2025

CS 4402

Comparative Programming Languages

LEARNING JOURNAL 3

SANA UR REHMAN

INSTRUCTOR: JIM CASALE

INTRODUCTION

In the third week of Comparative Programming Languages, I was introduced to a wide variety of data types and how they are utilized in various languages. I also learnt about primitive types: integers, characters, Boolean and enumerations in addition to composite structures such as records, arrays, strings and multi-dimensional arrays. I've also read about expressions, operator precedence, assignments, and the scope of variables. Low-level information was attended to in detail, such as binary and two's-complement notation, bitwise arithmetic, overflow conditions, and big- versus little-endian storage. Through these topics, I deepened my understanding of how programming languages handle data internally and how these choices affect performance and security (Patterson & Hennessy, 2021).

DIFFICULTIES FACED

One challenge this week was visualizing how multi-byte data values are arranged in memory under different endianness conventions. While reading about big and little endian helped conceptually, it took extra time to draw memory diagrams and trace byte order manually. Another difficulty arose with integer overflow: although I understood the theoretical limits of fixed-size integers, it was harder to predict overflow behavior in actual code examples across languages. Working through self-quizzes and a graded quiz forced me to check my assumptions and improve my ability to calculate wrap-around results correctly (Stallings, 2020).

ACTIVITIES PERFORMED

I participated in some activities that helped reinforce learning. To begin, I completed the weekly readings on expressions, data types, and storage models. I then completed a discussion post comparing big and little endian architectures and commenting on their trade-offs in terms of readability, performance, and network communication. In writing the post, I synthesized textbook and class note content rather than

copying example verbatim. I also performed self-quizzes on variable binding, composite data types, and bit operations and then the graded quiz, which challenged my knowledge under timed conditions. Collectively, these exercises cumulatively helped in translating theoretical knowledge into beneficial knowledge.

REACTIONS AND FEEDBACK

Making the discussion post caused me to think critically about why hardware designers and language implementers have varying choices. My instructor and peers gave feedback on my explanation of integer overflow as a bug, which made me confident that I had properly articulated the issue. They also gave me some other places where endianness does apply, such as file formats and encryption systems, which made me see things differently. This interaction helped me see the real-world impact of what at first appeared to be low-level details (Stallings, 2020).

FEELINGS AND LEARNING OUTCOMES

Initially, even the sheer number of new terminologies such as "two's complement," "binding," and "multi-dimensional arrays" overwhelmed me. However, as I progressed through examples and quizzes, I began to understand how all these pieces fall into place. I realized that not only am I learning facts but also a structured way of thinking around data representation and operations. One of the major takeaways was to enjoy how bitwise operations can improve performance in certain algorithms, and how appreciating storage order can prevent hidden bugs in cross-platform software (Patterson & Hennessy, 2021).

CONCLUSION

This week has been difficult but fulfilling. I am growing more comfortable comparing how programming languages specify and implement data types and how low-level attributes such as endianness and overflow affect software behavior at higher levels. In the future, I will apply these concepts by

examining the data structures in languages I am familiar with, experimenting with how they handle boundary cases, and thinking through portability and security implications.

REFERENCES

Patterson, D. A., & Hennessy, J. L. (2021). *Computer organization and design: The hardware/software interface* (6th ed.). Morgan Kaufmann.

Stallings, W. (2020). *Computer organization and architecture: Designing for performance* (11th ed.). Pearson.

Wordcount: 565