2025

CS 4402

Comparative Programming Languages

LEARNING JOURNAL 4

SANA UR REHMAN

INSTRUCTOR: JIM CASALE

INTRODUCTION

This week's focus was on control structures and subprograms, which are essential for understanding how programming languages manage the flow of execution. I reviewed sequence, selection, and repetition structures, paying close attention to the differences between if statements, switch/case logic, and loop constructs such as while, do-while, and for. I also explored the concept of invariants and how they help ensure loop correctness. The second half of the unit concentrated on subprograms, including procedures, functions, and different parameter-passing mechanisms. In particular, I examined call by value versus call by reference and their implications for performance and security. Finally, I studied recursion and stack architecture, learning how recursion leverages the stack to manage multiple function calls. These topics collectively helped me understand how program control structures shape the efficiency and reliability of algorithms (Sebesta, 2016).

DIFFICULTIES FACED

One of the main challenges I faced was fully understanding how recursion works in relation to stack architecture. While I had encountered recursion before, visualizing how function calls are stored and popped from the call stack was initially confusing. Another difficulty came from distinguishing the subtle differences between pass-by-value and pass-by-reference. Although the concepts appear straightforward in theory, their implications in practice particularly regarding memory safety were more complex than I expected (Sebesta, 2016). Additionally, some self-quiz questions on loop invariants required deeper critical thinking, as they pushed me to articulate why certain properties must hold throughout iterations rather than simply recalling definitions.

ACTIVITIES PERFORMED

I completed the weekly readings and engaged in the discussion activity, where I explained the differences between pass-by-value and pass-by-reference. I highlighted the security risks of pass-by-reference, such as aliasing and memory vulnerabilities, and used merge sort as an example to illustrate the benefits of recursion for divide-and-conquer algorithms (Cormen, Leiserson, Rivest, & Stein, 2009). This activity reinforced my understanding because it required me to present technical concepts in a structured and clear way. I also completed the self-quizzes, which tested my ability to apply theoretical concepts to practical programming problems. The quizzes provided instant feedback, which helped me identify areas that needed more revision. These activities, combined with reading and note-taking, created a balanced learning experience that strengthened both my theoretical and applied understanding.

CONCLUSION

This week deepened my appreciation for the role of control structures and subprograms in programming languages. I realized that while recursion can seem abstract, it offers powerful ways to solve complex problems efficiently. I also gained more awareness of the trade-offs between call by value and call by reference, particularly the security risks inherent in the latter. Overall, I feel more confident in analyzing how programming languages implement flow control and data passing. One important takeaway is that understanding these core mechanisms not only improves my programming skills but also sharpens my ability to evaluate which programming paradigms best suit different problem domains (Sebesta, 2016; Cormen et al., 2009).

REFERENCES

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Sebesta, R. W. (2016). *Concepts of programming languages* (11th ed.). Pearson.

Wirth, N. (1976). *Algorithms + data structures = programs*. Prentice-Hall.

Wordcount: 477