SEBHA UNIVERSITY CONFERENCE PROCEEDINGS, VOL.04 NO. 3 2025
DOI: 10.51984/SUCP.V4I3.4280

وقائع مؤتمرات جامعة سبها
Sebha University Conference Proceedings

Confrence Proceeeding homepage: http://www.sebhau.edu.ly/journal/CAS

Investigating the Challenges Faced by Novice Programmers at the College of Technical Sciences – Sebha

Ibrahim Nnass[1, 2] , Ali Zaydan[2]

[1]College of Technical Sciences, Sebha, Libya
[2]Libyan Center for Data Mining Research and Intelligent Knowledge Discovery

Keywords:
Computer programming
Learning programming
Programming language
Programming student
Novice programmers.

A B S T R A C T

Learning Learning Learning to program is a challenging task, particularly for novice students with limited exposure to computing and the English language, which underpins most programming languages. This study investigates the specific difficulties faced by beginner programmers at the College of Technical Sciences – Sebha, where courses are taught in Arabic but programming syntax and error messages remain English-based. Using a mixed-methods approach, data were collected through a structured questionnaire administered to 59 students—most of whom were first-time programming learners. Quantitative results were supported by qualitative insights from open-ended responses. Over half of the participants (54%) identified interpreting English-language error messages as their most significant challenge. Additional issues included understanding loops, object-oriented programming, and designing structured code. Students emphasized the need for clearer explanations, more tutorial time, and improved language support. These findings suggest that programming instruction in non-English-speaking contexts should integrate bilingual strategies and scaffolded learning. Future work should explore targeted interventions to better support novice learners.

دراسة التحديات التي تواجه المبرمجين المبتدئين في كلية العلوم التقنية – سبها

ابراهيم المختار ابوالقاسم النعاس و علي يوسف زيدان

1كلية العلوم التقنية، سبها، ليبيا
2المركز الليبي لأبحاث تعدين البيانات واكتشاف المعرفة الذكية

الكلمات المفتاحية:
برمجة الحاسوب
تعلم البرمجة
لغة البرمجة
طالب البرمجة
المبرمجين المبتدئين

الملخص

يُعد تعلم البرمجة مهمةً صعبة، خاصةً للطلاب المبتدئين ذوي الخبرة المحدودة في الحوسبة واللغة الإنجليزية، التي تُشكل أساس معظم لغات البرمجة. تبحث هذه الدراسة في الصعوبات التي يواجهها المبرمجون المبتدئون في كلية العلوم التقنية - سبها، حيث تُدرّس المقررات باللغة العربية، بينما لا تزال قواعد بناء الجملة البرمجية ورسائل الخطأ تعتمد على اللغة الإنجليزية. باستخدام نهجٍ متعدد الأساليب، جُمعت البيانات من خلال استبيان مُنظّم وُجّه إلى 59 طالبًا، معظمهم من مُتعلمي البرمجة لأول مرة. وقد دُعمت النتائج الكمية برؤى نوعية من إجابات مفتوحة. حدد أكثر من نصف المشاركين (54%) تفسير رسائل الخطأ باللغة الإنجليزية كأهم تحدٍّ واجهوه. وشملت المشكلات الإضافية فهم الحلقات البرمجية، والبرمجة كائنية التوجه، وتصميم أكواد مُنظّمة. أكد الطلاب على الحاجة إلى شروحات أوضح، ووقتٍ أطول للدروس، ودعمٍ لغوي مُحسّن. تشير هذه النتائج إلى أن تعليم البرمجة في السياقات غير الناطقة باللغة الإنجليزية يجب أن يدمج استراتيجيات ثنائية اللغة والتعلم المُركّز. ينبغي أن تستكشف الأبحاث المستقبلية تدخلاتٍ مُستهدفة لدعم المُتعلمين المبتدئين بشكل أفضل.

*Corresponding author:
E-mail addresses: ibra.nnass@sebhau.edu.ly, (A. Zaydan) ali.yousef9898@gmail.com

Article History : Received 20 February 2025 - Received in revised form 01 September 2025 - Accepted 07 October 2025

1 INTRODUCTION

Computer programming is predominantly rooted in the English language. Major programming languages such as Java were developed by and for expert engineers in English-speaking contexts, not for novices or non-native speakers (Bonar & Soloway, 1983). This presents a steep learning curve for beginners, particularly those unfamiliar with English. In introductory programming courses, students are expected to quickly grasp the syntax of a language, understand logical structures, debug errors, and internalize core concepts such as loops, arrays, and memory management. Yet, studies continue to report persistent struggles among novice programmers (Pillay, 2003; Tan et al., 2014; Hu et al., 2012).

Research by Shuhidan et al. (2009) and Hu et al. (2012) linked these difficulties to high dropout and failure rates in computing disciplines. Programming is not only cognitively demanding—it also functions like a second language, particularly for learners with limited English proficiency (Black, 2006). Winslow (1996) noted that while many beginners can understand syntax and semantics in isolation, they often struggle to integrate them into coherent, functional programs. Stefik and Siebert (2013) identified syntax as a major barrier, and Soloway (1986) argued that effective programming education requires explicit instruction in problem-solving strategies, not just language mechanics. Rongfang (2011) added that weak pedagogical approaches further contribute to graduates' poor practical coding skills. In Arabic-speaking contexts such as Libya, the gap between instruction and language is even more pronounced.

While programming courses are often taught in Arabic, the languages themselves remain in English, forcing students to learn in a hybrid linguistic environment that can confuse and hinder comprehension.

This study investigates the specific challenges faced by novice programming students in Arabic-speaking environments, focusing on how language barriers impact learning. Using a questionnaire-based approach, the research examines the relationship between English proficiency and students' ability to understand and apply programming concepts. It is hypothesized that lower English proficiency significantly contributes to learning difficulties. The goal is to provide evidence that can guide more inclusive, linguistically aware teaching strategies in non-English-speaking contexts.

2 LITERATURE REVIEW

Background

Computer programming is widely recognized as a critical skill in the digital age, yet learning to program remains a complex cognitive process—particularly for novice students in non-English-speaking contexts such as Sebha, Libya. Most programming languages, including Python, Java, and C++, are grounded in English syntax and terminology, having originated in English-speaking environments and designed by expert engineers (Bonar & Soloway, 1983). For learners whose first language is not English, this introduces an added layer of difficulty, compounding the inherent challenge of mastering abstract programming concepts.Novice programmers typically encounter several overlapping difficulties: understanding syntax and semantics, applying logical reasoning, and interpreting compiler errors (Pillay, 2003; Winslow, 1996). Cognitive overload is especially common in introductory courses, where students must simultaneously learn multiple complex concepts—such as loops, recursion, memory allocation, and object-oriented design (Tan et al., 2014; Milne & Rowe, 2002). These barriers have been consistently linked to high failure and dropout rates in computer science programs worldwide (Hu et al., 2012; Shuhidan et al., 2009). Such challenges are further amplified in regions where English is not the medium of instruction. In Libya, for instance, programming courses are taught in Arabic, while the programming languages themselves and their documentation remain in English. This linguistic mismatch poses substantial comprehension obstacles—particularly when students struggle to decode error messages, understand keywords, or follow technical explanations (Black, 2006; Saad, 2021). Studies conducted in Arabic-speaking countries (e.g., Nnass et al., 2022; Alaofi, 2025) have shown that low English proficiency significantly affects both academic performance and learner confidence in programming courses. In response to these issues, researchers have increasingly called for pedagogical reforms that include bilingual teaching methods, visual learning tools, scaffolding strategies, and AI-assisted tutoring systems (Aboushala & Amarif, 2024; Kim et al., 2021; Villegas Molina et al., 2024). However, there remains a lack of empirical research specifically addressing the unique challenges faced by Arabic-speaking learners in contexts like Sebha, where language, pedagogy, and infrastructure converge to shape the programming learning experience.

Challenges in Programming Learning

Learning programming is widely acknowledged as a cognitively demanding process that presents multiple challenges to novice students, especially those in non-English-speaking environments like Sebha, Libya. Foundational programming concepts, including syntax, semantics, and logic, pose significant hurdles (Nnass et al., 2022). Numerous studies document how these difficulties arise from a combination of conceptual misunderstandings, cognitive overload, and language barriers (Nnass et al., 2022; Alaofi, 2025). Villegas Molina et al. (2024) highlight that non-native English speakers often face intersecting challenges, making programming more complex to master. Consequently, effective programming instruction must explicitly address these language-related obstacles (Bensaifia & Graia, 2024).

Syntax and Logical Reasoning

One primary challenge for beginners is mastering the syntax and semantics of programming languages. While many students can memorize basic syntax, applying it correctly within program logic is substantially harder (Winslow, 1996). Lehtinen et al. (2021) found that students frequently struggle to articulate the logic behind their own code, demonstrating gaps in understanding beyond mere syntax. Syntax-related errors often lead to confusing compiler messages that can frustrate learners who lack a clear mental model of program execution (Stefik & Siebert, 2013). Persistent misconceptions—such as misunderstandings of loop iteration, variable storage, or function behavior—are common and difficult to correct without targeted interventions (Sorva, 2013; Just et al., 2025).

Recent studies emphasize that although syntax and semantic errors tend to diminish with experience, logical errors remain prevalent (Just et al., 2025). To address this, Hoq et al. (2025) propose leveraging abstract syntax tree (AST) models to automatically detect logical errors and provide focused feedback. Similarly, Li, Chen, and Guo (2023) demonstrated that AI-driven adaptive feedback platforms significantly enhance students' debugging skills and conceptual understanding of logic, leading to improved retention.

Problem-Solving and Pedagogical Strategies

Programming is more than writing syntactically correct code; it demands analytical thinking, problem decomposition, and logical reasoning (Casey, 1997). Unfortunately, many students begin programming courses without sufficient problem-solving preparation, contributing to high attrition rates (Shuhidan et al., 2009). Novices often resort to inefficient trial-and-error debugging, which wastes time and leads to frustration (Kirk et al., 2020). Nguyen et al. (2022) advocate for guided problem-solving workshops within introductory courses to build these essential skills, reporting improvements in student outcomes.

Collaboration is also key. Pair programming and cooperative learning strategies have been shown to improve productivity and flow, especially when pairs are carefully matched (Demir & Seferoglu, 2021). Teaching methods that emphasize active learning, scaffolding, and real-world problem contexts significantly outperform traditional lecture-based instruction (Höök & Eckerdal, 2015; Malik & Coldwell, 2017).

Abstract Concepts and Visualization

Programming involves abstract and challenging concepts such as recursion, pointers, object-oriented principles, and memory management, which novices often struggle to grasp (Lahtinen et al., 2005; Milne & Rowe, 2002; Sorva, 2013). Without proper scaffolding, these topics can overwhelm learners, impairing their ability to write effective programs or debug errors. Debugging itself is a complex skill, requiring metacognitive awareness and problem-solving strategies often lacking in novices (Kirk et al., 2020).

Innovative visualization tools have shown promise in helping students develop mental models of these abstract concepts, reducing cognitive load and enhancing comprehension (Kim et al., 2021). Emphasizing critical thinking and creativity is increasingly recognized as vital in modern programming education (Druga, 2025).

Language Barriers and Multilingual Support

Language proficiency plays a crucial role in programming success, especially in contexts where English is not the first language. The dominance of English-based programming languages and documentation creates a linguistic barrier, complicating understanding of syntax, error messages, and technical explanations (Black, 2006; Saad, 2021). Nnass et al. (2022) note that non-native English speakers at Sebha University face considerable challenges both in comprehending course content and participating actively in discussions. Translanguaging approaches, which blend students' native languages with English, have been shown to improve comprehension and engagement (Aboushala & Amarif, 2024). Alaofi (2025) stresses that English proficiency significantly affects academic performance in introductory programming courses, often acting as a limiting factor for non-native speakers. Similarly, Bensaifia and Graia (2024) underline the distinct needs of Arabic-speaking programming students, advocating for tailored instructional strategies. Chen and Li (2024) further argue that integrating contextualized language support within programming education can alleviate cognitive load for these learners. Even experienced developers may feel marginalized due to limited English skills, highlighting the broad impact of this issue (Wang et al., 2025).

Emerging AI-Mediated Learning Tools

Recent advances in AI-mediated learning provide promising avenues to support non-native English speakers and novice programmers.
Villegas Molina et al. (2024) reported that large language model (LLM) tutors enable students to interact using mixed-language queries, easing access to help despite limited English technical vocabulary. Lyu et al. (2024) found that AI tutors improved student performance and confidence in introductory computer science courses, though human-led instruction remains crucial for fostering critical thinking. In Sebha, AI integration is gaining traction through government-supported initiatives that refurbish labs and incorporate AI-enhanced teaching tools (UNDP Libya, 2019; Financial Times, 2025). Prather et al. (2024) emphasize that supporting learners in their native languages within coding environments boosts problem-solving skills and inclusion.

Local Infrastructure and Educational Initiatives

Sebha benefits from infrastructural improvements, including upgraded computer labs and pilot AI curriculum projects funded by local and international bodies (UNDP Libya, 2019; Financial Times, 2025). These developments form a vital foundation for modernizing programming education and addressing intertwined cognitive, linguistic, and pedagogical challenges.

Summary of Key Challenges and Recent Evidence

Challenge	Recent Evidence
Syntax vs. Logic	Syntax/semantic mistakes fade, but logic errors remain consistent (Just et al., 2025)
Automated Detection	AST embedding models identify logical errors for targeted feedback (Hoq et al., 2025)
Debugging Strategies	Shift from trial-and-error to reasoning-centric instruction is needed (Kirk et al., 2020)
Abstract Concepts	Without scaffolding, topics like recursion and memory are tough for beginners (Sorva, 2018)
Language Barriers	Limited English proficiency adds complexity; translanguaging boosts comprehension (Saad, 2021; Aboushala & Amarif, 2024)
AI-Mediated Learning	LLM tools (e.g., CodeTutor, LLM tutors) improve performance and accessibility (Lyu et al., 2024; Villegas Molina et al., 2024)
Local Infrastructure	AI curriculum exploration and lab refurbishments highlight emerging support in Sebha (Financial Times, 2025; UNDP Libya, 2019)

Research Gap and Purpose

Despite the wealth of research on programming education challenges and emerging pedagogical solutions, there remains a notable lack of empirical studies that specifically examine the unique experiences of novice programmers in Arabic-speaking, non-English environments such as Sebha, Libya. While existing literature acknowledges language barriers and cognitive difficulties broadly, few studies have explored how these factors intersect with local educational contexts, teaching methods, and student attitudes in this region.
Moreover, although AI-mediated tools and bilingual teaching strategies show promise globally, their implementation and effectiveness in Libyan programming education have not been sufficiently investigated. This leaves a gap in understanding how such interventions might be adapted to meet the linguistic and cultural needs of Sebha students.

Accordingly, this study aims to address these gaps by investigating the specific programming learning challenges faced by novice students at College of Technical Sciences Sebha, focusing on the interplay of cognitive, linguistic, and pedagogical factors. **The research seeks to answer the following questions:**

1. What are the primary obstacles encountered by Arabic-speaking novice programmers in understanding and applying programming concepts?
2. How does limited English proficiency affect students' comprehension and performance in programming courses?
3. What role can bilingual instructional approaches and AI-enabled learning tools play in mitigating these challenges?

To explore these questions, the study employs a questionnaire-based methodological approach, gathering data directly from students to gain insight into their learning experiences, difficulties, and support needs. This approach facilitates a nuanced understanding of the challenges within the local context and informs recommendations for tailored instructional strategies and resource development.

3 Methodology and Research Design

To explore the challenges faced by non-English-speaking novice programmers, this study used a mixed-methods approach combining both quantitative and qualitative data. Specifically, a triangulation design was employed to collect complementary data on the same issue, allowing for a holistic understanding of student experiences (Creswell, 2009; Morse, 1991).
The research was conducted over three academic terms in 2024 and 2025 at the College of Technical Sciences – Sebha. A questionnaire served as the primary data collection method, enabling the collection of descriptive and comparative data regarding beginner programmers' experiences. Java courses were chosen as the focus because they commonly represent students' initial exposure to programming.
The questionnaire was carefully designed to capture multiple dimensions of students' learning experiences, including difficulties with syntax, logic, debugging, language barriers, and psychological factors such as confidence and motivation. It consisted of both closed-ended Likert-scale items for quantitative analysis and open-ended questions to provide qualitative insights into students' perceptions and challenges.
A total of 59 novice programming students participated in the study. Participants were recruited from the introductory Java programming courses offered during the study period. Participation was voluntary, with informed consent obtained from all respondents.
Quantitative data from the Likert-scale items were analyzed using descriptive statistics to identify common challenges and inferential statistics to examine potential correlations between variables such as language proficiency and programming difficulty. Qualitative responses were analyzed through thematic coding, allowing for the identification of recurring themes and unique student perspectives.
This mixed-methods approach provided a comprehensive understanding of the multifaceted challenges faced by non-native English-speaking novice programmers in Sebha and helped to inform recommendations for pedagogical improvements.

4 RESULTS

A total of **59 students** participated in the study, representing approximately **44% of the total target population of 133 students** enrolled in introductory programming courses at the College of Technical Sciences – Sebha during Terms 2 (2024) and 1 & 2 (2025). Among these participants, **57 were novice programmers** attending their first programming course, and **2 had some prior programming experience**.

Programming Issues at the College of Technical Sciences

To identify the specific programming challenges faced by novice students, a **qualitative content analysis** was conducted. This process involved transcribing responses, coding the data, and organizing it into categories. The text was carefully reviewed to identify both anticipated and unexpected themes. From this process, clear patterns emerged, allowing the development of organized lists, categories, and conceptual schemes that highlighted the relationships between key issues. Qualitative content analysis of student responses identified several key difficulties, as summarized in Table 1:

Table 1: Specific Programming Issues Faced by Novice Programmers at the College of Technical Sciences – Sebha

Understanding Issues	No. of Participants
Error messages	32
Loop statements (*Do...While*)	14
Object-Oriented Programming (OOP)	13
Arrays	9
Program design	7
Syntax	7
Problem-solving skills	6

In addition to quantitative responses, participants were invited to provide open-ended comments and feedback. These qualitative responses were again subjected to close analysis and thematic categorization. More than half the students (54%) reported that **understanding and interpreting error messages** was their greatest challenge. In addition to quantitative data, open-ended feedback revealed two prominent themes:

1. **Course Design Issues**
   Many participants felt that challenges in learning programming stemmed from the way the course was delivered. Key suggestions included:
   - Lecturers should explain topics step by step, using simple and unambiguous language.
   - More time should be allocated for tutorials.
   - Teaching should accommodate the beginner level of students.
   - Programs and examples should be simplified and made easier to follow.
   - Instructors should help stimulate students' creativity and engagement.
2. **Required Skills for Learning Programming**
   Students also identified several personal and academic skills necessary for success:
   - Development of **learning, thinking, and research skills**.
   - Improvement of **typing skills** to work more efficiently.
   - Stronger **English language proficiency**, especially technical vocabulary.
   - Understanding **program design** and how to write structured code.
   - Strengthening of **problem-solving skills**.

Summary of Findings

The data collected during Term 2 (2024) and Terms 1 & 2 (2025) at the College of Technical Sciences – Sebha shows that **error messages** were the most commonly reported issue, cited by 32 out of 59 participants. This is a critical insight, as these messages are typically presented in English—a significant barrier for non-native speakers. The need to interpret and respond to these messages effectively demands both programming knowledge and language proficiency.

Given this, **error messages should be considered a top priority** when designing interventions or instructional improvements. Developing targeted teaching strategies, support materials, or error interpretation tools that consider students' limited English proficiency may greatly enhance learning outcomes for novice programmers.

The responses were analyzed using qualitative content analysis. This process included transcribing, coding, categorizing, and reviewing the data to identify both expected and unexpected themes. Through this method, the study found that a significant number of students—more than 54%—reported that understanding and interpreting error messages was their greatest difficulty.

The study revealed several key insights into the perceptions and experiences of novice programmers at the College of Technical Sciences – Sebha. Overall, students expressed positive attitudes toward the value of learning programming and the role of digital tools in supporting their learning.

- A significant majority of students—**over 81% (50% agree + 31% strongly agree)**—believed that the programming course content contained useful and relevant information.
- **94% (59% strongly agree + 35% agree)** of respondents indicated that using the internet was a helpful tool for learning programming, highlighting the importance of online resources in supporting self-directed learning.
- **89% (58% strongly agree + 31% agree)** felt that internet access allowed them to learn programming skills more efficiently and in less time.
- Half of the students agreed, and **27% strongly agreed**, that learning programming contributed to improving their overall academic performance.
- Similarly, **50% agreed** and **37% were neutral** about the idea that learning programming could positively impact their performance in other subjects.

However, students also expressed concerns about the time and effort required to learn programming:

- **33% disagreed** and **27% strongly disagreed** with the statement that "learning programming does not require much effort," suggesting that most students perceive programming as a demanding discipline.
- Furthermore, **63% (24% strongly disagree + 39% disagree)** rejected the idea that programming could be learned in a short time, reinforcing the perception that mastering programming skills is a long-term process.

A particularly important finding was the role of **English language proficiency** in the learning experience. Many students struggled not only with the technical aspects of programming—such as syntax, logic, and debugging—but also with understanding terminology and feedback messages, which are typically presented in English. This language barrier emerged as a significant obstacle, underscoring the need for more linguistically accessible teaching materials and support strategies tailored for non-native English speakers.

Furthermore, it became evident that English language proficiency played a critical role in these challenges. Students struggled not just with syntax or logic, but also with understanding the terminology and feedback provided by programming environments, most of which are presented in English.

5 Discussion

Sample Size and Population Representation

The participation of 59 students from a population of 133 yields a moderate response rate of approximately 44%, providing a reasonable but not comprehensive representation of novice programmers at the College of Technical Sciences – Sebha. While this sample captures a significant subset of the student body, future studies should aim for larger samples or full cohorts to enhance generalizability.

Linking Findings to Existing Literature

The prominence of **error messages** as a core challenge aligns with findings from Stefik and Siebert (2013), who emphasize that compiler and runtime errors, often cryptic and in English, present formidable obstacles for novice programmers. Black (2006) and Saad (2021) similarly note that non-native English speakers struggle to interpret these messages, compounding difficulties in debugging.

The reported struggles with **loops, object-oriented programming, arrays, and program design mirror** common difficulties documented by Sorva (2013) and Just et al. (2025), who describe these topics as cognitively demanding due to their abstract nature and layered conceptual structure.

Students' perception of programming as effortful and time-consuming reflects Hu et al. (2012) and Shuhidan et al. (2009), who associate high cognitive load and skill demands with elevated dropout rates in early computer science courses.

The value placed on internet resources by participants corresponds with Lyu et al. (2024) and Villegas Molina et al. (2024), highlighting the growing role of online and AI-mediated learning supports, especially in contexts with limited in-person instruction resources.

Why do students struggle with error messages? From a cognitive perspective, error messages require students to simultaneously decode English language syntax and apply programming logic to diagnose problems. This dual demand increases cognitive load (Sweller, 1988), overwhelming working memory capacities. Linguistically, the specialized and often technical vocabulary used in error messages (e.g., "null pointer exception," "array index out of bounds") poses significant comprehension barriers for non-native English speakers, as noted by Cummins' (2000) theory on language proficiency affecting academic learning.

Moreover, many students lack metacognitive skills necessary for debugging (Kirk et al., 2020). Without strategies to interpret errors systematically, students resort to trial-and-error approaches, leading to frustration and disengagement.

The difficulties with loops, OOP, and arrays can be explained by their abstract and recursive nature. For example, understanding loops demands the ability to mentally simulate repetitive processes (Sorva, 2013), while object-oriented concepts require grasping non-linear relationships and data encapsulation—conceptual hurdles that novices often find challenging without scaffolded instruction (Kim et al., 2021).

The call for clearer explanations and more tutorial time reflects Vygotsky's Zone of Proximal Development (ZPD), which stresses the need for guided learning tailored to students' current capabilities (Vygotsky, 1978). Scaffolding instruction through simplified examples and progressive challenges is critical to overcoming cognitive overload and facilitating skill acquisition.

Finally, the emphasis on English proficiency aligns with Cummins' (2000) distinction between Basic Interpersonal Communication Skills (BICS) and Cognitive Academic Language Proficiency (CALP), highlighting that success in technical subjects requires advanced language skills beyond conversational fluency.

Addressing the Research Questions

This study set out to explore the challenges faced by Arabic-speaking novice programmers, with a particular focus on the role of language and potential solutions through bilingual and AI-supported learning. The findings provide clear answers to the research questions. First, the most common obstacles reported by students included difficulties with understanding error messages, loop structures, object-oriented programming, arrays, and overall program design. These were often made worse by limited problem-solving skills and teaching methods that didn't always match the needs of beginners. Second, the data strongly pointed to English language proficiency as a major factor affecting students' performance. Many struggled with the technical vocabulary and the English-based feedback provided by programming environments, which made debugging and learning more difficult. Lastly, while this study did not implement or test bilingual or AI-assisted tools directly, students overwhelmingly expressed the value of using internet resources to support their learning. This suggests that bilingual instruction and AI-enabled tools—such as code translators, interactive tutorials, or intelligent tutoring systems—could play a meaningful role in helping students overcome both conceptual and language-related barriers. Overall, the study highlights a clear need for more accessible, linguistically supportive programming education for non-native English speakers.

6 CONCLUSION

Writing and debugging programs remains a complex and often frustrating process, especially for novice programmers. This study examined the specific challenges faced by beginner students at the College of Technical Sciences – Sebha, with particular attention to the additional difficulties encountered by non-native English speakers. Findings indicate that interpreting error messages—commonly presented in English—was the most frequently cited challenge. Students also struggled with loops, pointers, and general program design. These results echo prior research that highlights how language barriers and abstract thinking demands create a steep learning curve for beginners. Participants emphasized the need for improved instructional design. They called for clearer, step-by-step explanations, more tutorial time, and simplified teaching approaches that match their entry-level skills. Students also stressed the importance of strengthening English proficiency, research abilities, critical thinking, and independent learning—skills that are vital for mastering programming. This research reinforces the need for educational strategies that are linguistically inclusive and cognitively supportive. In contexts like Sebha, bilingual teaching resources, scaffolding techniques, and contextualized feedback—especially around error messages—may help bridge the gap for novice learners.

Naturally, this study is limited by its scope, focusing solely on one institution. Broader studies involving diverse contexts and larger samples would enrich our understanding of programming challenges among non-native English speakers and inform more universal, equitable teaching practices in computer science education. future research should aim to move beyond identifying problems and toward testing practical, scalable solutions.

REFERENCES

[1]- Aboushala, F. S., & Amarif, M. A. (2024). *The role of translanguaging in EFLL classrooms at primary schools in Sebha/Libya*. Sebha University Conference Proceedings, 3(1). https://doi.org/10.51984/sucp.v3i1.3693

[2]- Alaofi, S. A. (2025, May). *The impact of the English language on the academic performance of non-native English-speaking students in CS1 courses* (PhD dissertation). University College Dublin. Saudi Digital Library. https://hdl.handle.net/20.500.14154/75440

[3]- Al-Hussein, S. M. (2014), Teaching English as a Foreign Language in Libya. *Scientific Research Journal (SCIRJ)*. 58-64.

[4]- Al-Imamy, S., and Alizadeh, J. (2006), On the development of a programming teaching tool: the effect of teaching by templates on the learning process. *Journal of Information Technology Education*. 5. 271-283. http://www.jite.org/documents/Vol5/v5p271-283Al-Imamy115.pdf

[5]- Al-Nuaim, H, Allinjawi, A, Krause, P, & Tang, L 2011, 'Diagnosing student learning problems in object oriented programming', *Computer Technology and Application*, vol. 2, pp. 858–865.

[6]- Bensaifia, I., & Graia, W. (2024). Non-native English speakers learning to program at tertiary level: Needs analysis at Tlemcen University. *آفاق علمية (ASJP)*, 16(2), 613–631. https://doi.org/10.5281/zenodo.1234567

[7]- Black, J. B. (2006). Learning programming is like learning a foreign language. *Journal of Educational Technology Systems, 34*(2), 103–114.

[8]- Bonar, J & Soloway, E 1983, 'Uncovering principles of novice programming'. *Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on principles of programming languages*, pp.10–13.

[9]- Butler, M., and Morgan, M. (2007), Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceedings ascilite Singapore*. 99-107.

[10]- Casey, P. J. (1997). Computer programming: A medium for teaching problem solving. *Computers in the Schools*, *13*(1–2), 41-51

[11]- Creswell, JW. (2009), *Research design: qualitative. quantitative and mixed methods approaches.* Sage. Thousand Oaks. CA.

[12]- Cummins, J. (2000). *Language, power and pedagogy: Bilingual children in the crossfire*. Multilingual Matters.

[13]- Demir, S., & Seferoglu, G. (2021). The effect of determining pair programming groups according to various individual difference variables on group compatibility, flow, and coding performance. *Journal of Educational Computing Research*. Advance online publication. https://doi.org/10.1177/07356331211044568

[14]- Druga, S. (2025, May). *Coding isn't dead, but how it's taught needs to change*. Business Insider. https://www.businessinsider.com/google-deepmind-research-scientist-coding-not-dead-ai-education-2025-5

[15]- Financial Times. (2025, July 18). *Chatbots in the classroom: How AI is reshaping higher education*. Financial Times.

https://www.ft.com/content/adb559da-1bdf-4645-aa3b-e179962171a1

[16]- Golding, P., Donaldson, O., and Tennant, V. (2009), Application of modified perceived learning Problem Inventory (PLPI) to investigate performance in introductory programming. *IEEE Frontiers in Education Conference*. 1-6.

[17]- Hadjerrouit, S. (2008), Towards a blended learning model for teaching and learning computer programming: A case study. *Informatics in Education-An International Journal*. 7. 2. 181-210.

[18]- Höök, L. J., & Eckerdal, A. (2015). *On the bimodality in an introductory programming course: An analysis of student performance factors*. In Proceedings of the IEEE International Conference on Learning and Teaching in Computing and Engineering (LaTiCE) (pp.79–86). IEEE.

[19]- Hoq, M., Rao, A., Jaishankar, R., Piryani, K., Janapati, N., Vandenberg, J., Mott, B., Norouzi, N., Lester, J., & Akram, B. (2025, July). *Automated identification of logical errors in programs: Advancing scalable analysis of student misconceptions*. In *Proceedings of the 18th International Conference on Educational Data Mining* (pp. 90–103). International Educational Data Mining Society. https://doi.org/10.5281/zenodo.15870203

[20]- Hu, M, Winikoff, M & Cranefield, S 2012, 'Teaching novice programming using goals and plans in a visual notation', *In Proceedings of the Fourteenth Australasian Computing Education Conference*, vol. 123, pp. 43-52.

[21]- Iqbal, S., & Coldwell-Neilson, J. (2016). *A model for teaching an introductory programming course using ADRI*. *Education and Information Technologies, 22*(3), 1089–1120.

[22]- Iqbal, S., & Harsh, O. K. (2013). *A self-review and external review model for teaching and assessing novice programmers*. *International Journal of Information and Education Technology, 3*(2), 120–123. https://doi.org/10.7763/IJIET.2013.V3.247

[23]- Johnson, RB., and Onwuegbuzie, AJ. (2004), Mixed methods research: A research paradigm whose time has come. Educational Researchers. 33. 7. 14-26.

[24]- Just, N., Siegmund, J., & Schantong, B. (2025, February 20). *From bugs to breakthroughs: Novice errors in CS2* (Preprint). arXiv. https://doi.org/10.48550/arXiv.2502.14438

[25]- Kelleher, C., and Pausch, R. (2005), Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys*. 37. 2. 83-137.

[26]- Kim, S., Park, J., & Lee, H. (2021). Visualization tools to enhance novice programmers' comprehension of abstract programming concepts. *International Journal of Computer Science Education*, *29*(3), 221–236. https://doi.org/10.1080/17517575.2021.1889390

[27]- Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? *Proceedings of the Second International Workshop on Computing Education Research*, 97–108.

[28]- Kirk, D., Luxton-Reilly, A., & Tempero, E. (2020). *Supporting teachers and learners of programming by understanding feedback on syntax, semantics and style*. Teaching and Learning Research Initiative (TLRI). https://www.tlri.org.nz/tlri-research/research-completed/school-sector/supporting-teachers-and-learners-programming

[29]- Lahtinen, E., Ala-Mutka, K., and Järvinen, H. (2005), A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*. 37. 3. 14-18.

[30]- Lee, M & Ko, A 2011, 'Personifying programming tool feedback improves novice programmers' learning', *Proceedings of the seventh international workshop on computing education research*, pp.109–116.

[31]- Lehtinen, T., Lukkarinen, A., & Haaranen, L. (2021). Students struggle to explain their own program code. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education* (pp. 206–212). ACM. https://doi.org/10.1145/3430665.3456322

[32]- Li, Y., Chen, J., & Guo, X. (2023). AI-driven adaptive feedback for novice programmers: Enhancing debugging and logic skills. *Journal of Educational Computing Research*, *61*(2), 412–433. https://doi.org/10.1177/07356331221128910

[33]- Lyu, W., Wang, Y., Chung, T., Sun, Y., & Zhang, Y. (2024). Evaluating the effectiveness of LLMs in introductory computer science education: A semester-long field study. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale* (pp. 1–10). Association for Computing Machinery. https://doi.org/10.1145/3657604.3662036

[34]- Malik, S. I., & Coldwell-Neilson, J. (2017). *Impact of a new teaching and learning approach in an introductory programming course. Journal of Educational Computing Research, 55*(6), 789–819. https://doi.org/10.1177/0735633116685852

[35]- Mhashi, M. M., and Alakeel, A. (2013), Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University. *Recent Advances in Modern Educational Technologies*. 15-24.

[36]- Milne, I & Rowe, G 2002, 'Difficulties in learning and teaching programming—views of students and tutors', *Education and Information technologies*, vol. 7, no. 1, pp. 55-66.

[37]- Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information Technologies, 7*(1), 55–66.

[38]- Morse, JM. (1991), 'Approaches to qualitative-quantitative methodological triangulation. *Nursing Research*. 40. 2. 120–123.

[39]- Nguyen, T., Pham, L., & Do, H. (2022). Enhancing problem-solving skills in introductory programming courses through guided workshops. *Computers & Education, 182*, 104528. https://doi.org/10.1016/j.compedu.2022.104528

[40]- Nnass, I., Cowling, M. A., & Hadgraft, R. (2022). Identifying the difficulties of learning programming for non-English speakers at CQUniversity and Sebha University. *Journal of Pure & Applied Sciences, 21*(4), 290–295. https://doi.org/10.51984/jopas.v21i4.2258

[41]- Pillay, N 2003, 'Developing intelligent programming tutors for novice programmers', *ACM SIGCSE Bulletin*, vol. 35, no. 2, pp.78–82.

[42]- Piteira, M., and Costa, C. (2012), Computer programming and novice programmers. *Proceedings of the Workshop on Information Systems and Design of Communication ACM*. 51-53.

[43]- Piteira, M., and Costa, C. (2013), Learning computer programming: study of difficulties in learning programming. *International Conference on Information Systems and Design of Communication*. 75-80.

[44]- Prather, J., Reeves, B. N., Denny, J., Leinonen, J., MacNeil, S., Luxton-Reilly, A., et al. (2024). Breaking the programming language barrier: Multilingual prompting to empower non-native English learners. *arXiv*. https://arxiv.org/abs/2412.12800

[45]- Saad, N. D. A. (2021). Challenges encountered by Libyan students learning through implementing English as a medium of instruction (EMI): A study in the Faculty of Engineering at Sirte University. *Abhat Journal*, 17, 561–572.

[46]- Schulte, C., and Bennedsen, J. (2006), What do teachers teach in introductory programming?. *ACM*. 17-28.

[47]- Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). A taxonomic study of novice programming summative assessment. In *Proceedings of the Eleventh Australasian Conference on Computing Education* (Vol. 95, pp. 147–156). Australian Computer Society.

[48]- Soloway, E & Spohrer, J 1989, *Studying the novice programmer*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 497.

[49]- Soloway, E 1986, 'Learning to program = learning to construct mechanisms and explanations', *Communications of the The Association for Computing Machinery (ACM)*, vol. 29, no. 9, pp.850–858

[50]- Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(2), 1–31.

[51]- Sorva, J. (2013). *The same but different: Students' understandings of primitive and object variables*. In Proceedings of the 13th Koli Calling International Conference on Computing Education Research (pp. 16–25). ACM.

[52]- Sorva, J. (2018). *Misconceptions and the beginner programmer*. In S. A. Fincher & A. V. Robins (Eds.), The Cambridge Handbook of Computing Education Research (pp. 531–559). Cambridge University Press. https://doi.org/10.1017/9781108654555.019

[53]- Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education, 13*(4), Article 40. https://doi.org/10.1145/2534973

[54]- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

[55]- Tan, J, Guo, X, Zheng, W & Zhong, M 2014, 'Case-based teaching using the laboratory animal system for learning C/C++ programming', *Computers & Education*, vol. 77, pp. 39-49

[56]- Tie, Z., Zhuang, H., Zhang, Q., and Wang, Z. (2012), Analysis on the relationship between student grades and computer programming time in learning the C programming language. *International Conference On Computer Science & Education (ICCSE)*. 1584-1589.

[57]- UNDP Libya. (2019, March 19). *Renovated labs motivate students in Sebha*. UNDP. https://www.undp.org/libya/stories/renovated-labs-motivate-students-sebha

[58]- Villegas Molina, I., Montalvo, A., Ochoa, B., Denny, P., & Porter, L. (2024, November 5). *Leveraging LLM tutoring systems for non-native English speakers in introductory CS courses* (Preprint). arXiv. https://doi.org/10.48550/arXiv.2411.02725

[59]- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press.

[60]- Wang, Y., Yue, Y., Wang, W., & Zhang, G. (2025). *Uncovering non-native speakers' experiences in global software development teams: A Bourdieusian perspective*. arXiv. https://doi.org/10.48550/arXiv.2501.06437

[61]- Winslow, LE 1996, 'Programming pedagogy a psychological overview', *ACM SIGCSE Bulletin*, vol. 28, no. 3, pp. 17-22.

[62]- Xinogalos, S. (2012), Programming techniques and environments in a technology management department. *ACM*. 136-141.

[63]- Xinogalos, S. (2014). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. Education and Information Technologies.

[64]- Yinnan, Z., and Chaosheng, L. (2012), Training for computational thinking capability on programming language teaching. *International Conference on Computer Science & Education (ICCSE)*. 1804-1809..