EĞİTİMDE KURAM VE UYGULAMA (EKU)
JOURNAL OF THEORY AND PRACTICE IN EDUCATION
ISSN: 1304-9496
EKU, 2025: 21(2), 174-198

ARAŞTIRMA MAKALESİ

MÜHENDİSLİK ÖĞRENCİLERİNİN PROGRAMLAMA ÖZ-YETERLİKLERİ ALGI DÜZEYLERİNİN BELİRLENMESİ VE ÖĞRENCİLERİN PROGRAMLAMA HAKKINDAKİ GÖRÜŞLERİ

Fırat **YAVUZ**[1], Ayşen **KARAMETE**[2]

[1]Balıkesir Üniversitesi, Fen Bilimleri Enstitüsü, Balıkesir, Türkiye. frat.yvuzz@gmail.com
[2]Balıkesir Üniversitesi, Necatibey Eğitim Fakültesi, Balıkesir, Türkiye. karamete@balikesir.edu.tr

Öz

Bilgisayar programlama, mühendislik öğrencileri için kritik bir beceri haline gelmiş; bu alandaki öz-yeterlik algıları, öğrenme süreçlerini doğrudan etkilemektedir. Bu çalışmada, mühendislik öğrencilerinin bilgisayar programlamaya yönelik öz-yeterlik düzeyleri ve programlama dersi hakkındaki görüşleri incelenmiştir. 2022-2023 Bahar Yarıyılında, Türkiye'deki bir devlet üniversitesinin mühendislik fakültesinde yapılan araştırmada, karma yöntem yaklaşımı benimsenmiştir. Bu kapsamda, 355 öğrenciden nicel ve 12 öğrenciden ise nitel olmak üzere iki aşamalı veri toplanmıştır. Nicel veriler Bilgisayar Programlama Öz-Yeterlik Ölçeği ile elde edilmiş; nitel veriler ise araştırmacılar tarafından oluşturulan yarı yapılandırılmış görüşme formu ile toplanmıştır. Bulgular, öğrencilerin genel olarak orta düzeyde öz-yeterliğe sahip olduklarını ortaya koymuştur ($\bar{X}$=63.86). Özellikle iş birliği ve mantıksal düşünme boyutlarında yüksek, algoritma geliştirme boyutunda ise görece düşük puanlar gözlenmiştir. Cinsiyet öz-yeterlik üzerinde anlamlı bir fark yaratmazken; bölüm, bilgisayar sahipliği ve alınan ders sayısı gibi değişkenler anlamlı farklar göstermiştir. Nitel bulgular, öğrencilerin algoritma tasarımı, hata ayıklama ve iş birliğini programlama sürecinde önemli bulduklarını göstermektedir. Bu çalışma, mühendislik öğrencilerinin öz-yeterliklerini etkileyen faktörlerin anlaşılmasına katkı sunarken, öğretim programlarının öğrencilerin algoritma geliştirme ve uygulama becerilerini destekleyecek şekilde yeniden yapılandırılması gerektiğine işaret etmektedir.

Anahtar Kelimeler: Bilgisayar programlama, bilgi işlemsel düşünme, öz-yeterlik.

Geliş: 24.07.2025 **Kabul:** 10.12.2025

Sorumlu Yazar: Fırat YAVUZ
Kaynak Gösterimi: Yavuz, F. & Karamete, A. (2025). Mühendislik öğrencilerinin programlama öz-yeterlikleri algı düzeylerinin belirlenmesi ve öğrencilerin programlama hakkındaki görüşleri *Eğitimde Kuram ve Uygulama, 21*(2), 174-198. doi.org/10.17244/eku.1750065

RESEARCH ARTICLE

DETERMINATION OF ENGINEERING STUDENTS' PROGRAMMING SELF-EFFICACY PERCEPTION LEVELS AND STUDENTS' OPINIONS ON PROGRAMMING

Abstract

Computer programming is a critical skill for engineering students, and self-efficacy in this domain directly influences their learning. This study examined engineering students' self-efficacy levels in computer programming and their opinions about the programming course. A mixed-methods approach was adopted in a study conducted at the Faculty of Engineering of a state university in Turkey during the 2022-2023 Spring Semester. Accordingly, two-stage data were collected, comprising quantitative data from 355 students and qualitative data from 12 students. The Computer Programming Self-Efficacy Scale and a semi-structured interview form were used as data collection tools. Findings indicated that students had a moderate overall level of self-efficacy ($\bar{X}$=63.86). While cooperation and logical thinking dimensions had high scores, algorithm development scored relatively low. Gender did not significantly affect self-efficacy; however, department, computer ownership, and number of programming courses taken did. Qualitative findings emphasized the importance of algorithm design, debugging, and collaboration in developing self-efficacy. The study highlights the need to revise curricula to strengthen students' algorithmic thinking and practical skills in programming.

Keyword: Computer programming, computational thinking, self-efficacy.

Received : 24.07.2025 **Accepted:** 10.12.2025

Corresponding Author: Fırat YAVUZ

Citation: Yavuz, F. & Karamete, A. (2025). Mühendislik öğrencilerinin programlama öz-yeterlikleri algı düzeylerinin belirlenmesi ve öğrencilerin programlama hakkındaki görüşleri Eğitimde Kuram ve Uygulama, 21(2), 174-198. doi.org/10.17244/eku.1750065

Extended Summary

Introduction

The increasing importance of programming transcends its role as a growing industry; it is fundamentally recognized as an activity that cultivates systematic thinking, the ability to discern relationships, and crucial problem-solving skills. Programming education equips individuals with technical proficiency combined with creativity, leading to greater efficiency and productivity (Aytekin et al., 2018). It is considered the new method of "thinking" and "producing" essential for the 21st century (Sayın & Seferoğlu, 2016). Given that programming has largely replaced classic calculations and measurement techniques in engineering fields, it is an indispensable core competency for all engineering students, not just those in computer science (Gürer & Tokumacı, 2020).

The foundation of programming success in this information age lies in Computational Thinking (CT), defined as a problem-solving method utilizing core skills from mathematics and computer science, encompassing algorithmic thinking, critical thinking, creativity, and collaboration (Wing, 2006; ISTE, 2015). Research consistently shows that programming instruction, particularly block-based and game-based approaches, significantly develops CT skills (Alsancak Sırakaya, 2019; Özmen Yağız & Usluel, 2024), underscoring that the primary goal of this education is to strengthen CT's fundamental components.

Among the internal and personal resources influencing learning performance, Self-Efficacy (Bandura, 2001) is paramount. Programming self-efficacy—a belief in one's capacity to successfully execute programming tasks—directly impacts students' motivation, resilience in facing challenges, and ultimately, academic achievement (Mazman & Altun, 2012). Literature indicates that engineering students' programming self-efficacy is often moderate (Hanjani, 2019; Gezgin & Adnan, 2016). This perception is influenced by various factors, including the student's department, gender, prior knowledge, and project motivation, with a strong, positive relationship consistently reported between self-efficacy and attitude toward programming (Özyurt & Özyurt, 2015). Importantly, the belief is not fixed; it can be significantly shaped by pedagogical approaches. Creative, context-based learning activities like Digital Story Design (Yıldız Durak, 2018) and application-heavy methods are shown to enhance students' self-efficacy and engagement. Conversely, high programming anxiety, particularly observed in female students (Keskinkılıç & Kalelioğlu, 2024), acts as an inverse indicator, highlighting the need for environments that bolster confidence and reduce stress.

Given the existing literature pointing to engineering students' low self-efficacy and initial perception of programming as difficult (Abdüsselam et al., 2021; Askar & Davenport, 2009), while most research focuses on cognitive dimensions (e.g., misconceptions, learning process), studies concentrating on students' self-efficacy are limited.

Therefore, the aim of this study is to examine engineering students' self-efficacy toward computer programming and their views regarding the course. The study also intends to determine if programming self-efficacy varies across demographic and educational variables. To achieve this, the following research questions are addressed:

1. What is the level of engineering students' overall programming self-efficacy?
2. What is the level of engineering students' programming self-efficacy across the dimensions of control, logical thinking, debugging, algorithm, and collaboration?
3. Does engineering students' programming self-efficacy show a significant difference based on variables such as gender, department, number of courses taken, and computer ownership?
4. What are the engineering students' views regarding the programming course?

Method

This study employed an enriched mixed-methods design in which quantitative and qualitative data were collected concurrently. Demographic variables such as gender, department, number of programming courses taken, and computer ownership were considered independent variables, while programming self-efficacy and computational thinking skills were the dependent variables. The quantitative sample consisted of 355 engineering students from a public university located in western Turkey, selected through convenience sampling. The majority of participants were male. Additionally, qualitative data were collected through semi-structured interviews with 12 students. The data collection instrument comprised three sections. The first section included demographic questions. In the second section, the "Computer Programming Self-Efficacy Scale," originally developed by Tsai et al. (2019) and adapted into Turkish by Ekici and Çınar (2020), was used. The 16-item, 5-point Likert scale includes five sub-dimensions: control,

logical thinking, debugging, algorithm, and collaboration. The internal consistency of the scale was found to be high (α = .911; in this study, α = .962). The third section involved a semi-structured interview form developed by the researchers for qualitative data collection. Quantitative data were analyzed using SPSS. Normality of the data distribution was confirmed through skewness and kurtosis values, allowing for the use of parametric tests. Qualitative data were subjected to both descriptive and content analysis. Audio-recorded interviews were transcribed, irrelevant content was removed, and data were categorized and coded under themes. Validity was ensured through expert review and participant validation. The intercoder agreement was calculated as 95%, indicating high reliability. The trustworthiness of the qualitative findings was strengthened through triangulation, expert validation, and consistency checks.

Results

This study examined engineering students' self-efficacy perceptions regarding computer programming through both quantitative and qualitative data. The findings revealed that students generally have a moderate level of programming self-efficacy. Notably, lower scores in algorithm development indicate that students need more support in abstract thinking and problem-solving processes. In contrast, high self-efficacy scores in collaboration and logical thinking appear to align with the group work and problem-based structure of engineering education. The results showed that gender had no significant effect on self-efficacy, while variables such as academic department, computer ownership, and the number of programming courses taken significantly influenced self-efficacy levels. In particular, students from computer engineering departments exhibited higher self-efficacy, likely due to greater technical infrastructure and more opportunities for hands-on practice. Qualitative data highlighted several key factors affecting students' self-efficacy, including instructors' attitudes, peer support, individual interest, and opportunities for practice. Participants emphasized that insufficient practice reduced their confidence in programming, whereas supportive and patient instructors enhanced their motivation. Furthermore, the need for more applied instruction to foster computational and algorithmic thinking skills was strongly emphasized. In this context, it is recommended that structured, practice-based instructional approaches be designed to enhance students' programming self-efficacy, taking into account individual differences and providing a supportive learning environment..

GİRİŞ

Programlama, her geçen gün büyüyen bir sektör olmanın yanı sıra, bireylerin sistematik düşünme, olaylar arasındaki ilişkileri görebilme ve problem çözme becerilerini geliştiren bir etkinlik olarak ele alındığından önemi günden güne artmaktadır. Programlama eğitimi, bireylerin hayatlarında teknik becerilerini yaratıcılıkla birleştirmeyi, daha verimli ve üretken olmayı sağlamaktadır. Kodlamanın, insanların daha iyi kararlar almasına ve daha yüksek yaşam standartlarına ulaşmasına yardımcı olan bir araç olduğunu belirten (Aytekin vd., 2018), kodlamayı çağımızda ve gelecek dönemlerde önemli bir alan şeklinde tanımlamaktadır. Sayın ve Seferoğlu (2016) ise kodlamayı 21. yüzyılda ihtiyaç duyulan "düşünmenin" ve "üretmenin" yeni bir yolu olarak görmektedir.

Programlama becerisi, günümüzde ortaöğretim ve üniversite düzeyinde kritik bir yeterlik olarak kabul edilmektedir. Programlamanın yaygınlaşması; mühendislik alanındaki klasik hesaplamaların ve ölçüm tekniklerinin yerini alması nedeniyle mühendislik çalışmalarını programlamadan ve bilgisayardan ayrı düşünmek mümkün değildir (Gürer ve Tokumacı, 2020). Dolayısıyla, bilgisayar programlama sadece bilgisayar bilimleri alanında öğrenim gören öğrencilerin değil, mühendislik bölümlerindeki öğrencilerin de temel ihtiyacı olmaktadır (Gezgİn ve Adnan, 2016).

Yaşadığımız bilgi çağında hızla gelişen teknolojilerin etkisiyle bireylerde bulunması gereken temel becerilerden biri de bilgi işlemsel düşünme becerisidir (Wing, 2008). Wing (2006) tarafından bilgi işlemsel düşünme, matematik ve bilgisayar bilimlerinin temel becerilerinden faydalanarak, problem çözmeye, sistemler tasarlamaya ve insan davranışını anlamaya yönelik bir yöntem olarak tanımlanmıştır. Uluslararası Eğitim Teknolojileri Topluluğu'na (International Society for Technology in Education [ISTE]) (2015) göre bilgi işlemsel düşünme, algoritmik düşünme, eleştirel düşünme, problem çözme, yaratıcılık, iş birliği ve iletişim becerilerinin bir araya gelmesiyle oluşmaktadır.

Öğrencilerin ve bireylerin öğrenme performansını ve öğrenme ortamına katılımlarını etkileyen kavramlar arasında öz yeterlik gibi içsel ve kişisel kaynaklar önem kazanmaktadır (Chen, 2017). Bandura'ya (2001) göre öz yeterlik, bireylerin hedeflere ulaşma ve hayatlarını etkileyen çevreyi kontrol etme yetenekleri hakkındaki fikirleri olarak tanımlanmaktadır. Bireylerin öz yeterlik inançları, onların başarı için önemli olan kavramları ve teknolojileri öğrenmek ve uygulamak için motive etmeye yardımcı olur (Kher vd., 2013). Bilgisayar öz yeterliği ise, insanların sorunlarını çözmek ve durumları yönetmek için bilgisayarı etkili bir şekilde kullanma kapasiteleri ile ilgilidir (Marakas vd., 1998). Programlama alanındaki öz yeterlilik inancı, bireylerin zorluklarla başa çıkma direncini ve genel performansını doğrudan etkilemektedir; bu yönüyle programlama öz yeterliliği, akademik başarının kritik bir belirleyicisi olarak öne çıkmaktadır (Mazman ve Altun, 2012).

Alanyazın taraması, Programlama ve Bilgi İşlemsel Düşünme ile Programlama Öz Yeterliliği ve Tutum İlişkisi olmak üzere iki ana başlık altında toplanmıştır.

Alan yazında bilgisayar programlama ve bilgi işlemsel düşünme (BİD) konularında birçok çalışma yer almaktadır. Programlama eğitiminin önemi dünya genelinde kabul görmekte olup (Belmar, 2022), BİD'nin soyutlama, ayrıştırma, algoritma tasarımı ve genelleme gibi temel bileşenlere dayandığı vurgulanmaktadır (Chakraborty, 2024). Yapılan deneysel çalışmalar (Alsancak Sırakaya, 2019; Özmen Yağız ve Usluel, 2024), programlama öğretiminin özellikle de blok temelli programlama ve bilgisayar oyunu temelli yaklaşımların öğrencilerin bilgi işlemsel düşünme becerilerini anlamlı düzeyde geliştirdiğini göstermiştir. Bu bulgular, meta-analiz çalışmaları (Üzüm vd., 2024) tarafından da desteklenerek, programlama eğitiminin temel amacının sadece kodlama becerisi değil, aynı zamanda BİD'nin temel bileşenlerini güçlendirmek olduğunu kanıtlamaktadır. Ayrıca, BİD becerilerinin programlama kalitesini artırdığı (Boom vd., 2022) ve çeşitli faktörlerden etkilendiği (Herlambang ve Rachmadi, 2024) belirlenmiştir. Uluslararası düzeyde, programlamanın teknoloji ve matematik derslerine entegrasyonu (Hallström ve Vries, 2023) gibi müfredat çalışmaları ile BİD'nin yaygınlaştırılması hedeflenmektedir. Bu kapsamda, uygulamalı BİD becerilerinin gelişimine yönelik yapılan tüm bu çalışmalar, programlama eğitiminin mühendislik öğrencileri için olan kritik önemini teorik çerçeveden desteklemektedir.

Programlama öz yeterliliği alanındaki çalışmalar, bu algının düzeyini, belirleyici faktörlerini ve programlamaya yönelik tutumla olan güçlü ilişkisini odak noktasına almaktadır. Yapılan araştırmalar (Hanjani, 2019; Gezgin ve Adnan, 2016), mühendislik öğrencilerinin öz yeterlilik algılarının orta düzeyde olduğunu göstermiş, ancak bu algıların bölüm, cinsiyet ve programlama dili bilgisi gibi demografik ve eğitsel değişkenlere göre farklılaştığı tespit edilmiştir. Özellikle yoğun öğrenme isteği, mesleki fayda inancı, derste yüksek başarı (Benli ve Tek, 2021) ile dil hakimiyeti ve proje yapma motivasyonunun (Aksoğan vd., 2020), yüksek öz yeterlilik algısıyla doğrudan ilişkili olduğu belirlenmiştir. Ayrıca, uygulama ağırlıklı pedagojik yaklaşımların (Lewis vd., 2020) ve Ters-Yüz Sınıf Modeli gibi modern öğretim yöntemlerinin (Uysal ve Ocak, 2023), öğrencilerin öz yeterlilik ve bağlılık düzeylerini olumlu yönde etkilediği görülmüştür. Tutum ve öz yeterlilik arasındaki ilişkiye odaklanan çalışmalar (Özyurt ve Özyurt, 2015; Gürer ve Tokumacı, 2020), bu iki değişken arasında yüksek düzeyde, pozitif ve anlamlı bir ilişki olduğunu ortaya koymuş; tutum ve öz yeterlilik algılarının genellikle erkek öğrenciler ve programlama tecrübesi olanlar lehine anlamlı farklılıklar gösterdiğini saptamıştır.

Programlama öz yeterliliği, öğrencilerin öğrenme performansını etkileyen kritik bir değişken olarak akademik başarı ve genel motivasyonla güçlü bir ilişki sergilemektedir (Benjamin vd., 2023). Öğrencilerin öz yeterlik inançları, pasif bir yapıdan ziyade, pedagojik yaklaşımların ve öğrenme ortamlarının kalitesi tarafından doğrudan şekillendirilmektedir (Liu ve Chen, 2023). Yıldız Durak (2018), dijital hikaye tasarımı gibi yaratıcı, uygulamalı ve bağlamsal öğrenme aktivitelerinin, öğrencilerin programlama öz-yeterliklerini anlamlı ölçüde artırdığını göstermiştir.Öz yeterliliğin gelişimini etkileyen bir diğer önemli değişken ise, akademik

performansı olumsuz etkileyen ve öz yeterliliğin ters bir göstergesi olan programlama kaygısıdır. Bu bağlamda, Keskinkılıç ve Kalelioğlu'nun (2024) çalışmaları, özellikle kadın öğrencilerin kaygı düzeylerinin daha yüksek olduğunu ve bu kaygının sınıf düzeyi ilerledikçe değişebildiğini ortaya koymuştur. Pedagojik yaklaşımların etkinliğine odaklanan araştırmalar (Turan ve Erdoğan, 2025), blok temelli programlama eğitiminin (Scratch) öğretmen adaylarının TPACK gelişimini desteklediğini göstermiş; ancak bu tür yaklaşımların bilgi işlemsel düşünme ve tutumlar üzerindeki etkisinin bağlama ve hedef kitleye göre farklılaşabileceğini ortaya koymuştur. Tüm bu bulgular, programlama eğitiminde başarının artırılması için öz yeterliliği güçlendirici ve kaygıyı azaltıcı çevresel ve motivasyonel faktörlere odaklanmanın kritik önem taşıdığını işaret etmektedir. Mühendislik eğitimi, modern endüstriyel gereklilikler nedeniyle programlama becerilerine kritik bir önem vermektedir. Buna rağmen, öğrencilerin bu temel beceriye yönelik öz-yeterlik algıları üzerine odaklanan çalışmaların sınırlı kaldığı ve algılanan güçlü ve zayıf yönlerin derinlemesine incelenmediği görülmektedir. Programlama öz-yeterliği, öğrencilerin akademik başarısını ve mesleki gelişimini doğrudan etkileyen önemli bir faktör olduğundan, öğrencilerin programlama hakkındaki görüşlerinin ve algı düzeylerinin derinlemesine analiz edilmesi; mevcut programlama derslerinin ve öğretim yöntemlerinin etkinliğini değerlendirmeye ve öğrencilerin ihtiyaçlarına yönelik somut iyileştirme önerileri geliştirmeye olanak sağlayacaktır. Bu nedenle bu çalışma ile mühendislik öğrencilerinin öz-yeterlik algılarını belirleyerek mevcut durumu bilimsel verilerle ortaya koymayı ve bu veriler ışığında eğitim uygulamalarına yönelik pratik öneriler sunmak amaçlamıştır..

Araştırmanın Amacı

Alanyazında öğrencilerin programlama konusundaki öz-yeterliklerinin düşük olması ve daha en başından zor olarak algıladıkları için bu derste başarısız olabileceklerini belirten çalışmalar mevcuttur. (Abdüsselam vd., 2021; Askar ve Davenport, 2009; Mazman ve Altun, 2013). Programlama üzerine yapılan çalışmalar çoğunlukla programlama öğrenme süreci, öğrencinin kavram yanılgıları, öğrenci özellikleri ve acemi uzman programcı gibi bilişsel boyutlarla ilgilidir (Cetin ve Ozden, 2015). Öğrencilerin öz-yeterlikleri ile ilgili çalışmalar sınırlıdır. Bu çalışmanın amacı, mühendislik öğrencilerinin bilgisayar programlamaya yönelik öz-yeterlikleri ve bilgisayar programlamaya yönelik görüşlerinin incelenmesidir. Ayrıca mühendislik öğrencilerinin çeşitli değişkenlere göre programlamaya yönelik öz-yeterliklerinin belirlenmesi amaçlanmıştır. Bu amaca ulaşmak için aşağıdaki problemlere cevap aranmıştır.

1. Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri ne düzeydedir?
2. Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri kontrol, mantıksal düşünme, hata ayıklama, algoritma ve işbirliği boyutlarına göre ne düzeydedir?
3. Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri cinsiyet, bölüm, almış oldukları ders sayısı, bilgisayar sahibi olma gibi değişkenlere göre anlamlı farklılık göstermekte midir?
4. Mühendislik öğrencilerinin programlama dersi hakkındaki görüşleri nelerdir?

YÖNTEM

Çalışmada araştırma deseni olarak Karma Yöntem Araştırması yaklaşımı benimsenmiştir; bu model, tek bir çalışma içerisinde nicel ve nitel araştırma verilerinin toplanmasını, analiz edilmesini ve yorumlanmasını içerir (Leech ve Onwuegbuzie, 2009). Bu kapsamda, karma yöntem desenlerinden Açıklayıcı sıralı desen kullanılmış olup, Creswell ve Plano Clark'a (2023) göre bu desen, araştırmacının öncelikle nicel verileri toplayıp analiz ettiği, ardından elde edilen nicel bulguları daha derinlemesine açıklamak ve yorumlamak amacıyla nitel verileri topladığı iki aşamalı bir yaklaşımdır. Buna uygun olarak uygulama süreci, öncelikle öğrencilerden geniş kapsamlı nicel verilerin (N=355) toplanıp analiz edilmesiyle başlamış, ardından elde edilen nicel bulguları anlamlandırmak amacıyla belirlenen 12 kişilik bir grupla nitel görüşmeler yapılarak ikincil verilerin toplanmasıyla devam etmiştir; bu tasarım nicel sonuçları nitel bulgular aracılığıyla destekleyip detaylandırmayı amaçlamaktadır. Son olarak, araştırmada bağımsız değişken olarak

demografik değişkenler (cinsiyet, bölüm, programlama üzerine alınan ders sayısı) kullanılırken, bağımlı değişkenler ise bilgi işlemsel düşünme becerileri ve programlama öz-yeterlikleridir.

Örneklem

Çalışmanın örneklemini, Türkiye'deki bir devlet üniversitesinin mühendislik fakültesi öğrencilerinden uygun örnekleme yöntemiyle seçilen 355 öğrenci oluşturmaktadır. Uygun örnekleme yöntemi, seçkisiz olamayan örnekleme yöntemlerinden biri olup, zaman, para ve iş gücü gibi zorlukların ortaya çıkması ihtimali nedeniyle örneklemin kolay ulaşılabilir ve uygulama yapılabilir bir gruptan seçilmesini amaçlamaktır (Büyüköztürk vd., 2002). Çalışmanın nicel boyutuna katılan öğrencilerin genel demografik bilgileri Tablo 1'de sunulmuştur.

Tablo 1. Tüm örneklemin demografik bilgileri

Kategori	Alt Kategori	Kız	Erkek	Toplam
Bölüm	Bilgisayar Mühendisliği	29	32	61
	Makine Mühendisliği	27	131	158
	Endüstri Mühendisliği	15	31	46
	Elektrik-Elektronik Müh.	5	37	42
	İnşaat Mühendisliği	5	43	48
Yaş	18	3	9	12
	19	16	39	55
	20	24	46	70
	21	15	71	86
	22	11	56	67
	23	10	27	37
	24+	2	26	28
Bilgisayar sahipliği	Evet	78	258	336
	Hayır	3	16	19
Programlama dersi sayısı	1	35	153	188
	2	18	59	77
	3	5	11	16
	4+	23	51	74
Toplam		81	274	355

Tablo 1'e göre çalışmaya katılan öğrencilerin büyük çoğunluğu erkektir (n=274), kız öğrenci sayısı ise 81'dir. Katılımcıların en yoğun bulunduğu bölüm Makine Mühendisliği; yaş dağılımında ise 21 yaş grubu öne çıkmaktadır. Öğrencilerin büyük çoğunluğu bilgisayara sahiptir (n=336) ve genellikle 1 veya 2 programlama (n=188 ve n=77) dersi almışlardır. Nitel veri toplama sürecinde ise 12 öğrenci ile yarı yapılandırılmış görüşmeler gerçekleştirilmiştir.

Nitel verilerin toplanmasında ise 12 öğrenci ile derinlemesine görüşme yapılmış olup bu öğrencilere ait demografik bilgiler Tablo 2'de ayrıntılı olarak yer almaktadır.

Tablo 2. Görüşülen öğrenci grubunun demografik bilgileri

Bölüm	Kız	Erkek	Kod
Elektrik- Elektronik Mühendisliği	-	3	Ö1, Ö2, Ö3
Endüstri Mühendisliği	1	2	Ö4, Ö5, Ö6
Makine Mühendisliği	-	3	Ö7, Ö8, Ö9
Bilgisayar Mühendisliği	1	2	Ö10, Ö11, Ö12
Toplam	2	10	

Verilen tabloya göre, nitel görüşme örneklemini oluşturan 12 öğrencinin büyük çoğunluğu (%83.3), dört farklı mühendislik bölümüne (Elektrik-Elektronik, Endüstri, Makine ve Bilgisayar) dengeli dağılmış erkek öğrencilerden oluşmaktadır (2 kız ve 10 erkek).

Veri Toplama Araçları

Çalışmada veri toplama aracı üç bölümden oluşmaktadır. İlk bölümde öğrencilerin cinsiyet, bölüm, bilgisayar sahibi olma ve aldıkları programlama dersi sayısı gibi demografik bilgileri yer almaktadır. İkinci bölümde, mühendislik öğrencilerinin bilgisayar programlamaya yönelik öz-yeterliklerini belirlemek amacıyla, Tsai, Wang ve Hsu (2019) tarafından geliştirilen ve Ekici ve Çınar (2020) tarafından Türkçeye uyarlanan Bilgisayar Programlama Öz-Yeterlik Ölçeği kullanılmıştır. Ölçeğin Türkçe uyarlayıcısından kullanım izni alınmıştır. Ölçek, 16 maddeden oluşan ve 6'lı Likert tipidir. Ölçeğin güvenirliğinin belirlenmesi amacıyla iç tutarlık katsayısı Cronbach'ın alfa değeri, McDonald'ın Omega değeri ile Bileşik Güvenirlik (Composite Reliability, CR) değerleri ve madde toplam korelasyonu hesaplanmıştır buna göre ölçeğin güvenirlik katsayıları yüksek düzeydedir (α = .911; ω = .917; CR = .962). Ölçek; kontrol, mantıksal düşünme, hata ayıklama, algoritma ve iş birliği olmak üzere beş alt boyuttan oluşmaktadır. Ölçekten alınabilecek en yüksek puan 96 en düşük puan ise 16'dır. Ölçeğin faktör dağılımı Tablo 3'te verilmiştir.

Tablo 3. Bilgisayar Programalama Öz-Yeterlik Ölçeği Faktör dağılımı

Faktör (Boyut) Adı	Ölçek Maddeleri
Kontrol	1,2,3
Mantıksal Düşünme	4,5,6,7
Hata Ayıklama	8,9,10
Algoritma	11,12,13
İş birliği	14,15,16

Mühendislik öğrencilerinin Programlama dersi hakkındaki görüşlerini almak için araştırmacılar tarafından görüşme soruları oluşturulmuştur ve bir görüşme formu olarak düzenlenmiştir. Görüşme formunun içerik geçerliliğini sağlamak amacıyla, form alanında uzman üç öğretim üyesinin görüşüne sunulmuştur. Uzmanlardan gelen öneriler doğrultusunda formda dil, anlaşılırlık ve kapsam açısından gerekli düzenlemeler yapılmıştır; özellikle bazı ifadeler sadeleştirilmiştir. Örneğin, derleyiciler ile ilgili görüşme soruları uzman görüşleri doğrultusunda kaldırılmıştır. Uzman görüşleri sonrasında düzenlenen form, pilot çalışma ile üç öğrenci üzerinde uygulanmıştır. Pilot çalışma bulgularına göre, katılımcılar tarafından yanlış anlaşıldığı tespit edilen bir soru yeniden ifade edilerek iç tutarlılık ve anlaşılırlık en üst düzeye çıkarılmıştır. Bu süreçler sonucunda görüşme formu, 10 ana soru ve 1 sonda sorudan oluşan nihai yapısını almıştır.

Veri Analizi

Çalışma karma araştırma deseninde gerçekleştirilmiştir. Bu nedenle hem nicel hem de nitel verilerin analizleri yapılmıştır. Çalışmanın nicel veri analizi kısmında SPSS istatistik programı kullanılmıştır. Analizlere başlamadan önce veri setinin normal dağılıp dağılmadığı kontrol edilmiştir. Veri setinin normal dağılımı için, çarpıklık ve basıklık katsayıları Tablo 4'te verilmiştir.

Tablo 4. Basıklık ve çarpıklık katsayıları

			Çarpıklık		Basıklık	
		N	Değer	Standart Hata	Değer	Standart Hata
Demografik Bilgiler	Cinsiyet	355	-1.301	.129	-.309	.258
	Yaş		2.431		11.808	
	Bölüm		.105		-.628	
	Bilgisayar var olma durumu		3.984		13.953	
	Programlama dersi alma durumu		.886		-.829	
Bilgisayar programlama öz-yeterlik ölçeği	Kontrol Faktörü		-.671		.064	
	Mantıksal Düşünme Faktörü		-.518		-.555	
	Hata Ayıklama		-.756		.183	
	Algoritma		-.475		-.219	
	İş birliği		-.117		-.536	
	Toplam		-.933		.537	

Tablo 4'te verilerin normal dağılıp dağılmadığı çarpıklık, basıklık değerleri ve Q-Q Plot grafikleri ile incelenmiştir. Can (2018)'a göre bu değerler dağılımın normalliğinin kabul edilebilir olduğunu göstermektedir. Elde edilen değerler normal dağılımı işaret ettiğinden parametrik testler kullanılmıştır.

Çalışmanın nitel verileri yarı yapılandırılmış görüşmelere ait ses kayıtları alınmış ve yazılı transkriptleri çıkartılmıştır. Verilerde gereksiz metinler çıkartılmış ve hem betimsel hem içerik analizine tabii tutulmuştur. Betimsel analiz, nitel verilerin daha önceden belirlenmiş temalara göre sınıflandırılıp, özetlenip, yorumlanmış bir şekilde okuyucuya raporlanması (Yıldırım ve Şimşek, 1999); içerik analizi ise nitel verileri açıklamaya yardımcı olabilecek kavram ve ilişkilerin kurulmasıdır. Veriler, kategorilere; kategoriler uygun temalara, temalarda anlam bütünlüğünü göre kodlara ayrılmıştır. Alanda yapılan benzer çalışmalar incelenerek kategoriler ve temalar kontrol edilmiş ve çalışmada kullanılan son halini almıştır.

Geçerlik ve Güvenirlik

Çalışmada nicel verilerin analizlerinin geçerlilik ve güvenirliği için istatistiki yöntemler kullanılmış ayrıca alanda geçerliği ve güvenirliği önceden tespit edilmiş standart bir ölçek tercih edilmiştir. Çalışmada Ekici ve Çınar (2020) tarafından Türkçe diline uyarlanan Bilgisayar Programlama Öz-Yeterlik Ölçeği kullanılmış ve ölçeğin güvenirlik katsayısı, Cronbach'ın alfa katsayısı α =0.911 olarak hesaplanmıştır. Bu çalışma özelinde ise hesaplanan güvenirlik katsayısı, Cronbach'ın alfa katsayısı α =0.962'dir. Bu değer çalışmanın yüksek derecede güvenilir olduğunu şeklinde yorumlanabilir (Can, 2018; s 391).

Etik Bilgisi: Araştırma sürecine başlamadan önce çalışmanın yürütülmesi için ilgili üniversitenin Fen ve Mühendislik Bilimleri Etik Komisyonundan 30.11.2022 tarih ve E-19928322-302.08.01-203249 sayı ile onay alınmıştır. Ayrıca, çalışmaya katılan tüm öğrencilere araştırmanın amacı hakkında bilgi verilmiş ve gönüllü katılım ilkesine uyularak bilgilendirilmiş onam formu alınmıştır. Araştırma yayın ve etiğine uyulmuştur.

BULGULAR

Çalışmanın birinci problemi "Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri ne düzeydedir?" sorusu ile ilgili bulgular Tablo 5'te verilmiştir.

Tablo 5. Bilgisayar programlama öz yeterlilik ölçeği toplam puanları betimsel analiz

	N	En düşük puan	En yüksek puan	Ortalama	Standart Sapma
Toplam	355	16	96	63.86	17.8

Mevcut çalışma sonucunda ise öğrencilerin ölçekten aldığı puan ortalaması $\bar{X}$ = 63.86'dır. Bu durumda mühendislik fakültesi öğrencilerinin bilgisayar programlamaya yönelik öz yeterliliklerinin orta düzeyde olduğu söylenebilir. En düşük puanı (16) alan n=5 kişi; en yüksek puanı (96) alan n=7 kişi olduğu görülmektedir. Ayrıca ölçekten 64 puan alanların sayısının n=18 kişi olduğu ve en fazla puan yığılımının burada olduğu görülmektedir.

Çalışmanın ikinci problemi "Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri kontrol, mantıksal düşünme, hata ayıklama, algoritma ve iş birliği boyutlarına göre ne düzeydedir?" sorusu ile ilgili bulgular Tablo 6'da verilmiştir.

Tablo 6. Bilgisayar programlama öz yeterlilik ölçeği alt boyutlarına ilişkin betimsel analiz

Faktör Adı	En düşük puan	En yüksek puan	Ortalama	Madde Ortalama puanı	Standart Sapma
Kontrol	3	18	12.50	4.16	4.24
Mantıksal düşünme	4	24	16.69	4.17	4.86
Hata ayıklama	3	18	11.27	3.76	3.63
Algoritma	3	18	10.40	3.47	3.66
İş birliği	3	18	13.00	4.33	3.58

Tablo 6 incelendiğinde, öğrencilerin bilgisayar programlama öz yeterlilikleri ölçeği alt boyutları puan ortalamaları şöyledir: kontrol alt boyutu puan ortalaması $\bar{X}$ =12.5; mantıksal düşünme alt boyutunda puan ortalaması $\bar{X}$ =16.69; hata ayıklama alt boyutu puan ortalaması $\bar{X}$=11.27, algoritma alt boyutunda puan ortalaması $\bar{X}$ = 10.40 ve işbirliği alt boyutunda aldığı puan ortalamaları $\bar{X}$ =13.00. Alt faktörler açısından madde ortalama puanının 3.5 üzeri olması ilgili bileşene ilişkin yüksek yetkinlik algısına işaret etmektedir (Tsai vd., 2019; Ekici ve Çınar, 2020). Bu durum bize öğrencilerin bilgisayar programlama öz yeterliliklerinin, kontrol, mantıksal düşünme, hata ayıklama ve iş birliği boyutlarında oldukça yüksek olduğunu; algoritma alt boyutunda ise yükseğe yakın bir yetkinlikte olduklarını göstermektedir. Ölçek alt boyutlarına göre öğrencilerin en yüksek yetkinlikleri ise iş birliği alanındadır.

Çalışmanın üçüncü problemi "Mühendislik öğrencilerinin programlamaya yönelik öz-yeterlikleri cinsiyet, bölüm, almış oldukları ders sayısı, bilgisayar sahibi olma gibi değişkenlere göre anlamlı farklılık göstermekte midir?" sorusu ile ilgili bulgular sırasıyla sunulmuştur. Mühendislik öğrencilerinin programlamaya yönelik öz yeterliklerinin cinsiyete göre değişkenlik gösterip göstermediği incelenmiştir. Çalışmada, grup verileri normal dağılım gösterdiği için parametrik testlerden bağımsız örneklem t-testi yapılmıştır. Grubun varyans homojenliğinin sağlanıp sağlanmadığının kontrol edilmesi amacıyla Levene testi yapılmıştır. Testin sonucunda grubun homojen olarak dağıldığı sonucuna ulaşılmıştır (Levene değeri =2.94; p= .088: $p<0.05$). Bulgular Tablo 7'deki gibidir.

Tablo 7. Bilgisayar programlama öz yeterlilik ölçeği t-testi sonuçları

Cinsiyet	N	Ortalama	Standart Sapma	Sd	T	p
Kadın	81	65.12	16.44	353	.725	.469
Erkek	274	63.49	18.16			

p>0.05

Bulgulara göre kadınların bilgisayar programlama öz yeterlilik ölçeği puan ortalamaları $\bar{X}$ = 65.12; erkeklerin bilgisayar programlama öz yeterlilik ölçeği puan ortalamaları $\bar{X}$ = 63.49 arasında anlamlı bir fark görülmemiştir [t (353) = .725, p>0.05].

Mühendislik fakültesinin farklı bölümlerindeki öğrencilerin bilgisayar programlama öz-yeterlilik puan ortalamaları ANOVA testi ile karşılaştırılmıştır. Gruplar arasındaki farkın belirlenmesi için çoklu karşılaştırma testleri uygulanmış, öncesinde varyansların homojenliği Levene testi ile incelenmiştir. Levene testi anlamlı çıkmış (p = .00 <.05), yani varyanslar eşit olmadığından Tamhane's T2 testi tercih edilmiştir. Bu test, grup büyüklüklerinin eşit olmaması durumunda daha güvenilir sonuçlar verdiği ve istatistiksel gücünün yüksek olduğu için seçilmiştir (Kayri, 2009; Shingala & Rajyaguru, 2015; Taşpınar, 2017; Güçlü, 2020). Bu nedenle çalışmada Tamhane's T2 tercih edilmiştir. Mühendislik bölümleri arasındaki programlama özyeterlilik test puanları Tablo 8'de raporlanmıştır.

Tablo 8. Tamhane's T2 testi ile mühendislik bölümlerinin programlama öz yeterlilik puanlarındaki farklar

Sınıf Düzeyi (I)	Sınıf Düzeyi (J)	Ortalama Farkı (I-J)	Standart hata	P
Bilgisayar Mühendisliği	Makine Mühendisliği	17.430	3.315	.000
	Endüstri Mühendisliği	12.687	1.495	.000
	Elektrik-Elektronik Mühendisliği	16.611	3.172	.000
	İnşaat Mühendisliği	30.492	3.465	.000
Makine Mühendisliği	İnşaat Mühendisliği	17.805	3.472	.000
Elektrik-Elektronik Mühendisliği	İnşaat Mühendisliği	13.881	4.459	.026

Tablo 8 incelendiğinde farklı mühendislik bölümlerinin, programlama öz-yeterlilikleri ölçeğinden aldıkları puanların farklılaştığı görülmektedir. Bilgisayar mühendisliği öğrencileri, örneklem grubunda yer alan diğer tüm mühendislik bölümleri öğrencilerine göre programlama öz-yeterlilik ölçeği puanları farklılaşmaktadır. Öte yandan programlama öz-yeterlilikleri ölçeği puanları farklılık gösteren diğer gruplar ise inşaat mühendisliği öğrencileri ile makine (I-J = 17.805) ve elektrik-elektronik mühendisliği (I-J= 13.881) öğrencileridir.

Mühendislik fakültesi öğrencilerinin bilgisayarlarının olup olmama durumuna göre bilgisayar programlama öz yeterlilik ölçeği puanları normal dağılım gösterdiği için, t- testi ile analiz edilmiştir. Ardından varyans homojenliğinin sağlanıp sağlanmadığının kontrol edilmesi amacıyla yapılan Levene testi yapılmış ve testin sonucunda grubun homojen olmadığı sonucuna ulaşılmıştır (Levene değeri =4.17; p= .042: p<0.05). Analiz sonuçları Tablo 9'da verilmiştir.

Tablo 9. Bilgisayar programlama öz yeterlilik ölçeği bilgisayar durumuna göre t testi sonuçları

Bilgisayar sahip olma	N	Ortalama	Standart Sapma	Sd	T	P
Evet	336	64.61	17.22	19.22	2.66	.015
Hayır	19	50.74	22.36			

p< 0.05

Bulgulara göre bilgisayara sahip olan mühendislik öğrencilerinin, bilgisayarı olmayanlara göre bilgisayar programlama öz yeterlilik ölçeği puanlarında istatistiksel olarak anlamlı bir fark olduğu görülmektedir [t (19,22) =2.66, p<0.05]. Mühendislik alanlarında öğrenim gören öğrencilerden bilgisayara sahip olanların programlama öz-yeterliliklerinin, sahip olmayanlara göre daha yüksek olduğu söylenebilir.

Son olarak mühendislik öğrencilerinin programlamaya yönelik öz-yeterliklerinin aldıkları ders sayısına göre anlamlı şekilde farklılaşıp farklılaşmadığı incelenmiştir. Verilerin normal dağılım göstermesi üzerine ANOVA testi uygulanmış, varyans homojenliği ise Levene testiyle değerlendirilmiştir. Levene testi anlamlı çıktığı (F = 8.51; p = .00 <.05) için varyanslar eşit kabul edilmemiş ve çoklu karşılaştırmalarda Tamhane's T2 testi (Tablo 10) kullanılmıştır.

Tablo 10. Tamhane's T2 testi ile mühendislik öğrencilerinin ders sayısı ve programlama öz yeterlilik puanları

Alınan Ders Sayısı (I)	Alınan Ders Sayısı (J)	Ortalama Farkı (I-J)	Standart Hata	p
1 ders alan	2	-9.403	2.092	.000
	4 ve daha fazla	-19.360	1.858	.000
2 ders alan	4 ve daha fazla	-9.957	2.066	.000
3 ders alan	4 ve daha fazla	-13.642	4.366	.035
4 ve daha fazla	1 ders	19.360	1.858	.000
	2 ders	9.957	2.066	.000
	3 ders	13.642	4.366	.035

Tablo 10'da 4 veya daha fazla sayıda ders alan öğrencilerin programlama öz-yeterlilik ölçeğinden aldıkları puanlar özellikle 1 ders alan öğrencilerden daha farklı olduğu sonucuna ulaşılabilir. Çalışmanın dördüncü problemi "Mühendislik öğrencilerinin programlama dersi hakkındaki görüşleri nelerdir?" sorusu ile ilgili bulgular sırasıyla sunulmuştur. Çalışmada mühendislik fakültesi öğrencilerinin programlama dersi hakkındaki görüşleri alınmış ve çeşitli değişkenler açısından raporlandırılmıştır. Çalışmanın nitel kısmına katılan örneklem grubun demografik yapısı örneklemin açıklandığı Tablo 2'de verilmişti.

Görüşmeye katılan katılımcıların cevapları ve alan taraması sonucu programlama dersi ile ilgili çalışmada kullanılan kategoriler ve kodlar Tablo 11'te verilmiştir.

Tablo 11. Katılımcıların programlama dersine ilişkin görüşlerine ait kategoriler ve kodlar

Kategoriler	Kodlar
Programlama Temelleri	Programın yapısını anlama, Programlamanın temeli, Program akışı, Mantıksal Operatörler, Karşılaştırma işlemleri, Matematiksel işlemler
Hata yönetimi ve Verimlilik	Kod hatalarını azaltma, Hataları tespit etme, Zaman kazanma, Vazgeçilmez programlama süreci, Performans sorunları, Yapısal problemleri çözmemesi, Hata bulmada zorlanma
İş birliği ve Proje Yönetimi	Farklı bakış açıları, İşbirliğinin zorunluluğu, İş birliği ve hata yakalama, Zaman ve Verimlilik, Proje yönetimi zorlukları, Ekip çalışması
Problem Çözme ve Hesaplama	Problem çözme, Verimliliği artırma, Hesaplamalar ve modelleme, Makinelerle iletişim kurma, Algoritma tasarımında zorlanma, Algoritma önemi
Öğrenme Süreci ve Zorluklar	Analitik ve soyut düşünme, Programlama dönüşüm zorluğu, Mantık ve sentaks hataları, Belirsiz öğrenim içeriği, Dil söz dizim farkı, İngilizce eksikliği, Kütüphane yönetimi süreci, İlgi ve hevesin kaybolması, Ezbere dayalı öğrenme
Gelişim ve Yenilikçilik	Hızlı ve verimli süreç, İnovatif çözümler, Sistematik ürün gelişimi, Sürekli gelişen alan, Analitik düşünme gelişimi
Öğrenme Yöntemleri ve Motivasyon	Uygulamalı öğrenme, Kaynak kullanımı, Bağlamsal öğrenme, Motivasyon ve merak

Çalışmaya katılan mühendislik fakültesi öğrencilerinin kullandıkları programlama dilleri Tablo 12'de verilmiştir.

Tablo 12. Katılımcıların kullandıkları programlama dilleri

	C	C++	C#	Python	Java	PHP	Matlab	ASP.NET	Html/CSS	SQL
Ö1	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐
Ö2	☒	☒	☒	☒	☐	☐	☒	☐	☐	☐
Ö3	☒	☐	☐	☐	☒	☐	☐	☐	☒	☐
Ö4	☐	☐	☐	☒	☐	☐	☒	☐	☐	☒
Ö5	☐	☒	☐	☒	☐	☐	☐	☐	☐	☐
Ö6	☒	☒	☐	☒	☐	☐	☐	☐	☐	☐
Ö7	☒	☐	☒	☐	☐	☐	☐	☐	☐	☐
Ö8	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐
Ö9	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐
Ö10	☒	☒	☒	☒	☐	☒	☒	☒	☒	☒
Ö11	☐	☐	☒	☐	☐	☒	☐	☐	☐	☒
Ö12	☒	☒	☒	☒	☒	☒	☐	☐	☒	☒
Frekans	7	7	5	5	2	3	3	1	2	4

Tablo 12 incelendiğinde, çalışmaya katılan mühendislik fakültesi öğrencilerinin programlama dillerinden en fazla C ve C++ (n=7) dillerini kullanabildikleri; bir öğrencinin ise ASP.NET programını kullanabildiği belirlenmiştir.

Çalışmada katılımcıların programlama dillerini ne düzeyde kullandıklarına dair de veriler toplanmıştır. Katılımcıların belirttikleri programları kullanama düzeyleri Tablo 13'te gösterilmiştir.

Tablo 13. Katılımcıların belirttikleri programlama dilleri kullanma düzeyleri

	Frekans	Katılımcılar
Başlangıç	2	Ö2, Ö9
Orta	7	Ö1, Ö3, Ö4, Ö5, Ö6, Ö7, Ö8
İleri düzey	3	Ö10, Ö11, Ö12

Tablo 13 incelendiğinde belirttikleri programlama dillerini çoğunlukla orta düzeyde (n=7) kullandıklarını belirtmişlerdir. Katılımcıların, "Algoritma tasarlamanın programlamaya açısından önemi hakkında görüşlerinizden bahseder misiniz? Programlamaya başlamadan önce algoritma tasarımı yapıyor musunuz? Neden?" sorusuna yönelik görüşleri alınmıştır. Bu görüşlere ilişkin kategori ve kodlar Tablo 14'te verilmiştir.

Tablo 14. Katılımcıların algoritma tasarlama süreci kategori ve kodları

Kategoriler	Kodlar	Frekans	Katılımcı
Programlama Temelleri	Programın yapısını anlama	5	Ö1, Ö3, Ö4, Ö6, Ö10
	Programlamanın temeli	4	Ö3, Ö6, Ö7, Ö8
Hata Yönetimi ve Verimlilik	Verimlilik	2	Ö9, Ö12
	Kod hatalarını azaltma	4	Ö3, Ö5, Ö11

Tablo 14 incelendiğinde katılımcıların programlama açısından algoritma tasarımı ile ilgili görüşleri, programlama temelleri ve hata yönetimi ve verimlilik olmak üzere iki kategoride toplanmıştır. Algoritma tasarlama süreci hakkında katılımcıların kodlara göre görüşleri: Ö4 [Programın yapısını anlama] kodu ile "Algoritmalar, belirli bir problemin çözümü için adım adım takip edilmesi gereken bir yol haritası sağlar..." Ö5 [Kod hatalarını azaltma] kodu ile "Algoritma tasarlamak programı yazarken işlemleri kısaltmayı sağladığı ve kod hatalarını azalttığı için önemlidir." Ö9 [Verimlilik] kodu ile "Algoritma tasarımı programlama da problemin çözümü için oldukça önemlidir. Algoritma tasarımı problemin kesin çözümü için yapılması gerekir." Ö3 [Programlamanın temeli] kodu ile "…Hatta algoritma tasarlamanın program yazmanın asıl kısmı bile olabilir." Şeklinde görüş belirtmişlerdir. Öğrenciler, algoritma bilgisinin programlamanın temeli olduğunu ve yazılım geliştirme sürecini daha verimli hâle getirdiğini belirtmektedir. Ayrıca, algoritmaların problem çözmede yol haritası sunduğu ve hem anlık çözümler hem de gelecekteki değişiklikler için esneklik sağladığı ifade edilmiştir. Bu durum, algoritma bilgisinin programlama öz-yeterliği açısından önemli bir unsur olduğunu göstermektedir.

Katılımcılara "Programlama yaparken mantıksal işlemler denildiğinde ne anlarsınız? Bunların önemi nedir?" sorusu sorulmuştur. Katılımcıların programlarda mantıksal işlemler ile ilgili görüşlerine ilişkin kategoriler ve kodlar Tablo 15'te verilmiştir.

Tablo 15. Programlamada mantıksal işlemler ile ilgili görüşlerin kategori ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
Programlama Temelleri	Mantıksal Operatörler	5	Ö1, Ö3, Ö5, Ö10, Ö12
	Karşılaştırma işlemleri	3	Ö3, Ö4, Ö9
	Matematiksel işlemler	3	Ö2, Ö6, Ö11
	Program akışı	2	Ö7, Ö8
	Programlama için önemi	2	Ö3, Ö11

Katılımcıların programlama yaparken mantıksal işlemlere yönelik görüşleri programlama temelleri kategorisinde çeşitli kodlar altında toplanmıştır. Katılımcıların mantıksal işlemler ile ilgili görüşleri şöyledir: Ö3 [Mantıksal operatörler ve Matematiksel işlemler] kodları ile "…En temel anlamda And, Or, Not ve Exor işlemlerinin çeşitli kombinasyonlar ile birleşmeleri ile oluşan mantıksal karşılaştırmalardır.."; Ö11 [Matematiksel işlemler] kodu ile "…Sonuçta makine dili dediğimiz yapı tamamen matematiğe dayanır."; Ö8 [Program akışı] kodu ile "Girdiğimiz komutları mantıklı ve zamandan tasarruf edebilecek şekilde sırada olması." ; Ö3 [Programlama için önemi] kodu ile "…Programlama açısından önemine bakıldığında önemli şeyler mantıksal işlemlerden sonra başlar…Mantıksal işlemleri programlamadan çıkardığımızda geriye hiçbir şey kalmaz. Çünkü, tüm dijital evrenin yapıtaşı mantık devreleridir." Şeklinde görüş belirtmişlerdir. Öğrenciler, mantıksal işlemleri temel karşılaştırmalar, matematiksel işlemler ve program akışının düzenlenmesi olarak tanımlamış; bu yapıların programlama için kritik olduğunu belirtmiştir. Bu konudaki öğrenci farkındalığı, programlama eğitiminin temel kavramlarının öğrencilere iyi öğretildiğini göstermektedir.

Katılımcıların, "Programlama yaparken "hata ayıklama" işlemi hakkında neler düşünüyorsunuz? Hata ayıklama işleminin olumlu veya olumsuz yönlerinden bahseder misiniz?" sorusu yöneltilmiş, bu konuda katılımcıların görüşlerine ilişkin kategori ve kodlar Tablo 16'da verilmiştir.

Tablo 16. Programlamada hata ayıklama işlemi ile ilgili görüşlerin kategori ve kodları

Kategoriler	Kodlar	Frekans	Katılımcı
Hata yönetimi ve Verimlilik	Hataları tespit etme	4	Ö1, Ö6, Ö7, Ö9
	Zaman kazanma	3	Ö2, Ö8, Ö10
	Vazgeçilmez programlama süreci	3	Ö4, Ö11, Ö12
	Performans sorunları	2	Ö2, Ö5
	Yapısal problemleri çözmemesi	1	Ö3

Katılımcıların görüşleri hata yönetimi ve verimlilik kategorisi altında beş kod altında toplanmıştır. Katılımcıların kodlara göre görüşleri şu şekildedir: Ö9 [Hataları tespit etme] kodu ile "…Hata ayıklama işlemi programlama yaparken yazdığımız kodlardaki hataları bulma ve düzeltme işlemidir." Ö8 [Zaman kazanma] kodu ile "Hata ayıklama zamandan ve enerjimizden tasarruf etmemizi sağlıyor…" Ö4 [Vazgeçilmez programlama süreci] kodu ile "Programlamada hata ayıklama, yazılımın doğru şekilde çalışmayan bölümlerini tespit edip düzeltme sürecidir." Ö5 [Performans sorunları] kodu ile "Küçük veri setlerinde yeterince veri olmaması durumunda sorun yaratabilir…" Ö3 [Yapısal problemleri çözememesi] kodu ile "Hata ayıklama işlemleri yazım hataları gibi hatalarda yardımcı olabilirken yapısal problemlere çözüm üretemezler." Şeklinde görüş belirtmişlerdir. Öğrenciler, hata ayıklamanın kod hatalarını bulma, kaynakları verimli

kullanma ve yazılımın doğruluğunu sağlama açısından önemli olduğunu belirtmiş; bazıları ise bu sürecin yapısal sorunlarda yetersiz kalabileceğini ve küçük veri setlerinde performans sorunları yaratabileceğini ifade etmiştir.

Katılımcılara, "Programlama yaparken iş birliği yapmanın önemi nedir? Olumlu ve olumsuz yönleri neler olabilir?" sorusu yöneltilmiş ve görüşlerine ilişkin kategori ve kodlar Tablo 17'de sınıflandırılmıştır.

Tablo 17. Programlamada iş birliğinin önemi görüşlerine ilişkin kategori ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
İş birliği ve Proje Yönetimi	Farklı bakış açıları	6	Ö1, Ö2, Ö6, Ö8, Ö9, Ö10
	İşbirliğinin zorunluluğu	4	Ö3, Ö4, Ö11, Ö12
	İş birliği ve hata yakalama	1	Ö7
	Proje yönetimi zorlukları	4	Ö5, Ö9, Ö10, Ö12
Hata yönetimi ve Verimlilik	Zaman ve Verimlilik	1	Ö10

Katılımcıların programlama yaparken iş birliğinin önemine ilişkin görüşleri iki temel kategori ve beş kod altında (Tablo 17) toplanmıştır. Katılımcıların kategoriler ve kodlara göre görüşleri şu şekildedir: Ö10 [Farklı bakış açıları - Zaman ve verimlilik] kodları ile "Zamandan tasarruf sağlar.

Hataları bulmamızı kolaylaştırır. Birimizin bilmediğini diğeri bilebilir." Ö12 [İşbirliğinin zorunluluğu] kodu ile "Programlama aslında ekip işidir. Bir yazılımın birçok yönü bulunur." Ö7 [İş birliği ve hata yakalama] kodu ile "Hatayı daha kolay ayıklamak, daha kısa yollar var ise bunları farklı yönlerden görebilmek içindir" Ö5 [Proje yönetimi zorlukları] kodu ile "Kalabalık grupların beraber çalıştığı programlarda bu durum sorun yaratabilmektedir." Şeklinde görüş belirtmişlerdir. Öğrenciler, iş birliğinin hata ayıklamayı kolaylaştırıp süreci hızlandırdığını, ancak büyük projelerde iletişim ve kod uyumu gibi zorluklar yaşanabileceğini belirterek bu sürece dair farkındalıklarını ortaya koymuştur.

Katılımcılar "Programlama eğitiminin mühendislik eğitimi açısından öneminden bahseder misiniz?" sorusuna ilişkin yönelik görüşleri ile ilgili kategoriler ve kodlar raporlanmıştır (Tablo 18).

Tablo 18. Programlama eğitiminin mühendislik açısından önemine ilişkin görüşler ile ilgili kategori ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
Problem Çözme ve Hesaplama	Önemli ve zorunlu	7	Ö2, Ö4, Ö5, Ö7, Ö10, Ö11, Ö12
	Problem çözme	2	Ö1, Ö9
	Makinelerle iletişim kurma	1	Ö8
	Hesaplamalar ve modelleme	2	Ö3, Ö12
Hata yönetimi ve Verimlilik	Verimliliği artırma	3	Ö6, Ö7, Ö9

Programlama eğitiminin mühendislik açısından önemi ile ilgili katılımcılar problem çözme ve hesaplama ile hata yönetimi ve verimlilik kategorilerinde görüşlerini belirtmişlerdir (Tablo 18).

Katılımcıların görüşleri şöyledir: Ö2 [Önemli ve zorunlu] kodu ile "Teknoloji çağında olduğumuz için her mühendisin okuduğu alanla alakalı olarak programlama eğitimi görmesi gerektiğini düşünüyorum ben bu son derece benim açımdan önemli." Ö9 [Problem çözme] kodu ile "…mühendisin sorumluluklarını kolaylaştırmak, çözümleri günümüz şartlarına uyarlamak ve yeni oluşacak problemleri çözmede programlamada aldıkları eğitimler yardımcı olmaktadır." Ö6 [Verimliliği artırma] kodu ile "Özellikle kendi alanım için konuşmak gerekirse; bizim için verimliliği artırmak, karmaşık yapıları daha basit hale indirgemek önemlidir." Ö12 [Hesaplamalar ve modelleme] kodu ile "Bir inşaat mühendisinin yapacağı statik hesabında, bir çevre mühendisinin etki değerlendirmesi yapacağı istatistik hesabında ve yine bir makine mühendisinin bir motoru kontrol etmesinde programlama kullanılabilmektedir." Ö8 [Makinelerle iletişim kurma] kodu ile "Programlama bize makinelerin işleyiş ve çalışma prensiplerini gösterir onlarla tabiri caizse konuşmamızı sağlar bu da bence bir dildir ve bu dili bilen makineleri en iyi şekilde anlayıp kariyerlerinde başarılı olurlar." Şeklinde görüş belirtmişlerdir. Öğrenciler, programlamanın problem çözme, verimliliği artırma, hesaplamalar ve modelleme yapma ve makinelerle iletişim kurma gibi temel mühendislik becerilerini desteklediğini belirtmektedir. Özellikle, programlamanın mühendislik alanındaki çeşitli disiplinlerdeki uygulamalarına dikkat çekmeleri, bu becerinin çok yönlülüğünü ve önemini vurgulamaktadırlar.

Katılımcılara "Programlama eğitiminin ürün geliştirme sürecinizi nasıl etkilediğini düşünüyorsunuz? Açıklayınız." Sorusuna yöneltilmiş ve Tablo 19 elde edilmiştir.

Tablo 19. Programlama eğitiminin ürün geliştirme süreci ile ilgili kategoriler ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
İnovasyon ve Gelişim	Hızlı ve verimli süreç	5	Ö4, Ö6, Ö8, Ö9, Ö12
	İnovatif çözümler	2	Ö4, Ö11
	Sistematik ürün gelişimi	3	Ö2, Ö7, Ö10
	Sürekli gelişen alan	2	Ö11, Ö12
	Analitik düşünme gelişimi	4	Ö1, Ö3, Ö4, Ö5

Katılımcılar programlama eğitiminin ürün geliştirme sürecine etkisi ile ilgili görüşlerini "inovasyon ve gelişim" kategorisinde beş kod altında belirtmişlerdir. Katılımcıların görüşleri kodlara göre şu şekildedir: Ö4 [Hızlı ve verimli süreç – Analitik düşünme gelişimi] kodları ile "Programlama eğitimi, yazılımcıların ürün geliştirme sürecini daha hızlı ve verimli bir şekilde ilerletmelerini sağlar. İyi bir eğitim almış bir yazılımcı, yazılım geliştirme dillerine hakim olduğundan, hata oranlarını düşürür, daha az zaman kaybı yaşar ve hızlıca çözümler üretebilir." Ö2 [Sistematik ürün gelişimi] kodu ile "Fabrikalarda kullandığımız CNC vb. makineler, akıllı ev aletleri ve sistemleri 3 boyutlu yazıcılar vb. düşünecek olursak programlamanın ürün geliştirme sürecini oldukça etkilediğini görebiliriz." Ö11 [Sürekli gelişen alan] kodu ile "…Bir programcı hayat boyu kendini geliştirmek ve yeni şeyler öğrenmek zorundadır. Özetle programlama eğitimi hayat boyudur, hiç bitmez." Öğrenciler, programlama eğitiminin ürün geliştirme sürecini hızlandırdığını, verimliliği artırdığını ve analitik düşünme becerilerini geliştirdiğini belirtmektedir. Ayrıca, teknolojik gelişmelerle birlikte programlamanın ürün geliştirme sürecindeki rolünün giderek arttığına dikkat çekmektedirler. Öğrenciler programlama alanındaki sürekli öğrenme ve gelişimin önemini vurgulamaktadır.

Katılımcıların "Programlama öğrenirken veya yaparken zorlandığınız durumlardan bahseder misiniz? Sorusuna yönelik görüşleri kategorilere ve kodlara göre raporlanmıştır (Tablo 20).

Tablo 20. Programlama sürecinde zorlanılan durumlar ile ilgili kategoriler ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
Öğrenme Zorlukları ve Engeller	Analitik ve soyut düşünme	1	Ö1
	Programlama dönüşüm zorluğu	1	Ö5
	Mantık ve sentaks hataları	3	Ö4, Ö7, Ö10
	Belirsiz öğrenim içeriği	1	Ö3
	Dil söz dizim farkı	2	Ö7, Ö10
	İngilizce eksikliği	1	Ö2
	Kütüphane yönetimi süreci	1	Ö12
	Hata bulmada zorlanma	1	Ö10
	Algoritma tasarımında zorlanma	2	Ö6, Ö9
	İlgi ve hevesin kaybolması	1	Ö6
	Ezbere dayalı öğrenme	1	Ö8

Katılımcılar, programlama sürecinde yaşadıkları ve görüşleri, “öğrenme zorlukları ve engeller” kategorisi altında on bir kod altında toplanmıştır (Tablo 20). Katılımcıların kodlar ve görüşleri şöyledir: Ö1 [Analitik ve soyut düşünme] kodu ile “Soyut ve analitik düşünme becerilerinin ön planda olması gereken bir branştır donanımsal ihtiyaçlar giderilirse her öğrenci bir şekilde başarabilir.” Ö5 [Programlama dönüşüm zorluğu] kodu ile “Başlangıçta mantığını anlayamadım. Problemi bilgisayar diline çevirmekte zorlandım.” Ö7 [Mantık ve sentaks hataları] kodu ile “Sentaks hataları ve mantık hataları yapa yapa ilerleniyor.” Ö3 [Belirsiz öğrenim içeriği] kodu ile “Programlamayı kendi kendime öğrendim. Üniversitelerde özel olarak programlama, genel olarak tüm derslerde "ne" yapıldığından kimse bahsetmiyor.” Ö7 [Dil söz dizim farkı] kodu ile “Programlar arası dil ve notasyon farklılıkları zorlanmama sebep oldu.” Ö2 [İngilizce eksikliği] kodu ile “İngilizce eksikliği zorlanmama sebep oldu.” Ö12 [“Kütüphane yönetimi süreci] kodu ile “Özellikle yeni bir kütüphane kullanımı ve açık kaynak yazılımlarda zorlandım.” Ö6 [İlgi ve hevesin kaybolması – Algoritma tasarımında zorlanma] kodları ile “İlgi ve hevesimizin kırılmamasının bu süreçte önemli olduğunu düşünüyorum. Mantıksal ifadeleri düzenli kullanıp, algoritmasını tasarlama süreci de en zor bölümler.” Ö10 [Hataları bulmada zorlanma] kodu ile “Hataları bulmada zorlandım.” Ö8 [Ezbere dayalı öğrenme] kodu ile “Biraz ezbere dayalı olması benim açımdan öğrenimi güçleştiriyor ama onun dışında söyleyebileceğim herhangi bir zorluğu yok.” Şeklinde görüş belirtmişlerdir. Öğrenciler, programlama öğrenirken soyut ve analitik düşünme, döngüler, mantık ve sözdizimi hataları, belirsiz içerik, dil farklılıkları, İngilizce eksikliği, kütüphane yönetimi, algoritma tasarımı ve ezbere dayalı öğrenme gibi zorluklarla karşılaşmaktadır. Ayrıca, “ne yapıyoruz?” ve “neden yapıyoruz?” sorularına yanıt aramanın öğrenciye bırakılması, eğitimin sadece “nasıl” değil, "neden" sorusuna da odaklanması gerektiğini göstermektedir.

Katılımcıların “Programlamayı daha iyi öğrenmenizi sağlayan durumlardan bahseder misiniz?” Sorusuna ilişkin görüşleri Tablo 21’de raporlanmıştır.

Tablo 21. Programlama öğrenme ile ilgili kategoriler ve kodlar

Kategoriler	Kodlar	Frekans	Katılımcı
Öğrenme Yöntemleri ve Motivasyon	Uygulamalı öğrenme	6	Ö1, Ö4, Ö7, Ö8, Ö9, Ö10
	Algoritma önemi	2	Ö4, Ö5
	Ekip çalışması	3	Ö4, Ö6, Ö10
	Kaynak kullanımı	3	Ö4, Ö5, Ö10
	Bağlamsal öğrenme	1	Ö12
	Motivasyon ve merak	3	Ö3, Ö4, Ö11

Katılımcılar programlamayı öğrenme konusundaki görüşleri "öğrenme yöntemleri ve motivasyon" kategorisinde altı kod altında toplanmıştır (Tablo 21). Katılımcıların bu kodlar ile ilgili görüşleri şu şekildedir: Ö4 [Uygulamalı öğrenme – Algoritma önemi] kodları ile "Gerçek problemlerle karşılaşmak, öğrenilen bilgilerin pekişmesini sağlar. Algoritma ve veri yapıları, kodunuzu daha verimli hale getirmenizi sağlar. Başkalarının yazdığı kodları incelemek…" Ö6 [Ekip çalışması] kodu ile "Tek başına değil de bir veya bir kaç arkadaşımla beraber bir programda kodlama yapmak hem daha akılda kalıcı, hem de kendimi geliştirme noktasında daha verimli olabiliyor" Ö5 [Kaynak kullanımı] kodu ile "Daha fazla alıştırma yapmak ve farklı kaynaklara bakmak işimi kolaylaştırdı…" Ö12 [Bağlamsal öğrenme] kodu ile "Programlama gerçek hayatla ilişkilendirildiğinde daha iyi öğrenilir..." Ö11 [Motivasyon ve merak] kodu ile "Bana göre bu tamamen motivasyonla açıklanacak bir durumdur." Şeklinde görüş belirtmişlerdir. Öğrenciler, öğrenme sürecinde uygulamalı eğitim, gerçek projeler ve algoritma bilgisinin geliştirilmesini önemli bulmakta; ekip çalışması ve online kaynakların da süreci olumlu etkilediğini belirtmektedir. Programlamanın gerçek hayatla bağdaştırılması ve motivasyonun yüksek tutulması, öğrenme deneyimini zenginleştiren faktörlerdir. Bu da programlama eğitiminde teori ve pratiğin dengeli, motivasyonu destekleyen bir ortamın gerekliliğini göstermektedir

SONUÇ TARTIŞMA VE ÖNERİLER

Bu bölüm, mühendislik öğrencilerinin bilgisayar programlamaya yönelik öz-yeterlik düzeyleri ve bu konudaki görüşlerini inceleyen çalışmanın bulgularını özetlemekte, alanyazın ışığında tartışmakta ve gelecekteki çalışmalar ile eğitim uygulamalarına yönelik öneriler sunmaktadır.

Çalışmanın Öz-Yeterlik Düzeyi ve Boyutları ile ilgili nicel bulguları, mühendislik öğrencilerinin bilgisayar programlamaya yönelik genel öz-yeterlik düzeylerinin ortalama $\bar{X} = 63.86$ puan ile orta düzeyde olduğunu göstermiştir. Bu bulgu, öğrencilerin programlamayı en baştan zor olarak algılaması ve bu durumun başarılarını olumsuz etkilemesi üzerine yoğunlaşan alanyazındaki çalışmalarla (Abdüsselam vd., 2021; Askar ve Davenport, 2009) paralellik göstermektedir. Öğrencilerin genel öz-yeterliklerinin yüksek değil de orta düzeyde kalması, programlama eğitiminin başında öğrencilerin yaşadığı güven eksikliğini de desteklemektedir.

Alt boyutlar incelendiğinde ise, öğrencilerin en yüksek yetkinlik algısına iş birliği ve kontrol/mantıksal düşünme boyutlarında sahip olduğu; en düşük ortalamanın ise algoritma boyutunda olduğu saptanmıştır. Özellikle iş birliği boyutundaki yüksek öz-yeterlik algısı, mühendislik eğitimindeki grup projeleri ve ekip çalışmasının önemine dair öğrenci farkındalığının yüksek olduğunu göstermektedir. Nitel bulgularda da öğrencilerin iş birliğini farklı bakış açıları ve hata yakalama açısından zorunluluk olarak görmeleri bu durumu desteklemektedir. Algoritma boyutundaki yetkinlik algısının diğer boyutlara göre daha düşük olması dikkat çekicidir. Bu durum, nitel bulgularda öğrencilerin programlama sürecinde algoritma tasarımında zorlanma ve mantık hataları yaşadıklarını belirtmeleriyle örtüşmektedir. Algoritma, öğrencilerin programlamanın temeli olduğunu kabul etmelerine rağmen, bu alandaki uygulama ve soyut düşünme becerisi konusunda kendilerini yeterli hissetmediklerini düşündürmektedir. Bu, programlama eğitiminde analitik ve soyut düşünme becerilerini geliştirecek, algoritma odaklı pratiklerin artırılması gerektiğine işaret etmektedir.

Programlamaya yönelik öz-yeterlik puanları cinsiyete göre anlamlı bir farklılık göstermemiştir (p>.05). Bu sonuç, mühendislik eğitimi bağlamında programlama öz-yeterliğinde cinsiyetler arası bir fark olmadığını göstererek alanyazına önemli bir katkı sunmaktadır. Bu durum, mühendislik öğrencilerinin bu beceriyi cinsiyetten bağımsız olarak geliştirebildiği veya ders içeriğinin cinsiyet yanlısı bir algı yaratmadığı şeklinde yorumlanabilir. Programlamaya yönelik öz-yeterlik puanları mühendislik bölümleri arasında anlamlı farklılık göstermiştir (p<.05). Bilgisayar Mühendisliği öğrencilerinin öz-yeterlik puanları diğer tüm bölümlere göre anlamlı derecede yüksektir. Ayrıca İnşaat Mühendisliği öğrencilerinin puanları da Makine ve Elektrik-Elektronik Mühendisliği öğrencilerine göre farklılaşmaktadır. Bu beklenen bir sonuçtur; Bilgisayar Mühendisliği, programlamanın temel disiplini olması nedeniyle öğrencilerin bu alanda doğal olarak daha yüksek öz-yeterliğe sahip olması normaldir. Diğer bölümlerdeki farklılıklar ise, programlama derslerinin zorunluluk, içeriği ve verildiği dönem gibi faktörlere bağlı olabilir. Programlama dersi sayısı ile öz-yeterlik arasında anlamlı bir farklılık bulunmuştur (p<.05). 4 ve daha fazla ders alan öğrencilerin öz-yeterlikleri, 1, 2 ve 3 ders alan öğrencilere göre anlamlı ölçüde daha yüksektir. Bu bulgu, tecrübe ve pratik etme sıklığının öz-yeterlik inancını doğrudan artırdığını gösteren Bandura (1997)'nın sosyal öğrenme teorisiyle tamamen uyumludur. Ramalingam vd. (2004), öğrencilerin programlamaya yönelik öz-yeterlik inançlarının öğrenme sürecini ve başarısını önemli ölçüde etkilediğini ortaya koymuşlardır. Programlama öz-yeterliğinin geliştirilmesinde, teorik bilginin ötesinde tekrar ve derinlemesine pratik yapma fırsatlarının kritik öneme sahip olduğu sonucuna varılabilir. Bilgisayara sahip olan öğrencilerin öz-yeterlik puanları, sahip olmayanlara göre anlamlı düzeyde daha yüksektir (p<.05). Bu, özellikle programlama gibi uygulamaya dayalı bir alanda, öğrencinin bireysel pratik yapma ve ders dışında gönüllü olarak kodlama yapma imkanının öz-yeterlik inancını güçlendirdiğini göstermektedir. Kendi bilgisayarına sahip olmak, ders dışında ek alıştırmalar yapma, hataları ayıklama ve farklı programlama dillerini deneyimleme fırsatı tanıdığından, bu durum öğrenme sürecinin ayrılmaz bir parçası olarak değerlendirilmelidir.

Programlama dersine yönelik görüşleri içeren nitel bulgular, öğrencilerin programlamanın mühendislik eğitimi için önemine ve zorluklarına dair güçlü bir farkındalığa sahip olduğunu göstermektedir. Öğrenciler programlamayı problem çözme, hesaplamalar/modelleme ve makinelerle iletişim kurma açısından zorunlu ve temel bir beceri olarak görmektedir. Algoritma Tasarımı, programın temeli olarak kabul edilmekte, ancak öz-yeterlik boyutlarında en zayıf halka olarak öne çıkmaktadır. Hata ayıklama sürecini verimlilik ve zaman kazanma açısından kritik bulmuşlar, ancak bunun yapısal problemleri çözemediği yönünde fikir belirtmişlerdir. Öğrencilerin karşılaştığı zorluklar analitik ve soyut düşünme, mantık hataları ve algoritma tasarımında zorlanma başlıklarında toplanmıştır.

Öneriler, eğitim uygulamalarına ve gelecek araştırmalara yönelik olmak üzere iki ana başlıkta incelenebilir. Eğitim uygulamalarına yönelik olarak;

- Programlama öz-yeterliğinde en zayıf algılanan boyut olan Algoritma tasarımı ve soyut düşünme becerilerini geliştirmeye yönelik özel modüller oluşturularak eğitimin güçlendirilmesi düşünülmelidir.

- Öz-yeterliği artırdığı kanıtlanan ders sayısının fazlalığı ve bilgisayar sahibi olma durumunun olumlu etkisi göz önüne alındığında, öğretim programının, öğrencilere yoğun ve sürekli pratik yapma imkanı sunması sağlanmalıdır.

- Öğrencilerin motivasyonunu artırmak amacıyla, konuların öğrencilerin kendi mühendislik disiplinlerinden somut örnekler ve modellemelerle ilişkilendirilerek anlatılması sağlanmalıdır.

Gelecek araştırmalara yönelik ise

• Bu çalışmada anlamlı fark bulunmayan cinsiyet değişkeni, farklı programlama dilleri ve daha büyük/farklı mühendislik fakülteleri örneklemlerinde tekrar incelenmelidir.

• Programlama dili değişkeni detaylı olarak incelenmelidir.

• Algoritma öz-yeterliği düşük çıkan öğrencilerin bu zorlanmalarının bilişsel yük ve soyutlama becerileri ile ilişkisi deneysel çalışmalarla araştırılabilir.

• Yapay zekâ destekli öğrenme, blok tabanlı programlama, oyun temelli öğrenme ve sanal laboratuvarların etkisi programlama öz-yeterlikleri araştırmaları için odak noktası olabilir.

Araştırmacıların Katkı Oranı
Yazarlar çalışmaya eşit oranda katkı sunmuşlardır.

Etik Bilgisi
Araştırma yayın ve etiğini uyulmuştur. Araştırma sürecine başlamadan önce çalışmanın yürütülmesi için ilgili üniversitenin Fen ve Mühendislik Bilimleri Etik Komisyonundan 30.11.2022 tarih ve E-19928322-302.08.01-203249 sayı ile onay alınmıştır. Ayrıca, çalışmaya katılan tüm öğrencilere araştırmanın amacı hakkında bilgi verilmiş ve gönüllü katılım ilkesine uyularak bilgilendirilmiş onam formu alınmıştır.

Destek ve Teşekkür
Yazarlar çalışma için herhangi bir finansal destek almamışlardır.

Çıkar Çatışması
Yazarlar çalışmada herhangi bir çıkar çatışmasının bulunmadığını beyan etmiştir.

KAYNAKÇA

Abdüsselam, M. S., Turan Güntepe, E., & Durukan, Ü. G. (2021). Problem çözme süreci ve öz-yeterlik algısı üzerinden programlama öğretiminin incelenmesi. *Bayburt Eğitim Fakültesi Dergisi, 16*(31), 149-173. https://doi.org/10.35675/befdergi.758137

Aksoğan, M., Kalemkuş, F., & Özdemir, O. (2020). Üniversitede programlama dersi alan öğrencilerin eğitsel yazılım geliştirmeye yönelik öz-yeterlik algıları. *Adnan Menderes Üniversitesi Eğitim Fakültesi Eğitim Bilimleri Dergisi, 11*(2), 46-58.

Alsancak Sırakaya, D. (2019). Programlama öğretiminin bilgi işlemsel düşünme becerisine etkisi. *Türkiye Sosyal Araştırmalar Dergisi, 23*(2), 575-590. https://dergipark.org.tr/tr/pub/tsadergisi/issue/47639/448409

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin öz yeterlilik algısı ölçeğinin Türkçe formunun güvenirlik ve geçerlik çalışması. *Journal of Measurement and Evaluation in Education and Psychology, 3*(2), 297-308. https://dergipark.org.tr/en/pub/epod/issue/5802/77220

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming among engineering students. *Online Submission, 8*(1). https://files.eric.ed.gov/fulltext/ED503900.pdf

Aytekin, A., Çakır, F. S., Yücel, Y. B., & Kulaözü, İ. (2018). Geleceğe yön veren kodlama bilimi ve kodlama öğrenmede kullanılabilecek bazı yöntemler. *Avrasya Sosyal ve Ekonomi Araştırmaları Dergisi*, *5*(5), 24-41.

Bandura, A. (1997). *Self-efficacy: The exercise of control* (Vol. 11). Freeman.

Bandura, A. (2001). Social cognitive theory: An agentic perspective. *Annual review of psychology*, *52*(1), 1-26. https://doi.org/10.1146/annurev.psych.52.1.1

Belmar, H. (2022). Review on the teaching of programming and computational thinking in the world. *Frontiers in Computer Science*, *4*, 997222. https://doi.org/10.3389/fcomp.2022.997222

Benjamin, M. E., Brown, L. E., Sticklen, J., Ureel, L. C., & Jarvie-Eggart, M. (2023, October). Engaging Novice Programmers: A Literature Review of the Effect of Code Critiquers on Programming Self-efficacy. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). IEEE. https://ieeexplore.ieee.org/abstract/document/10342975/

Benli, K. S., & Tek, F. B. (2021). Programlamaya giriş dersini alan öğrencilerin programlama öz yeterlilik algılarının ve programlamaya bakış açılarının incelenmesi. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 9*(3), 328-347. https://doi.org/10.29130/dubited.770726

Boom, K.-D., Bower, M., Siemon, J., & Arguel, A. (2022). Relationships between computational thinking and the quality of computer programs. *Education and Information Technologies, 27*(6), 8289-8310. https://doi.org/10.1007/s10639-022-10921-z

Büyüköztürk, Ş. (2002). *Sosyal bilimler için veri analizi el kitabı: İstatistik, araştırma deseni, SPSS uygulamaları ve yorum*. Pegem Akademi Yayıncılık.

Can, A. (2018). *SPSS ile bilimsel araştırma sürecinde nicel veri analizi*. Pegem Akademi Yayıncılık.

Cetin, I., & Ozden, M. Y. (2015). Development of computer programming attitude scale for university students. *Computer Applications in Engineering Education*, *23*(5), 667-672. https://doi.org/10.1002/cae.21639

Chakraborty, P. (2024). Computer, computer science, and computational thinking: Relationship between the three concepts. *Human Behavior and Emerging Technologies*, *2024*(1), 5044787. https://doi.org/10.1155/2024/5044787

Chen, I.-S. (2017). Computer self-efficacy, learning performance, and the mediating role of learning engagement. *Computers in Human Behavior*, 72, 362-370. https://doi.org/10.1016/j.chb.2017.02.059

Creswell, J. W., & Plano Clark, V. L. (2023). Revisiting mixed methods research designs twenty years later. *Handbook of mixed methods research designs*, *1*(1), 21-36.

Ekici, M., & Çınar, M. (2020). Bilgisayar programlama öz-yeterlik ölçeğinin türkçe formunun geçerlik ve güvenirlik çalışması. *Anadolu Journal of Educational Sciences International*, *10*(2), 1017-1040. https://doi.org/10.18039/ajesi.725161

Gezgin, D. M., & Adnan, M. (2016). Makine mühendisliği ve ekonometri öğrencilerinin programlamaya ilişkin öz yeterlik algılarının incelenmesi. *Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi*, *17*(2), 509-525. https://dergipark.org.tr/en/pub/kefad/issue/59426/853600

Güçlü, İ. (2020). *Sosyal Bilimlerde Nicel Veri Analizi*. Gazi Kitabevi.

Gürer, M. D., & Tokumacı, S. (2020). Mühendislik fakültesi öğrencilerinin programlamaya yönelik tutumları. *Cumhuriyet Uluslararası Eğitim Dergisi, 9*(4), 1064-1082. https://doi.org/10.30703/cije.671244

Hallström, J., & de Vries, M. J. (2023). *Programming and computational thinking in technology education*. Brill. https://brill.com/downloadpdf/display/title/69429.pdf

Hanjani, G. E. (2019). *Computer programming self-efficacy levels of engineering bachelor students at Eastern Mediterranean University (EMU* (Unpublished master's thesis), Eastern Mediterranean University.

Herlambang, A. D., & Rachmadi, A. (2024). Computational thinking skills theorization in the vocational high school computer programming subject context. *Elinvo (Electronics, Informatics, and Vocational Education)*, *9*(1), 64-75. http://dx.doi.org/10.21831/elinvo.v9i1.64501

Huffman, A. H., Whetten, J., & Huffman, W. H. (2013). Using technology in higher education: The influence of gender roles on technology self-efficacy. Computers in Human Behavior, 29(4), 1779-1786.

ISTE standards at ISTE (2015). ISTE website. https://www.isteconference.org/2015/program/iste_standards.php.

Kayri, M. (2009). Araştırmalarda gruplar arası farkın belirlenmesine yönelik çoklu karşılaştırma (post-hoc) teknikleri. *Journal of Social Science*, 55, 22.

Keskinkılıç, F., & Kalelioğlu, F. (2024). Ön lisans bilgisayar bölümü öğrencilerinin programlama kaygılarının çeşitli değişkenler açısından incelenmesi. *Journal of Uludag University Faculty of Education, 37*(3), 1092-1110. https://doi.org/10.19171/uefad.1493874

Kher, H. V., Downey, J. P., & Monk, E. (2013). A longitudinal examination of computer self-efficacy change trajectories during training. *Computers in Human Behavior*, *29*(4), 1816-1824. https://doi.org/10.1016/j.chb.2013.02.022

Leech, N. L., & Onwuegbuzie, A. J. (2009). A typology of mixed methods research designs. *Quality & Quantity, 43*(2), 265-275. https://doi.org/10.1007/s11135-007-9105-3

Lewis, J. E., Robinson, B. S., & Hawkins, N. (2020, June). First-year engineering student perceptions in programming self-efficacy and the effectiveness of associated pedagogy delivered via an introductory, Two-Course Sequence in Engineering. In *2020 ASEE Virtual Annual Conference Content Access*.

Liu, Y. C., & Chen, H. Y. (2023, August). Exploring Self-Efficacy of Students' Computer Programming Learning. In *2023 IEEE 6th International Conference on Knowledge Innovation and Invention (ICKII)* (pp. 410-412). IEEE. https://ieeexplore.ieee.org/abstract/document/10332701/

Marakas, G. M., Yi, M. Y., & Johnson, R. D. (1998). The multilevel and multifaceted character of computer self-efficacy: Toward clarification of the construct and an integrative framework for research. *Information systems research*, *9*(2), 126-163. https://doi.org/10.1287/isre.9.2.126

Mazman, S. G., & Altun, A. (2013). Programlama–I dersinin BÖTE bölümü öğrencilerinin programlamaya ilişkin öz yeterlilik algıları üzerine etkisi. *Öğretim Teknolojileri ve Öğretmen Eğitimi Dergisi*, *2*(3). https://dergipark.org.tr/en/pub/jitte/issue/25082/264710

Miles, M. B., & Huberman, A. M. (1984). Drawing valid meaning from qualitative data: Toward a shared craft. *Educational researcher*, *13*(5), 20-30.

Özmen Yağız, B., & Usluel, Y. K. (2024). Bilgisayımsal düşünme becerilerinin oyun programlama aracılığıyla geliştirilmesi: ortaokul öğrencileri için bir çerçeve. *Ahi Evran Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 10*(2), 467-486. https://doi.org/10.31592/aeusbed.1444312

Özonur, M. (2022). Computer programming students' learning motivation in programming courses. *Journal of Advanced Education Studies, 4*(1), 61-69. https://doi.org/10.48166/ejaes.1123170

Özyurt, Ö., & Özyurt, H. (2015). Bilgisayar programcılığı öğrencilerinin programlamaya karşı tutum ve programlama öz-yeterliklerinin belirlenmesine yönelik bir çalışma *Eğitimde Kuram ve Uygulama, 11*(1), 51-67. https://dergipark.org.tr/en/pub/eku/issue/5464/74179

Patton, M. Q. (2014). Nitel araştırma ve değerlendirme yöntemleri. Pegem Akademi Yayıncılık.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 171-175). https://doi.org/10.1145/1007996.1008042

Sayın, Z., & Seferoğlu, S. S. (2016). Yeni bir 21. yüzyıl becerisi olarak kodlama eğitimi ve kodlamanın eğitim politikalarına etkisi. *Akademik Bilişim Konferansı*, *3*(5), 1-13.

Shingala, M. C., & Rajyaguru, A. (2015). Comparison of post hoc tests for unequal variance. *International Journal of New Technologies in Science and Engineering, 2*(5), 22-33.

Taşpınar, M. (2017). Sosyal bilimlerde spss uygulamalı nicel veri analizi. Pegem Akademi Yayıncılık.

Tsai, M.-J., Wang, C.-Y., & Hsu, P.-F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Educational Computing Research, 56*(8), 1345-1360. https://doi.org/10.1177/0735633117746747

Turan, S. B., & Erdoğan, A. (2025). Scratch's impact on technological pedagogical content knowledge, computational thinking skills, and computing attitudes of prospective elementary mathematics teachers. *Abant İzzet Baysal Üniversitesi Eğitim Fakültesi Dergisi, 25*(1), 425-445. https://doi.org/10.17240/aibuefd.2025..-1443206

Uysal, H., & Ocak, M. A. (2023). Mühendislik öğrencilerinin ters-yüz sınıf modeli ile programlama öğrenimindeki öz-yeterlilik ve bağlılık algıları. *Bayterek Uluslararası Akademik Araştırmalar Dergisi*, *6*(1), 36-70. https://doi.org/10.48174/buaad.1267998

Üzüm, B., Elçiçek, M., & Pesen, A. (2024). Programlama öğretiminin bilgi işlemsel düşünme becerisi üzerindeki etkisi: bir meta-analiz çalışması. *İstanbul Ticaret Üniversitesi Sosyal Bilimler Dergisi, 23*(49), 1690-1713. https://doi.org/10.46928/iticusbe.1469733

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Yıldırım, A., & Simsek, H. (1999). *Sosyal bilimlerde nitel araştırma yöntemleri*. Seçkin Yayıncılık.

Yıldız Durak, H. (2018). Digital story design activities used for teaching programming effect on learning of programming concepts, programming self-efficacy, and participation and analysis of student experiences. *Journal of Computer Assisted Learning*, *34*(6), 740-752. https://doi.org/10.1111/jcal.12281