b></td>
  <td><input type="email" placeholder="ornek@email.com" required></td>
 </tr>

 <tr>
  <td><b>Doğum tarihi:</b></td>
  <td><input type="date"></td>
 </tr>

 <tr>
  <td colspan="2" style="text-align: center;">
   <br>
   <input type="submit" value="Başvuruyu Gönder">
  </td>
 </tr>
</table>
```



Aşağıdaki tablolarda HTML'de kullanılan tüm etiketler ve öznitelikler, açıklamalarıyla birlikte gösterilmiştir.

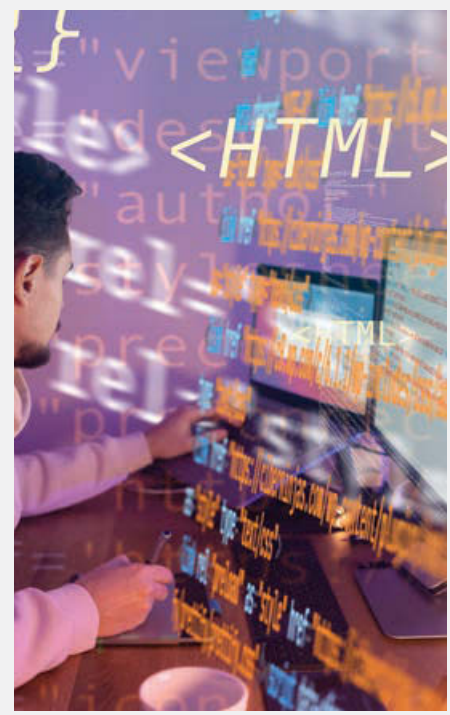
Etiket	Ne anlama gelir?	Açıklama / Nasıl kullanılır?	Kapanış etiketi var mı?
<h1> ile <h6>	Başlıklar	<h1> en büyük ana başlıktır, <h6> ise en küçük alt başlıktır.	Evet (örneğin </h1>)
** **	Paragraf	Normal metin yazmak için kullanılır.	Evet ( )
<a>	Köprü (Bağlantı)	Tıklanabilir bir bağlantı oluşturur.	Evet ()
<img>	Resim	Sayfaya bir resim ekler.	Hayır (tek başına kullanılır)
<table>	Tablo	Tablonun tamamını başlatır ve kapatır.	Evet (</table>)
<tr>	Tablo satırı	Tablonun içinde yeni bir yatay satır oluşturur.	Evet (</tr>)
<td>	Hücre	Satırın içinde verilerin bulunduğu normal hücredir.	Evet (</td>)
<th>	Başlık hücresi	Sütun başlığı için kullanılan hücredir (metin kalın olur).	Evet (</th>)
** **	Yeni satır	Metni veya öğeyi yeni bir satıra geçirir.	Hayır (tek başına kullanılır)

Tablo 2: HTML Etiketleri

Öznitelik	Hangi etikete eklenir?	Neyi kontrol eder? (Açıklama)	Nasıl yazılır? Örnek
href	Bağlantı etiketine (<a>)	Tıkladığımızda bizi nereye (hangi bağlantıya) götüreceğini belirtir.	href="https://google.com"
target	Bağlantı etiketine (<a>)	"_blank" değeri, bağlantının yeni bir sekmede açılacağını belirtir.	target="_blank"
src	Resim etiketine (<img>)	Resmin nereden alınacağını, yani tam dosya yolunu veya bağlantısını belirtir.	src="kopek.jpg"
alt	Resim etiketine (<img>)	Resmi kelimelerle açıklayan alternatif metindir.	alt="Siyah köpek yavrusu"
width	Resim (<img>) veya video etiketine	Öğenin genişliğini piksel cinsinden belirler.	width="300"
colspan	Tablodaki hücrelere (<td> / <th>)	Birden fazla sütunu yatay olarak birleştirir (örneğin çift ders saati için).	colspan="2"
rowspan	Tablodaki hücrelere (<td> / <th>)	Tabloda birden fazla satırı dikey olarak birleştirir.	rowspan="2"
border	Tablo etiketine (<table>)	Tablonun etrafına görünür bir kenarlık (çizgi) ekler.	border="1"
title	Herhangi bir etikete	Fareyi üzerine getirdiğinde küçük bir metin balonu gösterir.	title="Daha fazla bilgi"

Çok kolay bir şekilde bir ürünün veya kitabın basit bir tanıtımı oluşturulabilir. Bunun için bir başlık, açıklama içeren bir paragraf, kapak resmi, kitabın teknik ayrıntılarını içeren bir tablo ve yeni bir sekmede açılan satın alma bağlantısı kullanılır.

```
<h1>Kitap: Dijital Dünya</h1>
<p title="İçeriğin kısa açıklaması">Bu, bilgisayar teknolojisinin başlangıcı ve internetin gelişimi hakkında heyecan verici bir kitaptır.</p>
<img src="https://images.unsplash.com/photo-1543002588-bfa74002ed7e" alt="Teknoloji kitabı kapağı" width="250">

<h3>Teknik Detaylar</h3>
<table border="1">
 <tr>
  <th>Özellik</th>
  <th>Bilgi</th>
 </tr>
 <tr>
  <td><b>Yazar</b></td>
  <td>Marko Petrovski</td>
 </tr>
 <tr>
  <td><b>Dil</b></td>
  <td>Makedonca</td>
 </tr>
 <tr>
  <td><b>Sayfa sayısı</b></td>
  <td>240 sayfa</td>
 </tr>
</table>
```




```
<br><a href="https://www.amazon.com" target="_blank" title="Kitabı çevrimiçi satın al">Kitabı sipariş etmek için buraya tıklayın</a>
```

BİLGİ KONTROL SORULARI:

1. HTML öznitelikleri (attributes) HER ZAMAN nereye yazılır?
   - A) Belgenin en sonunda, </html> etiketinden sonra
   - B) Başlangıç etiketinin (opening tag) içinde
   - C) Kapanış etiketinin (closing tag) içinde
   - D) Başlangıç ve kapanış etiketleri arasında, metinle birlikte
2. Aşağıdaki tablo etiketlerinden hangisi yeni bir yatay satır oluşturmak için kullanılır?
   - A) <table>
   - B) <td>
   - C) <tr>
   - D) <th>
3. Kendi cümlelerinle HTML etiketi (tag) ile HTML özniteliği (attribute) arasındaki farkı açıkla. Ekranda bir şey görüntülemek için birlikte nasıl çalışırlar?

4. Bir resme width="300" özniteliğini verirsek ve height özniteliğini belirtmezsek, tarayıcıda resmin yüksekliğine ne olur?

- A) Yükseklik eksik olduğu için resim hiç görüntülenmez.
- B) Yükseklik 0 piksel olur ve resim görünmez.
- C) Tarayıcı, resmin bozulmaması için yüksekliği orantılı olarak otomatik şekilde ayarlar.
- D) Resim dikey olarak tüm ekran boyunca uzatılır.

5. Kullanıcının tıkladığında tamamen yeni bir sekmede https://www.google.com web sitesine yönlendirileceği bir resim (logotip.png) eklemek istiyorsun. Bunun için doğru HTML kodunu yaz.

6. Fare bir bağlantının üzerine getirildiğinde alternatif açıklama gösteren aşağıdaki kodlardan hangisi doğru şekilde yazılmıştır?

- A) Siteye git
- B) Siteye git
- C) Siteye git
- D) <link href="https://sajtedu.mk" title="Okul">Siteye git</link>

7. Aşağıdaki HTML tablo kodunu incele:

```HTML
<table border="1">
 <tr>
  <td>Makedonca</td>
  <td colspan="2">Matematik (çift ders)</td>
 </tr>
 <tr>
  <td>İngilizce</td>
  <td>Bilişim</td>
  <td>Görsel Sanatlar</td>
 </tr>
</table>
```

Kodu analiz et ve bu tablonun tarayıcıda neden görsel olarak bozulacağını (deforme olacağını) açıkla. Hücrelerin sayılmasındaki hata nerededir?

8. Görme engelli ve ekran okuyucu (Screen Reader) kullanan kişiler için bir web sitesi hazırladığını düşün. Aşağıdaki adımlardan hangisi onlar için önemlidir ve neden?

- A) Bağlantılarda her zaman target="_blank" kullanılmalıdır.
- B) <img> etiketindeki alt özniteliğine her zaman doğru ve açıklayıcı bir metin yazılmalıdır.
- C) Tablolarda her zaman border="1" kullanılmalıdır.
- D) Tüm başlıklar yalnızca <h1> olmalıdır.

9. En basit çevrim içi kartvizit (profil) için HTML kodu tasarla. Kod şunları içermelidir: favori başlığın, kendin hakkında bir paragraf, bir resmin ve favori sitene bir bağlantı. Etiketlerin doğru şekilde açılıp kapatılmasına ve özniteliklerin doğru sözdizimiyle yazılmasına dikkat et.

10. Yarınki okul ders programını hazırladığını düşün. 3. ve 4. derslerin aynı dersten blok ders (çift ders) olduğunu varsayalım. Tablonun yalnızca bu özel satırı (<tr>) için HTML kodunu yaz ve uygun birleştirme özniteliğini kullan.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

<a> etiketi ve href özniteliği yalnızca bir web sayfasından diğerine geçmek için kullanılmaz. Kullanıcının bilgisayarının veya telefonunun işletim sistemiyle de iletişim kurabilir.

Doğrudan e-posta gönderme bağlantısı (mailto:): Birinin bağlantıya tıkladığında Outlook veya Gmail gibi e-posta programının otomatik olarak açılmasını ve adresin hazır olarak yazılmasını istiyorsan bunu kullanabilirsin:

Bize e-posta gönder

Doğrudan arama bağlantısı (tel:): Bu, siteyi cep telefonundan görüntüleyen kullanıcılar için oldukça kullanışlıdır. Bağlantıya tıkladıklarında telefon otomatik olarak numarayı aramaya başlar:

Hemen ara

Çok uzun bir web sayfan varsa ve kullanıcının tek tıklamayla sayfanın en altındaki iletişim bölümüne "atlamasını" istiyorsan, href ile birlikte id özniteliğini kullanabilirsin:

İletişime geç

<h2 id="iletisim-bolumu">İletişim bilgilerimiz</h2>

loading="lazy" özniteliğine sahip akıllı resimleri, bilgisayarın "anlayabileceği" gelişmiş metin etiketlerini, download özniteliğini ve diğer özellikleri deneyebilirsin.

5.2.6 CSS – WEB SAYFALARINDA STİLLER

HTML, web sayfasının yapısından (iskeletinden) sorumluyken, CSS (Cascading Style Sheets) web sayfasının görünümünden, tasarımından ve estetiğinden sorumlu dildir. CSS olmadan tüm web sayfaları, mavi bağlantılara sahip sıradan siyah-beyaz Not Defteri metinleri gibi görünürdü.

CSS, HTML etiketlerinin ekranda nasıl görüntülenmesi gerektiğini tanımlar. HTML belgemize CSS eklemenin üç yolu vardır:

- **Satır içi (Inline):** style özniteliği aracılığıyla doğrudan HTML etiketinin içinde kullanılır. (Örnek: <p style="color: red;">).
- **Dahili (Internal): HTML** belgesinin içinde, <head> bölümüne yerleştirilen özel <style> etiketi içerisinde kullanılır.
- Harici (External) – Profesyonel yöntem: Kod, .css uzantılı ayrı bir dosyaya (örneğin style.css) yazılır ve daha sonra <link> etiketi aracılığıyla HTML dosyasına bağlanır.
- Bir öğeyi renklendirmek veya değiştirmek için öncelikle bilgisayara hangi öğeyi değiştirdiğimizi söylememiz gerekir. Bunu seçiciler (selectors) yardımıyla yaparız.

Temel seçiciler:

- **Etikete göre seçici (Element selector):** Sayfadaki o türden tüm etiketleri seçer.

```
p {
  color: blue; /* Tüm paragrafları mavi yapar */
}
```

- **Sınıf seçicisi (Class selector):** class özniteliğine sahip olan bir grup öğeyi hedefler. Her zaman nokta (.) ile başlar.

```
.kirmizi-metin {
  color: red;
}
```

- **Kimlik seçicisi (ID selector):** id özniteliğine sahip yalnızca bir benzersiz öğeyi hedefler. Her zaman diyez (#) işaretiyle başlar.

```
#ana-baslik {
  font-size: 30px;
}
```

CSS'te metnin rengi color komutuyla değiştirilir. Renkleri üç şekilde tanımlayabiliriz:

1. Renk adıyla: red, blue, green, orange.
2. HEX koduyla (diyez işaretiyle): #ff0000 (kırmızı), #ffffff (beyaz), #000000 (siyah).
3. RGB değerleriyle: rgb(255, 0, 0).

```
h2 {
  color: #2c3e50; /* Koyu mavi ton */
}
```

Metnin düzenlenmesi, web sayfasının güzel görünmesi için çok önemlidir. En önemli komutlar şunlardır:

- **font-family:** Harflerin yazı tipi stilini seçer (örneğin Arial, Verdana, Times New Roman). Bilgisayarda belirli yazı tipi bulunmuyorsa alternatif olarak kullanılmak üzere her zaman sans-serif (süslemesiz) veya serif (süslemeli) gibi genel bir yazı tipi ailesinin belirtilmesi önerilir.
- **font-size:** Harflerin boyutunu belirler. Genellikle piksel (px) cinsinden ifade edilir.
- **font-weight:** Metnin kalınlığını belirler (kalın için bold, normal için normal).

```
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
}
```

Sayfanın tamamının (<body> etiketi) veya belirli bir öğenin (örneğin tablo ya da paragraf) arka planı şu özelliklerle düzenlenir:

- **background-color:** Arka plan rengini değiştirir.
- **background-image:** Bir bağlantı üzerinden (url('resim.jpg')) arka plan resmi ekler.

```
body {
  background-color: #f4f4f4; /* Açık gri, yumuşak arka plan */
}
```

HTML'deki her öğeyi (başlık, resim, paragraf) bilgisayar görünmez bir dikdörtgen kutu olarak görür. Bu kutuların düzenini ve çevresindeki boşlukları üç temel komutla kontrol ederiz:

- **padding (İç boşluk):** Kutunun içindeki, metin ile kenarlık arasındaki boşluktur.
- **border (Kenarlık)**: Öğenin etrafındaki çizgidir. Kalınlığı, stili ve rengi olabilir, örneğin 2px solid black.

- **margin (Dış boşluk): Kutunun** dışındaki **boşluk**tur. Bu öğe ile çevresindeki diğer öğeler arasında mesafe oluşturur.

```
.kutu {
  border: 1px solid #ccc;
  padding: 20px; /* İçerideki metin kenarlıktan uzaklaşır */
  margin: 15px;  /* Kutu diğer öğelerden uzaklaşır */
}
```

Aşağıdaki örnekte harflerin ve arka planın rengini değiştirmek için kullanılan komutlar gösterilmiştir.

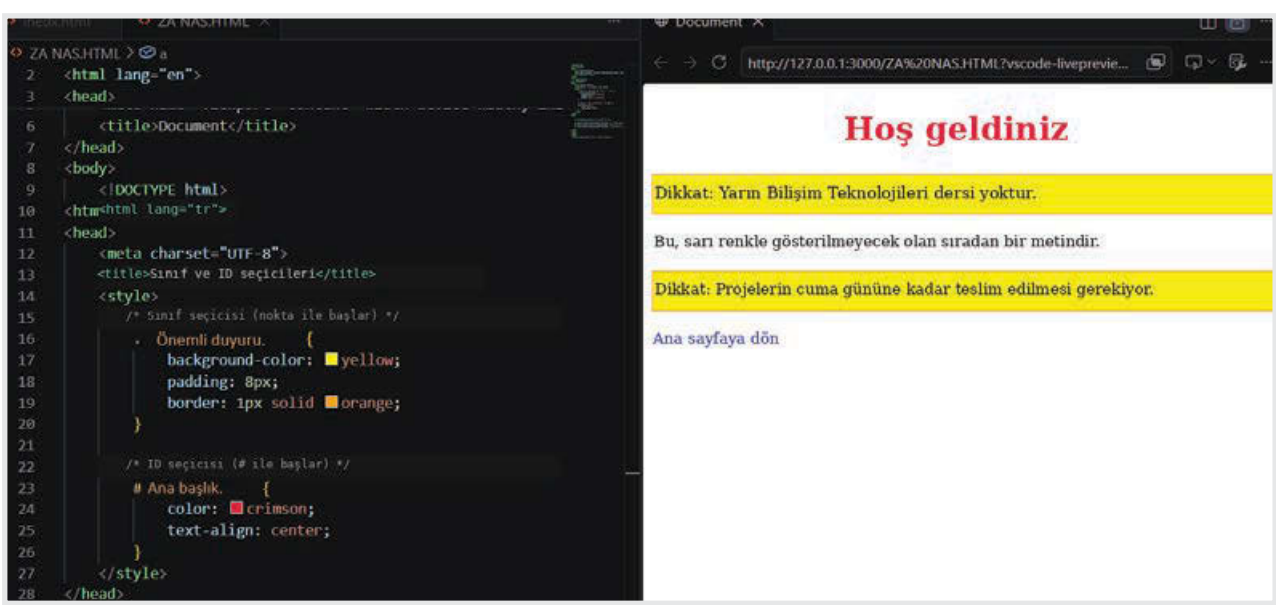
```
<!DOCTYPE html>
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Basit CSS örneği</title>
</head>
<body style="background-color: #f0f0f0; font-family: Arial;">

  <h1 style="color: blue;">İlk stillendirilmiş başlığım</h1>

  <p style="background-color: red; color: white; padding: 10px;">
    Bu metnin kırmızı arka planı ve beyaz harfleri vardır!
  </p>

</body>
</html>
```

Metnin hangi dilde görüntüleneceğini hangi komut tanımlar? CSS hangi yöntemle eklenmiştir?

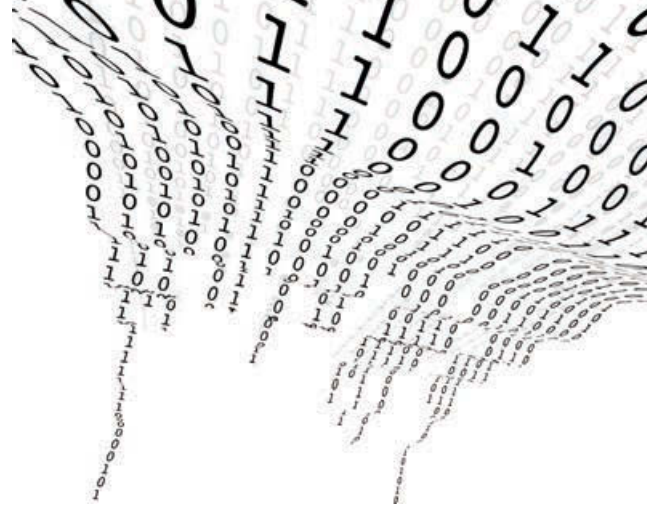
Sayfanın belirli (spesifik) öğelerini stillendirmek istediğimizde, tüm öğeleri aynı anda biçimlendirmek yerine her birini ayrı ayrı tanımlamak daha iyidir. Bunun için iki farklı "işaretleyici" (seçici) türü kullanılır: Sınıf (Class) ve Kimlik (ID).

```
<!DOCTYPE html>
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Sınıf ve ID seçicileri</title>
  <style>
    /* SINIF seçicisi (nokta ile başlar) */
    .onemli-duyuru {
      background-color: yellow;
      padding: 8px;
      border: 1px solid orange;
```

```
        }

        /* ID seçicisi (diyez ile başlar) */
        #ana-baslik {
          color: crimson;
          text-align: center;
        }
      </style>
    </head>
    <body>

      <h1 id="ana-baslik">Hoş Geldiniz</h1>

      <p class="onemli-duyuru">Dikkat: Yarın Bilişim Teknolojileri dersi yapılmayacaktır.</p>
      <p>Bu, sarı renkle gösterilmeyecek olan sıradan bir metindir.</p>
          <p class="onemli-duyuru">Dikkat: Projelerin cuma gününe kadar teslim edilmesi gerekmektedir.</p>

    </body>
    </html>
```



Resim 13: Sınıf ve kimlik (ID) ile biçimlendirme.



BİLGİ KONTROLÜ İÇİN SORULAR:

1. CSS kısaltması neyi ifade eder?
- A) Computer Style Sheets
- B) Creative Style Systems
- C) Cascading Style Sheets
- D) Colorful Style Software

2. CSS'te SINIF (class) seçicisi hangi sembolle başlar?
- A) Diyez işareti (#)
- B) Nokta (.)
- C) Ünlem işareti (!)
- D) Soru işareti (?)

3. p seçicisi ile #glaven-paragraf seçicisinin işlevleri arasındaki farkı açıklayınız. Her biri web sayfasındaki hangi öğeyi değiştirir?

4. Bir öğeye background-color: black; özelliği verildiğinde ve bilerek color komutu unutulduğunda, bu öğedeki metin neden kullanıcı tarafından görünmez hâle gelir?

5. Web sayfasındaki tüm <h2> başlıklarının metin rengini yeşil, yazı boyutunu 24 piksel ve yazı tipini Arial yapacak doğru CSS kodunu yazınız.

6. Aşağıdaki CSS kodlarından hangisi bir görselin etrafına 2 piksel kalınlığında siyah bir kenarlık doğru şekilde ekler?
- A) img { margin: 2px black; }
- B) img { border: 2px solid black; }
- C) img { padding: 2px solid; }
- D) img { line: 2px black; }

7. Aşağıdaki durumu inceleyiniz: Alt alta bulunan iki kutunuz var. Bu kutular arasındaki mesafeyi (boş alanı) artırmak istiyorsunuz. Box Model (Kutu Modeli) içinde hangi komutu kullanmalısınız ve bu durumda padding komutunun neden işe yaramayacağını açıklayınız.

8. Bir sınıf arkadaşınız tüm CSS kodunu inline (satır içi) stiller kullanarak, yani doğrudan her HTML etiketinin içine style="..." yazarak oluşturmuştur. Projede 5 farklı web sayfası bulunmaktadır. Bu yaklaşımı değerlendiriniz: Öğretmen ertesi gün 5 sayfanın tamamındaki arka plan renginin griden beyaza değiştirilmesini isterse ne olur? Harici bir CSS dosyası kullanmak daha iyi bir seçim olur muydu?

9. .urun-karti adlı bir sınıf için eksiksiz bir CSS stili tasarlayınız. Kutu, açık gri bir arka plana, 15 piksel iç boşluğa (padding), ince bir kenarlığa ve içerideki metin için Verdana yazı tipine sahip olmalıdır.

10. Benzersiz id="eylem-basligi" tanımlayıcısına sahip bir ana başlık oluşturacağınız kısa bir HTML ve CSS kodu yazınız. CSS bölümünde bu başlığın kalın (bold) yazılmasını ve sarı bir arka plana (background-color) sahip olmasını sağlayınız.

5.2.7. DÜZEN VE DUYARLI (RESPONSIVE) TASARIM

Bir web sayfası oluşturduğumuzda yalnızca öğeleri renklendirmek yeterli değildir. Öğeleri ekranda nasıl düzenleyeceğimizi (sola, sağa, yan yana) ve bu düzenin sayfa bilgisayarda, tablette veya cep telefonunda görüntülendiğinde nasıl otomatik olarak uyarlanacağını bilmek de önemlidir. Bu, uygun düzen ve duyarlı tasarım kullanılarak gerçekleştirilir.

Düzen, web sayfasının yapısını veya "iskeletini" ifade eder. Logonun nerede, menünün nerede, ana metnin nerede ve kenar çubuğunun (sidebar) nerede bulunacağını belirler.

CSS'te öğeler varsayılan olarak birbiri altında sıralanır (örneğin , <h1>, <p> gibi blok öğeleri). Farklı bir düzen oluşturmak için display komutunu kullanırız. Öğeleri düzenlemenin en modern ve en kolay yollarından biri Flexbox'tır (Flexible Box – Esnek Kutu). Bir üst kutuya display: flex; komutunu verdiğimizde, içindeki tüm öğeler otomatik olarak yan yana (sütunlar hâlinde) sıralanır, birbiri altında sıralanmaz.

```
.konteyner {
 display: flex;
 justify-content: space-between; /* Öğeleri sola ve sağa eşit aralıklarla uzaklaştırır */
}
```

Duyarlı Web Tasarımı (Responsive Web Design), web sayfalarının oluşturulmasında kullanılan ve sayfanın görünümünün ve içeriğinin, erişim sağlanan cihazın ekran boyutuna otomatik olarak uyarlanmasını sağlayan bir yaklaşımdır. Bu cihazın akıllı telefon, tablet, dizüstü bilgisayar veya büyük bir masaüstü monitörü olması fark etmez.

Mobil telefonlar ve bilgisayarlar için sayfanın ayrı sürümlerini oluşturmak yerine, duyarlı tasarımda aynı kod kullanılır. Bu kod, mevcut alana bağlı olarak esnek bir şekilde yeniden düzenlenir, daralır veya genişler.

Duyarlı tasarımın temel ilkeleri:

1. Akışkan düzenler (Fluid Layouts): Sabit piksel ölçüleri kullanmak yerine (örneğin width: 900px;), yüzdeler kullanırız (width: 100%; veya width: 50%;). Böylece öğeler, ekranın boyutuna bağlı olarak genişler veya daralır.
2. Akışkan görseller: Görselin telefon ekranının dışına taşmaması için CSS'te ona şu altın kuralı uygularız:

```
img {
 max-width: 100%;
 height: auto;
}
```

Modern web sayfaları için neden önemlidir?

- Mobil trafiğin hâkimiyeti: Küresel internet trafiğinin yarısından fazlası mobil cihazlardan gelmektedir. Web siteniz telefonda kötü görünüyor veya gezinmesi zorsa, çok sayıda potansiyel ziyaretçiyi kaybedersiniz.
- Daha iyi kullanıcı deneyimi (UX): Kullanıcılar sürekli yakınlaştırma yapmak, yatay olarak kaydırmak veya küçük düğmelere basmak istemezler. Duyarlı tasarım, her cihazda temiz bir görünüm, kolay okunabilir metin ve basit bir gezinme olanağı sunar.

- SEO açısından önemi (Arama Motoru Optimizasyonu): Google gibi arama motorları "mobile-first indexing" (mobil öncelikli indeksleme) kullanır. Bu, Google'ın arama sonuçlarında bir web sitesini hangi sıraya yerleştireceğine karar verirken öncelikli olarak sitenin mobil sürümünü değerlendirdiği anlamına gelir.
- Daha kolay bakım ve ekonomik olması: Masaüstü için bir, mobil cihazlar için başka olmak üzere iki farklı web sitesinin geliştirilmesi ve güncellenmesi için zaman ve para harcamak yerine, her yerde çalışan yalnızca tek bir sistemi yönetirsiniz.

Günümüzde duyarlı tasarım, artık yalnızca bir "ek özellik" veya lüks değildir. İnternette başarılı ve görünür olmak isteyen her profesyonel web sitesi için temel bir standarttır.

Web sitesini farklı ekran boyutlarına uyarlamak için CSS'teki en önemli ar**aç Media Quer**ies (Medya Sorguları) olarak adlandırılır. Bunlar filtreler veya koşullar gibi çalışır: "Eğer ekran 600 pikselden küçükse, bu yeni CSS kurallarını uygula". Bilgisayarda, yan yana yerleştirilmiş üç kutu öğeniz olduğunu (flex) düşünün. Mobil cihazda ekran dar olduğu için bu kutu öğeleri birbirine sıkışacaktır. Media Query yardımıyla bu öğelerin mobil cihazda alt alta sıralanmasını sağlayabiliriz.

```
/* Bilgisayarlar için standart CSS (öğeler yan yana) */
.ana-meni {
  display: flex;
  flex-direction: row;
}

/* Cep telefonları için özel kural (600 px'den daha dar ekranlar için) */
@media (max-width: 600px) {
  .ana-meni {
    flex-direction: column; /* Öğeleri alt alta sıralayacak şekilde düzenleriz */
  }
}
```

HTML'deki her öğe/kutu, Box Model (Kutu Modeli) adı verilen bir modele sahiptir. Bir kenarlık görmeseniz bile bilgisayar her öğe için dört şeyi hesaplar:

- **İçerik (content):** Metnin veya görselin kendisi.
- **İç boşluk (padding):** Metnin kenarlara yapışmaması için metnin etrafındaki boş alan.
- **Kenarlık (border)**: Öğenin etrafındaki çizgidir (görünmez olabilir veya renklendirilebilir).
- **Dış boşluk (margin):** Kutunun komşu kutulardan uzaklaşmasını sağlayan ve birbirlerine yapışmalarını önleyen boşluktur.

Bir web sayfası tasarlarken aslında yaptığımız şey, bu kutularla çalışmaktır: onları genişletir, daraltır, arka planlarını renklendirir ve aynı satırda mı yoksa alt alta mı duracaklarına karar veririz!

Aşağıdaki örnekte bir web sayfasının duyarlı tasarımı (responsive design) gösterilmektedir. Kodu VSC editöründe kullanın ve display: flex; ile @media özelliklerinin nasıl çalıştığını gözlemlemek için pencerenin boyutunu küçültüp büyüterek test edin.

```
<!DOCTYPE html>
```

```
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Kısa Duyarlı Tasarım</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      padding: 20px;
    }

    /* KONTEYNER: Bilgisayarda kartları yatay olarak (aynı satırda) sıralar */
    .kutu-grubu {
      display: flex;
      flex-direction: row;
      gap: 15px; /* Kutular arasındaki boşluk */
    }

    /* KART: Her bir kutu için stil */
    .kart {
      background-color: white;
      padding: 20px;
      border: 1px solid #ddd;
      flex: 1; /* Kutulara genişliği eşit şekilde paylaşmalarını söyler */
    }

    /* MEDYA SORGUSU: 600 pikselden daha küçük ekranlar için (mobil cihazlar) *
    @media (max-width: 600px) {
      .kutu-grubu {
        flex-direction: column; /* KUTULARI ALT ALTA SIRALAR */
      }
    }
  </style>
</head>
<body>

  <h2>Bizim Duyarlı Düzenimiz</h2>

  <div class="kutu-grubu">
    <div class="kart">
      <h3>Kutu 1</h3>
```

```
      <p>Bu, ilk kutunun içeriğidir.</p>
    </div>
    <div class="kart">
      <h3>Kutu 2</h3>
      <p>Bu, ikinci kutunun içeriğidir.</p>
    </div>
    <div class="kart">
      <h3>Kutu 3</h3>
      <p>Bu, üçüncü kutunun içeriğidir.</p>
    </div>
  </div>

</body>
</html>
```

BİLGİ KONTROLÜ İÇİN SORULAR:

1. Esnek düzen modelini (Flexbox) etkinleştirmek için hangi CSS komutu kullanılır?

- A) display: block;
- B) display: flex;
- C) position: static;
- D) float: left;

2. CSS @media kuralı (Media Queries) kullanılarak ne elde edilir?

- A) Web sayfasına video ve ses dosyaları eklenir.
- B) Ekranın boyutuna bağlı olarak farklı CSS stilleri uygulanır.
- C) Web sayfasının dili değiştirilir.
- D) Web sayfası sosyal ağlara bağlanır.

3. Duyarlı (responsive) bir web sayfası tasarlarken, ana konteynerlerin genişliği için sabit piksel değerleri (px) yerine yüzdeleri (%) kullanmak neden daha iyidir?

4. flex-direction özelliği row olarak ayarlandığında öğelerin davranışı ile column olarak ayarlandığında öğelerin davranışı arasındaki farkı açıklayınız.

5. Sayfadaki tüm görsellerin (<img>) duyarlı olmasını, yani kendi üst öğelerinin genişliğini aşmamasını ve en-boy oranını korumasını sağlayacak CSS kuralını yazınız.

6. Maksimum ekran genişliği 768 piksel olan cihazları (örneğin tabletleri) doğru şekilde hedefleyen kod aşağıdakilerden hangisidir?

- A) @media (width: 768px)
- B) @media (max-width: 768px)
- C) @screen (max-device: 768px)
- D) @media (min-width: 768px)

7. Öğrenci, bilgisayarda navigasyon menüsü ile ana içeriğin yan yana durduğu bir düzen oluşturmuştur. Siteyi mobil cihazda açtığında içerik sıkışmakta ve metin okunamamaktadır. Bu problemi analiz edin ve hangi duyarlı tasarım (responsive design) ilkesinin ihlal edildiğini ve mobil cihazlarda düzenin nasıl değiştirilmesi gerektiğini açıklayın.

8. Okul için yeni bir web sitesi geliştirmek üzere bir strateji seçmeniz gerektiğini düşünün. Birinci seçenek, iki ayrı sürüm oluşturmak (biri bilgisayar için eksiksiz, diğeri mobil için farklı bir bağlantıya sahip özel sürüm), ikinci seçenek ise Media Queries kullanan tek bir duyarlı (responsive) sayfa oluşturmaktır. Her iki seçeneği de gelecekte bilgilerin bakımı ve değiştirilmesi açısından değerlendirin.

9. Bilgisayarlarda ekran genişliğinin tam olarak %33,3'ünü kaplayacak .kutu-blok adlı bir sınıf için basit bir CSS kodu oluşturun. Ancak Media Query kullanarak mobil cihazlarda (600 pikselin altında) genişliğinin otomatik olarak %100 olmasını sağlayın.

10. Basit bir site üstbilgisi (Header) için HTML yapısı ve CSS kodu tasarlayın. Bu üstbilgide Başlık (solda) ve Bağlantılar (sağda) bulunmalıdır. Bilgisayarda Flexbox kullanılarak aynı satırda (yan yana) durmalıdırlar. Ekran 480 pikselin altına daraldığında ise başlık ve bağlantılar ortalanmalı ve alt alta sıralanmalıdır.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Duyarlı (responsive) tasarım yaparken yüzdeler (%) çoğu zaman yeterli değildir, çünkü "üst öğenin" (parent) kutusunun boyutuna bağlıdır. Bu nedenle profesyoneller, doğrudan ekranın Viewport (görüntü alanı) boyutuna bağlı olan ölçü birimlerini kullanırlar:

vw (Viewport Width – Görüntü Alanı Genişliği): 1vw, **ekranın g**enişliğinin %1'ine eşittir. width: 50vw; yazarsanız, öğe hangi cihazda açılırsa açılsın ekranın tam yarısını kaplar.

vh (Viewport Height – Görüntü Alanı Yüksekliği): 1vh, **ekranın** yüksekliğinin %1'ine eşittir. Bu, site yüklendiğinde ekranın tamamını yükseklik açısından kaplaması gereken başlangıç ("Hero") bölümleri oluşturmak için mükemmel bir yöntemdir:

```
.baslangic-bolumu {
  height: 100vh; /* Ekranın tamamının yüksekliğini kaplar */
}
```

Duyarlı tasarımdaki en büyük zorluklardan biri yazı tipi boyutudur. Başlık bilgisayarda çok büyükse, mobil cihazda üç satıra bölünebilir. Yalnızca yazı tipleri için onlarca Media Query yazmak yerine, modern CSS'te clamp() işlevi bulunur.

Bu işlev üç değer alır: minimum, önerilen (akışkan) ve maksimum boyut.

```
h1 {
  font-size: clamp(20px, 5vw, 45px);
}
```

Flexbox, öğeleri tek bir yönde (ya yalnızca satırda ya da yalnızca sütunda) sıralamak için harika olsa da, CSS Grid iki boyutlu bir düzen (aynı anda hem satırlar hem de sütunlar) oluşturmak için tasarlanmıştır. Grid ile gerçek bir gazete veya dergideki gibi bir düzen oluşturabilirsin.

```
.dergi-duzeni {
  display: grid;
   grid-template-columns: repeat(3, 1fr); /* Tam olarak 3 eşit sütun
oluşturur */
  grid-gap: 20px;
}
```

Mobil cihazlarda ise yalnızca bir medya sorgusuyla sütun sayısını kolayca değiştirebilirsin:

```
@media (max-width: 600px) {
  .dergi-duzeni {
    grid-template-columns: 1fr; /* Mobilde her şey 1 sütuna dönüşür */
  }
}
```

Bazen büyük bir tablo veya karmaşık bir görseli mobil telefonda göstermek mantıklı değildir, çünkü kullanıcının internetini yavaşlatabilir ve ekrandaki alanı gereksiz şekilde sıkıştırabilir. Media Queries (Medya Sorguları) ile belirli bir öğeyi mobil cihazlarda tamamen gizleyebilir, ancak bilgisayarda görünür bırakabilirsin:

```
/* Mobil cihazlarda yan taraftaki reklam bölümünü gizle */
@media (max-width: 600px) {
  .yan-reklam {
    display: none;
  }
}
```

5.2.8. ÜRETİCİ YAPAY ZEKÂNIN WEB GELİŞTİRMEDE KULLANIMI

Üretici yapay zekâ (Generative AI) teknolojisi, web sitelerinin oluşturulma biçiminde devrim yaratmıştır. ChatGPT, Claude, Microsoft Copilot gibi araçlar ve VS Code içindeki özel eklentiler (örneğin GitHub Copilot) artık birkaç saniye içinde karmaşık HTML ve CSS kodları yazabilmektedir.

Bu içerik, bu sürecin nasıl işlediğini, bilgisayara nasıl doğru talimatlar vereceğinizi ve yapay zekânın sizin için oluşturduğu kodu nasıl eleştirel bir şekilde analiz edeceğinizi anlamanıza yardımcı olacaktır.

1. Talimat (Prompt) Kavramı ve Açık Talimatların Oluşturulması

Yapay zekâ ile çalışırken en önemli kavramlardan biri prompt (istem) kavramıdır. Pr**ompt, kullanıcının** belirli bir sonuç elde etmek için yapay zekâ aracına girdiği metinsel talimat veya komuttur. Bizim durumumuzda bu sonuç web kodudur.

Yapay zekâya "Bana bir okul web sitesi oluştur." derseniz, ortaya çıkan sonuç çok genel ve basit olacaktır. Profesyonel bir kod elde etmek için yapılandırılmış ve ayrıntılı bir prompt kullanmanız gerekir.

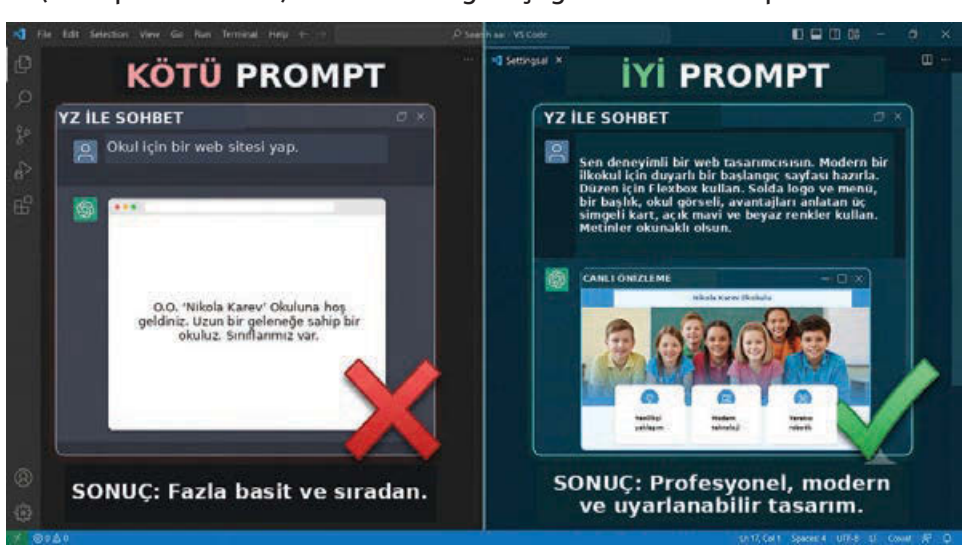
Açık ve net bir prompt nasıl oluşturulur?

Kaliteli bir prompt aşağıdaki unsurları içermelidir:

- **Rol (Role):** Yapay zekânın rolünü tanımlayın. (Örneğin: "Sen deneyimli bir web tasarımcısısın.")
- **Bağlam/Amaç: Web s**ayfası için tam olarak ne tasarlandığını belirtin.
- **Teknik gereksinim**ler: Hangi dillerin kullanılacağını belirtin. (HTML, iç/dış CSS stilleri, Flexbox düzeni vb.)
- **Özel ayrıntılar:** Renkler, bölüm sayısı ve mobil cihazlara uyumluluk gibi ayrıntıları belirtin.

Kötü bir prompt örneği: "Bana bir iletişim web sayfası için HTML kodu yaz."

Mükemmel bir prompt örneği: "Sen profesyonel bir web programcısısın. Bana 'İletişim' adlı duyarlı bir web sayfası için eksiksiz HTML ve dahili CSS kodu yaz. Sayfada Ad, E-posta ve Mesaj alanlarından oluşan bir form ve bir Gönder düğmesi bulunmalıdır. #2c3e50 koyu mavi arka plan ve beyaz metin kullanarak modern bir tasarım oluştur ve formu ekranın ortasına yerleştirmek için Flexbox kullan. Mobil cihazlarda (600 px'in altında) form, ekran genişliğinin %95'ini kaplamalıdır."



Resim 14: Genel ve özel prompt örneği

2. Web Sayfalarının Otomatik Olarak Oluşturulması ve Sonuçların Analizi

Yapay zekâ (YZ) asistanı kullanılarak web sayfası oluşturma süreci birkaç temel adımdan oluşur. Kaliteli bir kod elde etmek için süreç yalnızca "kopyala-yapıştır" işleminden ibaret değildir, planlama, talimat verme ve kontrol aşamalarını da içerir.

İşte sürecin adım adım tamamı:

- Yapının Planlanması (Kod Yazmaya Başlamadan Önce)

YZ aracını açmadan önce ne yapmak istediğiniz konusunda net bir fikriniz olmalıdır. Sayfanın düzenini (layout) taslak hâlinde belirleyin, ana menünün nerede bulunacağını, kaç adet içerik kutusu olacağını ve hangi renklerin kullanılacağını planlayın.

- Açık ve Anlaşılır Bir Prompt (Talimat) Oluşturma

Bu, sürecin en önemli adımıdır. Kısa ve belirsiz cümleler yerine, YZ asistanına ayrıntılı bir açıklama vermelisiniz.

- **Rol belirleyi**n: "Sen deneyimli bir web programcısısın."
- **Amacı tanımlayın:** "Bana ... için bir web sayfası oluştur."
- **Teknik kuralları belirtin:** "HTML5 ve <style> etiketi içerisinde dahili CSS kullan. Sayfanın düzenini Flexbox ile oluştur. Kod, medya sorguları kullanılarak mobil cihazlara uyumlu (responsive) olmalıdır."

- Kodun Oluşturulması ve Aktarılması

YZ asistanı kodu birkaç saniye içinde oluşturacaktır. Sizin göreviniz:

1. Kodu sohbet penceresinde incelemek.
2. Uygun kod bloğunu kopyalamak.
3. Kodu VS Code'da oluşturduğunuz yeni dosyaya (örneğin, index.html) aktarmak ve dosyayı kaydetmek.

- Tarayıcıda Analiz ve Test (Sonucun Kontrol Edilmesi)

Dosyayı kaydettikten sonra Chrome veya Edge tarayıcısında açın. Ardından oluşturulan çalışmayı eleştirel bir şekilde değerlendirin:

- **Görünü**m: Renkler ve yazı tipleri istediğin gibi mi?
- **Yapı: HTML** etiketleri mantıksal olarak doğru şekilde yerleştirilmiş mi (başlıklar, paragraflar, kutular)?
- **Uyumluluk (Responsive): Sayfa**nın küçük ekranlarda düzgün görüntülenip görüntülenmediğini kontrol etmek için tarayıcı penceresini küçültün. Öğelerin gerektiğinde doğru şekilde alt alta sıralanıp sıralanmadığını kontrol edin.

- Yineleme ve Manuel Düzenleme (Hataların Düzeltilmesi)

YZ asistanının ilk yanıtı neredeyse hiçbir zaman yüzde yüz kusursuz değildir. Bu aşamada programcı olarak siz devreye girersiniz.

- **Eğer küçük bir hata varsa:** Hatayı VS Code'da kendiniz manuel olarak düzeltin. Örneğin, CSS bölümünde bir rengi kırmızıdan maviye değiştirebilirsiniz. Bu, öğrenmenin en iyi yollarından biridir.
- **Daha büyük bir hata varsa veya tamamen eksik bir bölüm bulu**nuyorsa: YZ asistanına geri dön ve ek bir prompt ver. Örneğin: "Kod iyi, ancak metnin altına yeşil olacak bir buton daha ekle."

Bu döngüsel süreç sayesinde yapay zekâ, zor ve mekanik işleri hızlı bir şekilde tamamlamana yardımcı olurken, sen web sayfasının mantığı ve son görünümü üzerindeki tam kontrolü elinde tutarsın.

İyi bir YZ kullanıcısı olmak için her iki yaklaşımın da avantajlarını ve dezavantajlarını bilmelisin. İşte doğrudan bir karşılaştırma:

Özellik	Elle yazılmış kod (Hand-coded)	YZ tarafından oluşturulan kod (AI-generated)
Geliştirme hızı	Yavaştır (her satırın ve parantezin elle yazılmasını gerektirir).	Aşırı hızlıdır (tüm sayfalar birkaç saniye içinde hazır olabilir).
Kontrol ve anlama	Maksimumdur. Her kod satırının neden orada olduğunu tam olarak bilirsin ve bir hata olduğunda kolayca düzeltebilirsin.	Sınırlıdır. Kod çok uzunsa, bir şey bozulduğunda sorunun nerede ortaya çıktığını anlamak zor olabilir.
Optimizasyon ve mantık	Kod temizdir, tam olarak kendi ihtiyaçlarına göre yazılmıştır ve gereksiz tekrarlar içermez.	<img src="youtube-logo.png" alt="YouTube Logosu" width="200">
Konunun öğrenilmesi	Öğrenmenin en iyi yoludur. Gelişim, kendi hatalarını yaparak ve düzelterek gerçekleşir.	Sadece okunmadan körü körüne kopyalanıp yapıştırılırsa öğrenmeye yardımcı olmaz.

Yapay zekâ harika bir asistand**ır, ancak kötü bir yöneticidir. Onu kullanmanın en** iyi yolu, hızından yararlanarak sana web sayfasının hızlı bir iskeletini (şablonunu) hazırlatmak, ardından kodu kendin manuel olarak gözden geçirmek, temizlemek ve kendi kesin isteklerine ve kurallarına göre uyarlamaktır!

- Mevcut HTML/CSS Kodunun Yapı ve Görünüm Açısından Analizi

Yapay zekâ promptuna göre kodu oluşturduğunda iş bitmiş değildir. Geleceğin bir YZ kullanıcısı olarak sonucu analiz etmelisin. Yapay zekâ hatasız değildir – çoğu zaman güzel görünen, ancak arka planda ciddi eksiklikler içerebilen kodlar oluşturabilir.

YZ kodunu analiz ederken nelere dikkat etmelisin?

- **İşlevsellik: Tüm** bağlantılar çalışıyor mu? Form metin girişine izin veriyor mu?
- **Uyumluluk (Duyarlılık):** Tarayıcı penceresini sağa ve sola doğru hareket ettir. YZ, cihazlar için uygun @media sorgularını eklemiş mi, yoksa kod mobil cihazlarda bozuluyor mu?
- **Kodun temizliği:** Sınıflar açık ve anlaşılır şekilde adlandırılmış mı (örneğin .karticka-proizvod) yoksa kafa karıştırıcı mı?

YZ'nin iyi bir iş yapıp yapmadığını anlamak için kodu "okumalı" ve yapısını (HTML) ve g**örünümünü (CSS) ana**liz etmelisin.

YZ'nin aşağıdaki kod bölümünü oluşturduğunu düşünelim:

HTML

```
<div class="kart">
  <h2 class="baslik">Hoş Geldiniz</h2>
  <p>En son haberleri buradan okuyun.</p>
</div>
```

```
CSS
.kart {
  background: #fff;
  padding: 20px;
  border-radius: 5px;
}
.baslik {
  color: red;
}
```

Görünümün analizi (CSS): Kutunun beyaz bir arka planı, yuvarlatılmış köşeleri ve metnin kenarlara yapışmaması için iç boşluğu (padding) vardır. Başlık belirgin bir kırmızı renge sahiptir. Kodun yapısı ile görünümü birbirinden düzgün bir şekilde ayrılmıştır.

BİLGİ KONTROLÜ İÇİN SORULAR:

1. Kullanıcının kod elde etmek amacıyla YZ aracına girdiği metinsel talimat veya komut için hangi terim kullanılır?

- A) Script (Komut dosyası)
- B) Prompt (İstem)
- C) Tag (Etiket)
- D) Comment (Yorum)

2. YZ asistanının CSS kodunda oluşturduğu medya sorgularının (@media) temel görevi nedir?

- A) Web sayfasının farklı ekran boyutlarına uyum sağlamasını mümkün kılmak.
- C) Sitedeki görsellerin yüklenmesini hızlandırmak.
- B) Web sitesini sosyal medya platformlarına bağlamak.
- D) HTML yapısındaki mantıksal hataları bulmak.

3. "Bana spor ayakkabılar için bir web sayfası yap." promptu, YZ asistanıyla çalışırken neden eksik ve kötü bir prompt olarak kabul edilir?

4. YZ asistanı yüzde (%) veya akışkan bir düzen yerine piksel (px) cinsinden sabit genişlikler içeren bir kod oluşturduysa, bir web sayfasının cep telefonundaki görünümünde ne olur? Açıklayınız.

5. YZ asistanından üç bağlantıya sahip, koyu yeşil arka planlı ve düzeni Flexbox kullanılarak oluşturulmuş bir navigasyon menüsü (Header) oluşturmasını isteyen ayrıntılı ve yapılandırılmış bir prompt yazınız.

6. YZ asistanı tarafından oluşturulan kodda aşağıdaki satırı görürseniz: <meta name="viewport" content="width=device-width, initial-scale=1.0">, YZ asistanı sayfanın hangi cihazda doğru şekilde görüntülenmesini sağlamıştır?

- A) Yalnızca büyük monitörlü masaüstü bilgisayarlarda.
- B) Yalnızca sayfanın kâğıda yazdırılması sırasında yazıcılarda.

- C) Cep telefonları da dâhil olmak üzere tüm cihazlar için.
- D) Yalnızca CSS'yi desteklemeyen tarayıcılar için.

7. Bir öğrenci, bir kayıt formu oluşturmak için YZ asistanını kullanıyor. Kodu VS Code'a kopyaladıktan sonra "Gönder" düğmesinin, alanların altında en altta olması gerekirken sayfanın en üstünde durduğunu fark ediyor. Kodun hangi bölümünde (HTML veya CSS) yapısal hatanın yapıldığını analiz edin ve bu hatayı nasıl düzelteceğinizi açıklayın.

8. YZ tarafından oluşturulan aşağıdaki CSS kodunu inceleyin: .tekst { color: blue; font-size: 16px; font-style: italic; }. Öğrenci, kodun eğik yazı içermemesini istemişti. İsteğin yerine getirilmesi için bu kodda hangi komutun silinmesi veya değiştirilmesi gerektiğini açıklayın.

9. Yeni başlayanların programlamayı öğrenmesi ve anlaması açısından, YZ tarafından oluşturulan kodu elle yazılmış kodla karşılaştırarak kullanmanın avantajlarını ve dezavantajlarını değerlendirin.

10. YZ asistanının bilgisayarda yan yana duran iki karttan oluşan mükemmel bir düzen oluşturduğunu, ancak kodu tarayıcıda test ettiğinizde mobil cihazlar için responsive tasarım eklemeyi unuttuğunu düşünün. 600 pikselden küçük ekranlarda .kart sınıfına sahip kartları alt alta yerleştirmek için eksik olan CSS medya sorgusunu oluşturun (yazın).

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Yapay zekâ, yalnızca o anda yazdığınız promptu görmez, önceki konuşmayı da hatırlar. Bu hatırlama alanına bağlam penceresi (context window) adı verilir. Daha fazla bilgi edinmek istiyorsanız, bundan yararlanmanın en iyi yolu, kod istemeden önce YZ'ye önceden belirlenmiş kurallar vermektir.

Örneğin, günün ilk promptu şu şekilde formüle edilebilir:

"Bugünkü oturum boyunca senden HTML ve CSS kodu isteyeceğim. Görevin, hiçbir zaman eski CSS komutlarını (örneğin float) kullanmamak, her zaman Flexbox kullanmak ve kod içerisindeki yorumları her zaman Makedonca yazmaktır."

Böylece, kod oluşturmaya başlamadan önce asistanı sizin belirlediğiniz standartlara göre çalışması için yönlendirmiş olursunuz.

2.5.9. YZ KODUNUN DÜZENLENMESİ VE UYARLANMASI

YZ aracı kullanılarak bir web sayfası oluşturma örneğini inceleyelim. Daha önce VS Code'un bir YZ asistanına sahip olduğundan bahsetmiştik. VS Code'u açın ve aşağıdaki talimatı yazın:

Sen profesyonel bir web tasarımcısı ve kodlayıcısın. "IT ve Oyun Geliştirme Kulübü Kod-Elit" adlı bir okul kulübü için eksiksiz, temiz ve anlaşılması kolay HTML ve dahili CSS kodu yaz.

Sayfa aşağıdaki teknik ve tasarım standartlarını karşılamalıdır:

1. Yapı (HTML):

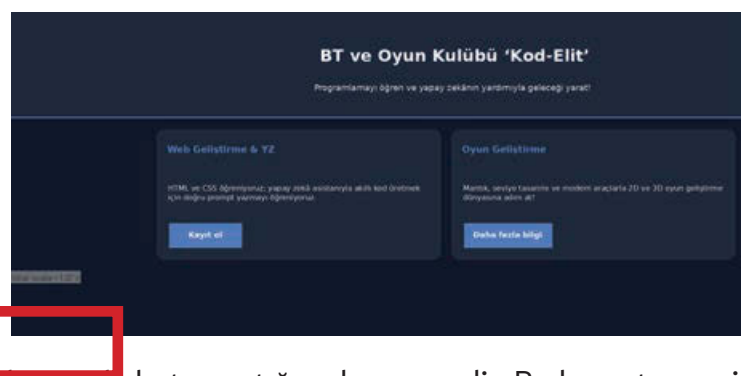
- Bir ana başlık ve kulübü açıklayan bir alt başlık içeren bir üst bilgi (header) bölümü olmalıdır.
- İçerik için iki karttan (kutudan) oluşan bir ana kapsayıcı olmalıdır: biri "Web Geliştirme ve YZ", diğeri "Oyun Geliştirme" için.
- Her kartta bir başlık, kısa bir metin ve işlem için bir düğme (bağlantı) bulunmalıdır.

2. Görünüm ve Düzen (CSS):

- Kontrast ve kolay okunabilirlik için çok açık renkli yazılara sahip modern bir koyu tema (koyu gri veya koyu mavi arka plan) kullan.
- Kartların ana düzenini Flexbox (display: flex) ile oluştur, böylece bilgisayar ekranında kutular yatay olarak, yan yana, eşit genişlikte ve aralarında uygun bir boşluk olacak şekilde yerleşsin.
- Kartların köşeleri yuvarlatılmış (border-radius) ve kenarlıkları sade olmalıdır.

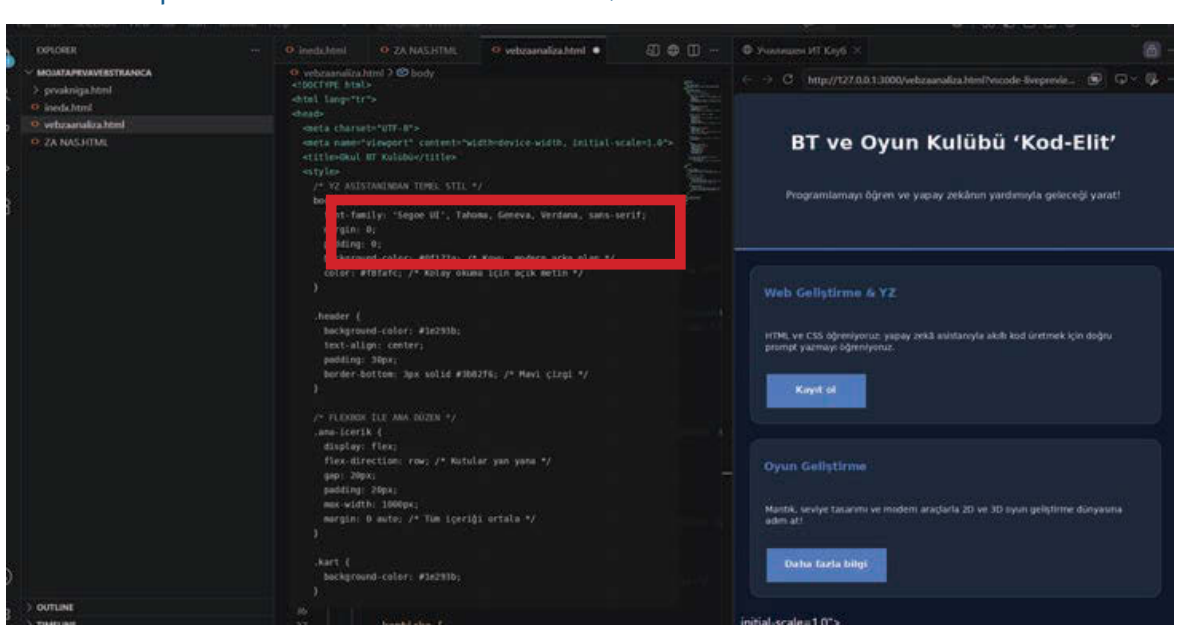
3. Uyarlanabilirlik (Responsive):

- Maksimum genişliği 768 piksel olan mobil cihazlar için bir medya sorgusu (@media) ekle.
- Sayfa mobil cihazda açıldığında Flexbox düzeni sütun (column) düzenine dönüşmeli, böylece kartlar dar ekranda daha anlaşılır olacak şekilde otomatik olarak alt alta yerleşmelidir.
- Oluşturulan kodu VS Code'da index.html uzantısıyla kaydedin ve sonucu görmek için tarayıcıda açın. Kodun bir satırının web sayfasında göründüğünü fark ediyorsunuz: initial-scale=1.0">.

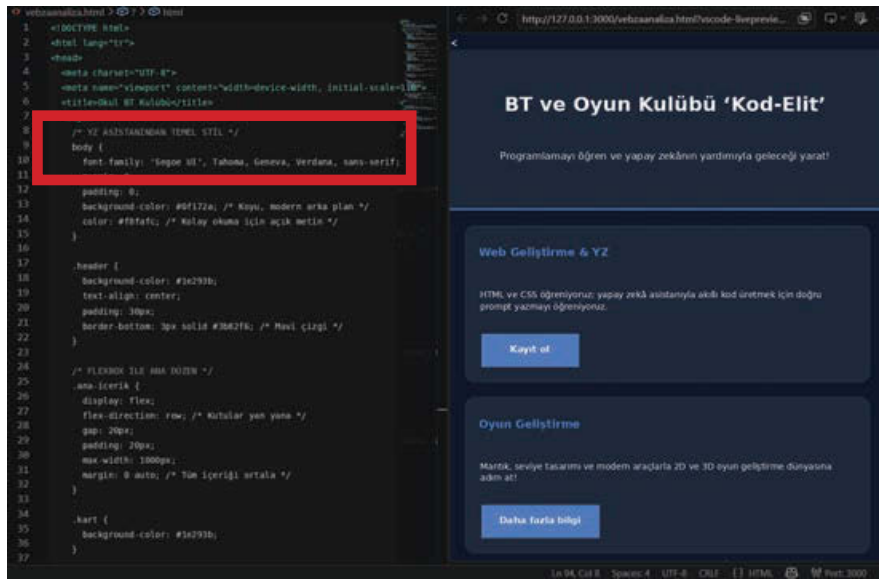


Bu, YZ asistanının bir yerde hata yaptığı anlamına gelir. Bu komutun ne işe yaradığını ve nereye yazılması gerektiğini analiz edin. YZ asistanından kodu düzeltmesini isteyin. Eğer başarılı olamazsa, <head> satırının altına bu komutu manuel olarak yazarak düzeltin.

<meta name="viewport" content="width=device-width, initial-scale=1.0">



Değişiklikleri kaydedin, F5 tuşuna basın ve web sayfası düzgün şekilde görüntülenecektir.



Web sayfasında yeni değişiklikler isteyin. Örneğin: "Bu kodda ana temayı değiştirmek istiyorum. CSS'yi, arka plan koyu yeşil olacak ve ana mavi düğmeler ile çizgiler neon sarısı olacak şekilde yeniden düzenle." veya: "Bu koddaki kutuların yapısını analiz et ve başlığı 'Siber Güvenlik' olan üçüncü bir kutu ekle. Bu kutu aynı stile sahip olsun ve Flexbox düzenine uyum sağlasın." Web sayfasının duyarlılığının kontrol edilmesi de istenebilir: "Alttaki medya sorgusuna bak. Mobil cihazlarda başlıktaki (header) metnin daha küçük olması ve ortalanması için kodu nasıl değiştirebilirim?"

5.2.10 WEB PROJESİ OLUŞTURMA

BİLGİ KONTROLÜ İÇİN SORULAR:

1. Bir öğrenci yapay zekâ asistanından duyarlı bir sayfa oluşturmasını istemiştir. Aldığı kodda viewport satırı hatalı yazılmıştır: <meta name="viewport" content="width=device-width", initial-scale=1.0">"> Bu sayfa tarayıcıda açıldığında ne olur?

- A) Sayfa hiç yüklenmez ve boş bir ekran görüntülenir.
- B) Sayfanın en altında initial-scale=1.0"> gibi fazladan bir metin görünür.
- C) Tüm CSS stilleri silinir ve metin kırmızı olur.
- D) İletişim formu otomatik olarak gizlenir.

2. Kartları yan yana sıralaması gereken aşağıdaki CSS kodunu inceleyiniz:

```
.kutu-grubu{
    display: block;
    flex-direction: row;
    gap: 15px;
}
```

3. Flexbox düzeninin doğru şekilde çalışmasını engelleyen yanlış komut hangisidir?

- A) gap: 15px;
- B) flex-direction: row;
- C) display: block;
- D) .gruplandırma-kutular sınıfının adında bir yazım hatası var.

4. Yapay zekâ asistanı tarafından oluşturulan aşağıdaki HTML kodunu inceleyiniz:

```
<div class="kart">
  <h3>Web Geliştirme</h3>
  <p>Yapay zekâ yardımıyla kod yazmayı öğreniyoruz.
</div>
```

Bu kod bölümünde hangi yapısal hata bulunmaktadır ve bu hata, altındaki diğer öğelerin görünümünü nasıl etkileyebilir?

5. Medya sorguları bölümünde yapay zekâ asistanı aşağıdaki sözdizimi hatasını yapmıştır:

```
@media max-width: 768px {
  .ana-icerik {
    flex-direction: column;
  }
}
```

Bu medya sorgusu neden tarayıcıda çalışmaz ve ilk satırda doğru kodu oluşturmak için hangi işaretler eksiktir?

6. Bir öğrenci yapay zekâ kodunu, kartın arka plan rengini değiştirmek için kullanmak istemektedir. Yapay zekâ asistanı aşağıdaki kodu yazmıştır:

```
.kart {
  background-color: #0f172a
  padding: 20px;
  border-radius: 12px;
}
```

Bu kodda hangi sözdizimi işareti eksiktir ve hangi CSS kuralı hiç uygulanmayacaktır?

7. Kodun italik yazı içermemesi istenmektedir. Aşağıdaki kodda hangi komut silinmeli veya değiştirilmelidir?

```
.buton {
  display: inline-block;
  background-color: #3b82f6;
  color: white;
  font-style: italic;
}
```

8. Yapay zekâ asistanının cep telefonları için duyarlı bir düzen oluşturma girişimini inceleyiniz:

```
.kart {
  width: 400px;
}

@media (max-width: 600px) {
```

```
    .kart {
      width: 400px;
    }
}
```

Kodu analiz ediniz ve bu tasarımın 360 piksel genişliğindeki standart bir mobil ekranda neden duyarlı olmayacağını açıklayınız. Medya sorgusundaki width değeri nasıl düzenlenmelidir.

9. Bir öğrenci aşağıdaki kodu kullanarak sayfa başlığı oluşturmuştur:

```
<h1 id="ana-baslik" class="ana-baslik">İT Kulübü Kod-Elit</h1>
```

CSS bölümünde iki kural bulunmaktadır: biri .ana-baslik { color: red; }, diğeri #ana-baslik { color: blue; }. Başlık hangi renkte görüntülenir ve neden? Seçicilerin özgüllüğü ilkesini açıklayınız.

10. Yapay zekâ asistanı, iki kartın bilgisayarda yan yana, mobil cihazda ise alt alta olması gereken bir kod oluşturmuştur. Ancak kod şöyledir:

```
.ana-icerik {
  display: flex;
  flex-direction: column;
}

@media (max-width: 768px) {
  .ana-icerik {
    flex-direction: column;
  }
}
```

Bu kodun bilgisayar ve mobil cihazlardaki işlevselliğini değerlendiriniz ve düzeni düzeltmek için flex-direction değerlerinin nasıl olması gerektiğini belirtiniz.

11. Bu, iki kart için yazılmış HTML kodudur ve yapay zekâ asistanı sınıfları kopyalarken mantıksal bir hata yapmıştır:

```
<div class="ana-icerik">
  <div class="kart">
    <h3>Web Geliştirme</h3>
    <p>Yapay zekâ ile kod yazmayı öğreniyoruz.</p>
  </div>
  <div class="ana-icerik">
    <h3>Oyun Geliştirme</h3>
    <p>2D ve 3D oyunlar oluşturuyoruz.</p>
  </div>
</div>
```

İkinci kutudaki yanlış sınıfı bulunuz ve düzeltiniz. Böylece ikinci kutunun da birinci kartla aynı stile sahip olmasını sağlayacak doğru HTML kodunu oluşturunuz.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Yapay Zekâ ile "Önce Mobil" (Mobile-First) Yaklaşımı

Çoğu başlangıç düzeyindeki kullanıcı, yapay zekâdan önce bilgisayar için bir web sayfası oluşturmasını ister, ardından sayfayı mobil cihazlar için düzeltir. Ancak modern bilişim sektöründe Mobile-First (Önce Mobil) yaklaşımı kullanılmaktadır. Bu yaklaşım, web sayfasının önce en küçük ekran için tasarlanması, daha sonra büyük monitörlere göre genişletilmesi anlamına gelir.

Yapay zekâdan bu yaklaşımla kod isterken talimatları şu şekilde değiştirirsiniz:

Temel CSS'nin tek sütunlu olarak (cep telefonu için) yazılmasını istersiniz.

Medya sorgularının max-width (maksimum genişlik) yerine min-width (minimum genişlik) kullanmasını istersiniz.

Bu şekilde, @media (min-width: 1024px) kullanıldığında yapay zekâ, ekran bilgisayar ekranı kadar büyüdüğünde etkinleşecek kurallar ekler. Bunun sonucunda web sayfası telefonlarda çok daha hızlı yüklenir.

Yapay Zekâ "Gereksiz Kodlarının" Otomatik Temizlenmesi (Refactoring)

Yapay zekâ çok fazla kod yazma eğilimindedir. Çoğu zaman aynı iş için üç farklı sınıf oluşturabilir veya tarayıcının zaten varsayılan olarak kabul ettiği CSS komutlarını ekleyebilir. Bu kodun temizlenmesi ve optimize edilmesi işlemine refactoring (yeniden yapılandırma) denir.

Yapay zekâ kodu oluşturduktan sonra, gelişmiş bir teknik olarak aynı kodu bir optimizasyon prompt'u ile tekrar yapay zekâya verebilirsiniz:

"Bu, oluşturduğun koddur. Şimdi bunu yeniden yapılandır: yinelenen CSS sınıflarını birleştir, tekrarlanan gereksiz margin değerlerini kaldır ve HTML yapısını daha kısa ve okunması daha kolay hâle getir."

VS Code'a Akıllı Eklentilerle Entegrasyon

İleri düzey programcılar kodu tarayıcıdan (örneğin ChatGPT'den) kopyalamak yerine, doğrudan VS Code editörünün içinde bulunan eklentileri (örneğin GitHub Copilot veya Codeium) kullanırlar.

Bu araçlar, klasörünüzdeki tüm dosyalara aynı anda erişebilir. Bu, yalnızca yeni kod oluşturabilecekleri anlamına gelmez, aynı zamanda yazma biçiminizi analiz edebilir ve siz daha yazmaya başlamadan, yalnızca bir tuşa (Tab) basarak bir sonraki kod satırını otomatik olarak önerebilirler.

Yapay zekânın nasıl çalıştığını ve kodda nasıl uyarlamalar yapıldığını öğrendikten sonra, artık bir web sayfası oluşturma sürecinin tamamına, yani kâğıt üzerindeki ilk fikirden son görünüme kadar olan sürece bakalım.

Profesyonel bir web sayfası oluşturma süreci her zaman 4 temel adımdan oluşur ve bu adımlar belirli bir sırayla uygulanmalıdır.

1. Planlama (Ne yapacağız ve kimin için?)

Planlama, web sayfasının temelidir. Tek bir kod satırı bile yazmadan önce aşağıdaki sorular cevaplanmalıdır:

- **Sayfanın amacı nedir?** (Bir portföy, okul web sitesi veya çevrim içi mağaza mı?)
- **Hedef kitle kimdir?** (Sayfayı gençler, öğretmenler veya oyuncular için mi hazırlıyoruz?)
- **Neleri içerecek?** Gerekli bölümlerin bir listesi hazırlanır: üst bilgi (header), ürünlerin bulunduğu kartlar, görsellerden oluşan galeri, iletişim formu.

2. Yapı (Kod nasıl düzenlenecek?)

- Yapı, sayfanın iskeletini, yani HTML etiketlerinin mantıksal olarak düzenlenmesini ifade eder. Profesyonel tasarımcılar bu aşamada tel çerçeve (**Wireframe) çizer. Bu, her öğe**nin nerede bulunacağını gösteren dikdörtgenlerden oluşan basit bir taslaktır.

Kodda bunu temiz ve anlamsal HTML5 yapısına dönüştürürüz:

- En üstteki menü için <header> kullanırız.
- Öğeleri gruplandırmak için açık sınıflara sahip kapsayıcılar (veya <section>) kullanırız.
- Başlıkların hiyerarşisine dikkat ederiz (<h1> ana başlık, <h2> ve <h3> kartların alt başlıkları için kullanılır).

3. Tasarım (Sayfa nasıl görünecek?)

Sağlam bir yapıya sahip olduktan sonra CSS ile biçimlendirme aşamasına geçeriz. Tasarım, görsel kimliği ve kullanıcı deneyimini belirler.

Burada öğrendiğimiz tüm CSS kurallarını uygularız:

- **Düzen (Layout):** Öğelerin esnek bir şekilde genişlemesi ve akıllıca sıralanması için Flexbox kullanırız.
- **Tipografi ve renkler:** Kolay okunabilen yazı tipleri ve birbiriyle uyumlu 2-3 renkten oluşan bir renk paleti seçeriz (örneğin, kontrast oluşturan mavi ayrıntılara sahip koyu ve modern bir tema).
- **Boşluk:** Öğelerin sıkışık ve birbirine "yapışık" görünmemesi için dış boşluk (margin) ve iç boşluk (padding) kullanırız.
- **Duyarlılık (Responsive):** Medya sorguları (@media) aracılığıyla tasarımın bilgisayarda üç sütundan mobil cihazda tek sütuna uyarlanmasını sağlarız.

4. İçerik Entegrasyonu (Sayfanın Gerçek Verilerle Doldurulması)

Entegrasyon, YZ asistanının bize verdiği "sahte" veya genel metinleri gerçek ve doğru bilgilerle değiştirdiğimiz son adımdır:

- Paragraflara (<p>) son metni yazarız.

Yapının analizi (HTML): Kod mantıklıdır. İçinde bir başlık ve bir paragraf bulunan dikdörtgen bir kapsayıcı (div) bulunmaktadır. Bu yapı, tüm öğelerin birlikte hareket etmesini ve tek bir bütün olarak düzenlenmesini sağlar.

- Geçici bağlantıları, gerçek resimlere ve harici web sitelerine giden doğru yollarla değiştiririz.
- Düğmeleri uygun işlevlere bağlarız.

Ancak bu 4 adımın tamamı belirtilen sırayla tamamlandıktan sonra eksiksiz, işlevsel ve profesyonel olarak hazırlanmış bir web sayfamız olduğunu söyleyebiliriz.

Hazırlık aşamalarını daha iyi öğrenmek için kendi kişisel web sayfanı (portföyünü) oluşturman en iyisidir. Burada yaptığın veya gelecekte yapmayı planladığın tüm alıştırmaları, oyunları veya web sitelerini gösterebilirsin. Profesyonel yazılımcıların her zaman sahip olduğu şeylerden biri de budur.

Web sayfasının içermesi gereken mimari ve öğeler:

1. Yapı ve anlamsal HTML etiketleri

Sayfa, açık ve mantıksal bölümlere ayrılmalıdır:

- <header>: En üstte, adı ve kısa bir alt başlıkla birlikte yer alır (örneğin: "Geleceğin web geliştiricisi").
- <main>: İçeriğin bulunacağı ana bölümdür.
- <section> sınıfı .proekti: Projelerine ait kartlar burada yer alacaktır.
- <footer>: En altta, telif hakkı bilgilerini içeren metin bulunur.

2. Düzen (Layout) – Flexbox

Projeler bölümünde 3 kart bulunmalıdır (**örneğin**: "Okul Web Sitesi", "İlk Oyunum", "Yapay Zekâ Çalışması").

- Bu 3 kart, display: flex; ve flex-direction: row; özelliklerine sahip tek bir üst kapsayıcının içine yerleştirilmelidir. Böylece bilgisayarda düzgün bir şekilde aynı satırda sıralanırlar.

3. Duyarlı Tasarım (@media sorguları)

Sayfa mobil telefonda da kusursuz görünmelidir.

- Kartların alt alta sıralanması için @media (max-width: 768px) eklenmeli ve burada Flexbox düzeni flex-direction: column; olarak değiştirilmelidir.
- Ekranda gereksiz metin oluşmaması için <head> bölümüne doğru viewport satırı mutlaka eklenmelidir.

Bu proje için YZ asistanı nasıl kullanılabilir? (Çalışma süreci)

Öğrenciler projeyi VS Code'da YZ ile aşağıdaki 3 adımı izleyerek yürütmelidir:

- **İlk prompt (İskeletin oluşturulması):** Öğrenci YZ'ye şunu yazar: "Sen benim mentörümsün. Kişisel BT portföyüm için bana HTML5 yapısı ve başlangıç CSS kodu yaz. Koyu gri bir arka plan, başlık, Flexbox kullanılarak yerleştirilmiş 3 proje kartından oluşan ana bölüm ve alt bilgi (footer) istiyorum. Ayrıca 768px'in altındaki mobil cihazlar için duyarlı (responsive) tasarım ekle."
- **Test etme ve hataları bulma (Analiz):** Öğrenci kodu tarayıcıda çalıştırır. YZ'nin herhangi bir noktalı virgülü (;) unutup unutmadığını, metnin bozulup bozulmadığını veya istemediği halde yanlışlıkla italik bir metnin ortaya çıkıp çıkmadığını kontrol eder.

- **İkinci prompt (Uyarlama ve geliştirme):** Öğrenci kodu analiz ettikten sonra değişiklik ister: "Kod harika, ancak şimdi ikinci kartın başlığını 'Benim YZ Asistan Projem' olarak değiştir ve tüm kartlardaki düğmelerin rengini yuvarlatılmış köşelerle birlikte açık mavi yap."
- **Kişiseldir:** Öğrenci kendisini tanıtır ve gerçek bir programcı gibi hisseder.
- **Tüm standartları içerir:** Yapı (HTML), estetik ve duyarlı tasarım (CSS) hakkında düşünmeyi ve teknolojiyle doğru iletişim kurmayı (Prompt mühendisliği) gerektirir.
- **Kolayca kontrol edilebilir:** Öğretmen/değerlendirici olarak tarayıcı penceresini küçülttüğünüzde Flexbox'ın düzgün çalışıp çalışmadığını hemen görebilirsiniz.

BİLGİ KONTROL SORULARI:

1. Bir öğrenci, bilişim kulübü için bir web sitesi yapmak istiyor. Kodlamaya başlamadan önce sınıf arkadaşlarının en çok ihtiyaç duyduğu bilgileri araştırıyor ve bölümlerin bir listesini hazırlıyor. Öğrenci bu sırada projenin hangi aşamasındadır?

- A) Tasarım
- B) Yapı
- C) Planlama
- D) İçerik entegrasyonu

2. Başlıkların, resimlerin ve kartların nerede bulunacağını belirlemek için "wireframe" (tel kafes) veya dikdörtgenlerden oluşan basit bir taslak oluşturduğumuzda aslında neyi tanımlıyoruz:

- A) Sitenin arka plan rengini
- B) Sayfanın HTML yapısını
- C) Sunucunun yüklenme hızını
- D) Harici PDF belgeleriyle bağlantıyı

3. Aşağıdaki durumu incele: Öğrenci HTML kodunu tamamlamış ve şimdi proje kartlarını Flexbox kullanarak düzenlemek, kutuların köşelerini yuvarlatmak ve cep telefonları için medya sorguları eklemek istiyor. Öğrenci bu adımda projenin hangi aşamasını gerçekleştiriyor ve hangi dili kullanıyor? Açıkla.

4. "İçerik entegrasyonu" aşaması ne anlama gelir ve öğrencinin portföyünün işlevsel hâle gelmesi için Yapay Zekâ asistanının href="#" olarak oluşturduğu bağlantılarla tam olarak ne yapması gerekir?

5. Bir öğrenci, iletişim formunu görüntülemek için kodunda şu bağlantıyı oluşturmuştur: Aç. Bu yolun neden yanlış ayarlandığını (yerel yol yerine mutlak yol) analiz ediniz ve başka bir kişi sayfayı kendi bilgisayarında açmaya çalıştığında bunun nasıl bir etkisi olacağını açıklayınız.

DAHA FAZLA BİLGİ EDİNMEK İSTEYENLER İÇİN

Web geliştirmede planlama, ziyaretçinin sayfa içerisindeki her hareketini ve tıklamasını önceden tam olarak belirleyen ayrıntılı bir kullanıcı akışı (User Flow) oluşturmayı da içerir. Bu ileri aşamada profesyoneller, ilk HTML kodu satırı yazılmadan önce bile gerçek bir site gibi görünen ve tepki veren etkileşimli bir prototip oluşturmak için Figma veya Adobe XD gibi özel dijital araçları kullanırlar.

Buna ek olarak, SEO stratejisi (arama motoru optimizasyonu) de planlanır. Bu strateji sayesinde, sayfanın daha sonra arama motorları tarafından kolayca bulunabilmesi için başlıklarda ve meta etiketlerde kullanılacak anahtar kelimeler önceden belirlenir.