FROM SECURITY TO EDUCATION: CHOOSING PROGRAMMING LANGUAGES FOR CS1 AND CS2

CZU: [004.056:004.43]:378
DOI: https://doi.org/10.53486/tids2026.29

ZINOVIA TOACĂ
Academy of Economic Studies of Moldova,
e-mail: toaca@ase.md
ORCID ID: 0000-0002-8304-1961

LUCIA GUJUMAN
Academy of Economic Studies of Moldova,
e-mail: gujuman.lucia@ase.md
ORCID ID: 0000-0001-7940-4291

Abstract. One of the main problems in the process of developing and using computer systems is software security. C++ is one of the languages used in the development of high-performance applications and complex software infrastructures. This language provides direct control over memory and system resources, but this can lead to security vulnerabilities. The way to avoid these problems is to follow good practices in the programming process, such as using modern language mechanisms, validating input data, avoiding unnecessary manual memory management, using automatic analysis procedures. Security is not a final stage of the programming process, but a principle from the first line of code. The C++ language is in continuous development and one of the issues addressed is the security of the written code. Today, such mechanisms as RAI (Resource Acquisition Yes Initialization), smart pointers, modern standard libraries and advanced static code analysis tools, which are a significant support in code security, reducing the risk of memory errors and increasing software reliability. In this context, teaching the C++ language in university programs in the field of information technologies remains essential for training specialists capable of understanding the internal functioning of computer systems, resource management and the development of high-performance applications. The study of this language not only contributes to the development of algorithmic thinking, understanding computer architecture but also to the formation of the necessary skills for the development of critical systems. This article analyzes the experience of universities in selecting programming languages for CS1[1] and CS2[2] courses.

Keywords: C++, software security, memory vulnerabilities, buffer overflow, RAII, smart pointers, secure programming, computer science education.

JEL Classification: A23, C88, I23

INTRODUCTION

Society's increasing dependence on networked software systems is accompanied by an increase in their vulnerability. This leads to loss and compromise of sensitive data, system damage, loss of productivity and financial losses. There is increasing evidence that criminals consider vulnerabilities in software systems as a tool to achieve their goals, thus software vulnerabilities are discovered (Seacord, 2013) at a rate of over 4,000 per year. These vulnerabilities are caused by software designs and implementations that do not adequately protect systems, by development practices that do not focus enough on eliminating implementation defects and that lead to security errors. Currently there

[1] Computer science 1
[2] Computer science 2

is little evidence of improving the security of most products; many software developers do not understand the lessons learned about the causes of vulnerabilities or do not apply adequate mitigation strategies. CERT/CC (Computer Emergency Response Team Coordination Center) continues to observe the same types of vulnerabilities in newer versions of products that were discovered in previous versions. The same is confirmed by recent empirical studies (Gueye & Mell, 2021)

C/C++ programming languages are low-level and offer the programmer the ability to manage memory and hardware resources. This is one of the reasons why these languages are quite efficient for developing high-performance applications, thus being widely used (Stroustrup, 2015) in the development of operating systems, embedded systems, financial applications and critical software infrastructures.

Studying the C/C++ programming languages is necessary and useful in the process of training specialists (Aljas et al., 2023) in the field of information technologies, because it develops algorithmic thinking, deep understanding of memory, and promotes the development of good code writing practices. And this requires that attention to detail be paid to the learning process. Moreover, the experience of programming in C/C++ creates premises for the easy acquisition of other programming languages, such as Python, Java, C#.

1. Unsafe programming practices and common errors that can lead to vulnerabilities

The main causes of software vulnerabilities, such as buffer overflows, integer range errors, and invalid format strings, are improper code writing practices. According to CERT guidelines, one of the main causes of vulnerability in information systems is improper memory management (Seacord, 2013). The CERT C/C++ Secure Coding standard defines a set of rules that must be taken into account in the code writing process to prevent software vulnerabilities. In the paper (SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems (2016 Edition), nd) vulnerabilities are aggregated into 15 groups. It is noteworthy that each type of vulnerability is initially presented through non-compliant code, and then the solution is proposed, which leads to the development of less vulnerable code. Most of the vulnerabilities included in the source target basic, initial knowledge in C/C++ programming. This moment confirms once again that securing an application is not the final stage of the project, but it must be a concern from the first line of code written and these skills must be developed starting from the first lessons in the field. In **Error! Reference source not found.**, the most relevant rules have been selected, which must be implemented, developed to students in the process of teaching the subject "Computer Programming".

Table 1. CERT rules relevant to preventing vulnerabilities in C/C++

CERT rule code	Description	Vulnerability type prevented	Example of risk	Recommendation
EXP33-C	Don't use uninitialized variables	Undefined behavior	using a random value	Initialize all variables
MEM30-C	Do not access memory after it has been freed	Use-after-free	invalid pointer dereferences	Use smart pointers
MEM31-C	Frees dynamically allocated memory	Memory Leake	excessive memory consumption	Uses RAII
ARR30-C	Do not access elements outside the bounds of an array	Buffer overflow	buffer overflow	Check the dimensions
STR31-C	Provides enough space for strings	Buffer overflow	strcpy dangerous	Use safe features

Table 1 (Continued)

CERT rule code	Description	Vulnerability type prevented	Example of risk	Recommendation
INT30-C	Avoid numeric overflow	Integer overflow	incorrect values	Check numerical limits
EXP34-C	Do not dereference null pointers	Null pointer dereferences	application crash	Check pointers
FIO34-C	Check for errors in file operations	I/O errors	incorrect read/write	Error handling
ERR33-C	Detects and handles all errors	Logical vulnerabilities	incorrect execution	Use exceptions
MSC00-C	Write clear and secure code	General errors	hard-to-maintain code	Meets standards

Source: *(SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems (2016 Edition) , nd)*

2. Comparing the security of the main programming languages

In the process of developing modern programming languages, one of the main criteria in choosing the technologies used for application development is software security. In this context, languages such as Rust, Java, Python have been designed with additional mechanisms to protect against common classes of vulnerabilities, especially those associated with memory management. Therefore, a comparative analysis (**Error! Reference source not found.**) between these languages provides a useful perspective on the advantages and limitations of each language from a security point of view.

Table 2. Comparison of memory security mechanisms in major programming languages

Characteristic	C++	Rust	Java	Python
Memory management model	manual (new / delete) + RAII	ownership + borrowing	garbage collection	garbage collection
Detecting memory errors	mostly at runtime	mostly at compilation	runtime	Runtime
Typical vulnerabilities	buffer overflow, dangling pointers, use-after-free	very rare (verified at compilation)	low (memory isolation)	LOWER
Memory control	very high	high	Limited	very limited
Performance	very high	very high	high	Moderate
Implicit safety	moderate	very high	high	High
Modern protection mechanisms	RAII, smart pointers, sanitizers	borrow checker	sandbox, garbage collector	runtime safety
Main areas of use	operating systems, embedded, HPC	secure systems, distributed systems	enterprise applications	scripting, data science

Source: *Developed by the author based on (Seacord, 2013), (Klabnik & Nichols, nd), (Grgic et al., 2018; Gueye & Mell, 2021; Stroustrup, 2015)*

Comparing these languages highlights that memory security is directly influenced by the resource management model adopted by each language. C/C++ languages offer the highest level of control over memory, which allows for performance optimization, but also involves significant risks of vulnerabilities if good programming practices are not followed.

Rust introduces an innovative model based on ownership and borrowing, which allows for the detection of many memory errors at compile time. Java and Python use garbage collection mechanisms. collection, reducing the risk of vulnerabilities such as memory leaks or invalid pointer dereferences, but partially sacrificing control over resources.

3. Implications for university education

The dilemma of which programming language should be taken as a beginner CS1[3] and CS2-course is a concern of several researchers. It is obvious that there is no single answer. From the experiments exposed in (Ali & Qayyum, 2021) it was concluded that, if the goal is speed, efficiency and reliability, then C++ is the king of all languages, taking the lead from C, as it offers broader support for approaches due to the support for OOP (Object Oriented Programming), which is missing in C. When developing system software, such as drivers and operating systems, C holds strong compared to C++. Java is the middle ground among all languages; considering that C is complex and has the speed that Python lacks. The authors (Ali & Qayyum, 2021a) consider Java to be a good choice because it not only provides support for "Write Once Run Anywhere", but automatically applies optimizations, something that a novice programmer may not be familiar with. Python is best used if writing software where speed is not a major concern. It is also an excellent choice for beginners and for fields such as Data Science and Artificial Intelligence, due to the large number of libraries it offers. At the same time, the languages C, C++, Java are considered the most difficult to learn (Gomes *et al.*, 2026) .

In 2017, Murphy, Crick and Davenport (Murphy *et al.*, 2017) analyzed introductory programming courses offered in 80 undergraduate Computer Science programmers at UK universities, analyzing the languages, paradigms and tools used in these courses. Java appears to be the dominant language, being used in more than half of the courses, despite Python being considered more accessible for both teaching and learning and the fact that students are generally not experts in programming. The C family of languages (C, C++ and C#) is used in almost a third of cases. The predominant paradigms were object-oriented (50%), followed by procedural (33.8%) and finally functional (8.75%).

In 2018, the paper (Avoris, 2018) presented the results on higher education institutions in Greece, using a public database containing information on study programs, courses and recommended textbooks, with a focus on pedagogical approaches, programming languages and tools used. The C language was identified as the most used language in the analyzed programming courses, present in 37% of the courses, followed by MATLAB (17%) and Basic (11%). According to the results of various studies, Java and Python remain, however, the most popular languages for teaching beginner programmers (Siegfried *et al.*, 2021) . The same study shows that 88% of all schools surveyed use only one of the four programming languages (Java, Python, C++ and C), with Python adoption constantly increasing to the detriment of the other languages. Java continues to be the most popular choice for CS2 courses, followed by C++ but with a significant difference.

In 2024, (Mason *et al.*, 2024) researchers conducted a global study of introductory programming courses, involving instructors from 18 countries across six continents. The study identified Python and Java as the most commonly used programming languages, each accounting for approximately 40% of the courses analyzed. Other languages, such as C++ and C, also appeared with significant frequency (12% and 10%, respectively). JavaScript, Racket, Scala, Go, and Bash were also mentioned, but more marginally, the latter two being used in conjunction with another

[3] **CS1** and **CS2** are standard names used in computer science education for **the first two fundamental programming courses** in university.

programming language. Among the factors influencing language choice, pedagogical benefits were the most important, followed by platform independence and cost/accessibility for students. Industry relevance was also mentioned, but to a lesser extent than in previous regional studies. The study also revealed regional variations, with Python dominating the Americas and Java being more prevalent in Oceania. Regarding the paradigm taught, the results show: procedural (54%), followed by object-oriented (41%) and functional (5%).

For the study (Gomes *et al.*, 2026), institutions were selected using the 2023 Shanghai Ranking for Computer Science and Engineering. Finally, programming languages were successfully identified in 150 courses, including 83 CS1 courses and 67 CS2 courses, from a total of 83 institutions in 14 countries, representing four continents: North America, Asia, Europe and Oceania. According to the study, Java, Python, C and C++ are the programming languages most often chosen for the two initial courses. Java is the most used language in the two introductory programming courses, which are required in 55 (36.7%) of the courses analyzed. It is significant that it is the most used language in CS2, 38 courses – (56.7%) and the second - in CS1- in at least 16 (19.3%) courses. Of the 13 institutions that use the same language in both introductory courses, 10 use Java. Python is second in the choice ranking, but is the main choice for introductory programming courses, with at least 27 occurrences (32.5%). However, it is chosen less often for CS2, dropping to fourth place, with only 5 occurrences (7.5%). C and C++, sometimes used together, are the third and fourth most used programming languages, respectively, with a balanced presence in CS1 and CS2. As a required language, C appears in 27 courses (18.0%), being used 14 times in CS1 (16.9%) and 13 times in CS2 (19.4%). C++, on the other hand, is required in 19 courses (12.7%), with 10 in CS1 (12.7%) and 9 in CS2 (13.5%). Two institutions use C++ in both courses.

The transition between programming languages is also important. (**Error! Reference source not found.**). The most widespread transition is from Python to Java, occurring in 12 institutions, followed by C to Java (6 institutions), Python to C (5), and Python to C++ (4).

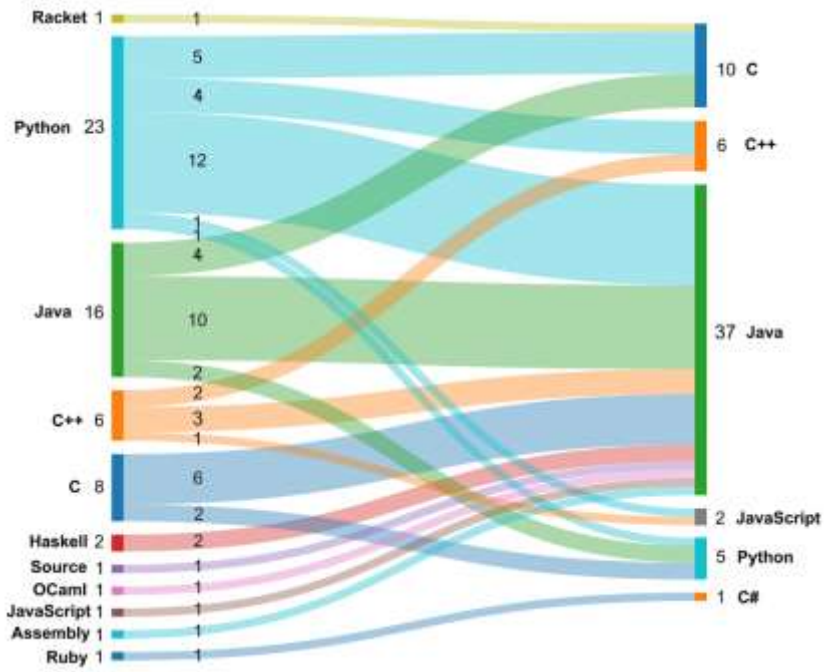


Figure 1. Programming language transition between CS1 and CS2 courses. The width of each band represents the number of institutions transitioning from one language to another.
Source: *(Gomes et al., 2026)*

4. Studying languages within CS1 and CS2 courses in the Republic of Moldova

Analysis of the UTM website the curriculum for specialties in the field 0613 Software and Application Development indicates that CS1 is the discipline "Computer Programming", where C/C++ is taught, and CS2 can be considered the disciplines "Applied Programming Techniques" in some specialties, and "Programming Techniques" in others, where the Python language is taught. The discipline "Object-Oriented Programming" is found in semester 3, but due to the lack of information on the site, it was not determined which programming language is taught.

Within USM , in the plans for programs in the same field in semester 1 there is the discipline "Fundamentals of Programming", and in semester 2 there are the disciplines "Object-Oriented Programming" and "Phyton for Applications". Due to the lack of information on the university website, it is not known which programming languages, except for the last discipline.

Teachers choose a programming language based on its features. For example, Python is valued for its relative ease and readability, Java is valued for its emphasis on object-oriented approaches, and learning C++ allows users to develop an understanding of memory management and resource utilization (Ali & Qayyum, 2021b). In a beginner programming course, best practices should be emphasized, as students' bad habits persist if not reinforced (Dougherty, 2023). The C++ programming language, although a relatively old language, has been updated with new versions regularly since 2011, at a rate of once every three years. Each new version contains important features that make C++ more complex for backward compatibility, but often introduces new features to simplify the process of developing computer projects while respecting cybersecurity requirements. (Seacord, 2020). Currently, the C++23 version is used, and the C++26 version is in the process of being developed.[4]

CONCLUSIONS

Pregnancy universities, which educate specialists in the field of information technologies is to educate primarily algorithmic thinking. In the current conditions, when cyber vulnerability is becoming increasingly important, the educational process must explore programming practices that will help students avoid developing bad habits and will allow them to develop secure programs during their professional careers. From the analysis presented, the programming languages C and C++ are the most vulnerable in this context. On the net textbooks can be found that address the fundamental programming errors in C and C++ that lead to the most common, dangerous, and disruptive software vulnerabilities recorded since the founding of CERT in 1988. The book "Secure Coding in C and C++" (Seacord, 2013) provides both an in-depth engineering analysis of the programming errors that led to these vulnerabilities, as well as mitigation strategies that can be applied effectively and pragmatically to reduce or eliminate the risk of exploitation. Such textbooks can serve as support for cybersecurity disciplines, both in undergraduate and graduate study programs, and would serve as significant support in formulating individual assignments for students, which would allow them to advance in the field, to become established specialists. In the process of teaching programming languages in the first year of a bachelor's degree, especially in the first semester, it is more complicated to pay special attention to security issues, especially when the teacher is primarily tasked with developing programming skills. It is necessary to mention that the skills of first-year bachelor's students to write code are very different: from "never wrote code" to "wrote code in languages such as C++, Java, Python". Some of those who have programming skills have quite in-depth knowledge.

[4]https://isocpp.org/std/status

The choice of programming languages taught in CS1 and CS2 depends on the objectives of the study program. Another important factor is the current requirements of the labor market. It is possible that this choice should also be influenced by the complexity of the language taught.

REFERENCES

Ali, S., & Qayyum, S. (2021). *A Pragmatic Comparison of Four Different Programming Languages*. https://doi.org/10.14293/S2199-1006.1.SOR-.PP5RV1O.v1

Aljas, J., Amores , H., & Lincopinis , D. (2023). *An Overview on C++ Programming Language*.

Avouris, N. (2018). Introduction to computing: A survey of courses in Greek higher education institutions. *Proceedings of the 22nd Pan- Hellenic Conference on Informatics*, 64–69. https://doi.org/10.1145/3291533.3291549

Dougherty, RE (2023). *Creation of a CS1 Course with Modern C++ Principles*. https://arxiv.org/abs/2311.07807

Gomes, JP, Monteiro, R., Moreira, AS, Mendonça, V., Pinho, A., & Cunha, CR (2026). Teaching introductory programming in top universities: A global study of languages, paradigms, assessment, and AI. *Journal of Computer Languages*, *86*, 101384. https://doi.org/10.1016/j.cola.2025.101384

Grgic, H., Mihaljevic, B., & Radovan, A. (2018). Comparison of garbage collectors in Java programming language. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1539–1544. https://doi.org/10.23919/MIPRO.2018.8400277

Gueye, A., & Mell , P. (2021). *A Historical and sTATISTICS Study of the Software Vulnerability Landscape*. https://arxiv.org/abs/2102.01722

Klabnik, S., & Nichols, C. *The Rust Programming Language*. https://doc.rust-lang.org/book/ch03-01-variables-and-mutability.html Accessed 09.02.2026

Mason, R., Simon, Becker, BA, Crick, T., & Davenport, JH (2024). A Global Survey of Introductory Programming Courses. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 799–805. https://doi.org/10.1145/3626252.3630761

Murphy, E., Crick, T., & Davenport, JH (2017). An Analysis of Introductory Programming Courses at UK Universities. *The Art, Science, and Engineering of Programming*, *1* (2), 18. https://doi.org/10.22152/programming-journal.org/2017/1/18

Seacord, RC (2013). *Secure coding in C and C++* (2nd ed.). Addison- Wesley.

Seacord, RC (2020). Effective C. *Computer*, *53* (11), 79–82. https://doi.org/10.1109/MC.2020.3016369

SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems (2016 Edition) . (n.d.)

Siegfried, RM, Herbert-Berger, KG, Leune , K., & Siegfried, JP (2021). Trends Of Commonly Used Programming Languages in CS1 and CS2 Learning. *2021 16th International Conference on Computer Science & Education (ICCSE)*, 407–412. https://doi.org/10.1109/ICCSE51940.2021.9569444

Stroustrup, B. (2015). *The C++ programming language: C++ 11* (4th ed., 4th print). Addison- Wesley.