CS4402 Learning Journal

The Learning Journal is a tool for self-reflection on the learning process. In addition to completing directed tasks, you should use the Learning Journal to document your activities, record problems you may have encountered and to draft answers for Discussion Forums and Assignments. The Learning Journal should be updated regularly (on a weekly basis), as the learning journals will be assessed by your instructor as part of your Final Grade.

Your learning journal entry must be a reflective statement that considers the following questions:

* Describe what you did. This does not mean that you copy and paste from what you have posted or the assignments you have prepared. You need to describe what you did and how you did it.
* Describe your reactions to what you did
* Describe any feedback you received or any specific interactions you had. Discuss how they were helpful
* Describe your feelings and attitudes
* Describe what you learned

Another set of questions to consider in your learning journal statement includes:

* What surprised me or caused me to wonder?
* What happened that felt particularly challenging? Why was it challenging for me?
* What skills and knowledge do I recognize that I am gaining?
* What am I realizing about myself as a learner?
* In what ways am I able to apply the ideas and concepts gained to my own experience?

Finally, describe one important thing that you are thinking about about the activity.

I started my learning with the reading materials.

From the reading material, I gained some understanding of what is a programming language instead of seeing them as separate tools.

The programming languages share some common features in that they are all a set of rules that specify symbols to compute the description of the program.

Then I learned what is a stored-program computer and symbolic assembler.

In the abstraction mechanism, programming is an abstraction that allows a programmed specification to get executed.

I learned that the higher the abstraction the more detail is lost. This also reminded me to be cautious when it seems appealing to use highly abstract language for tasks that involve many details.

I also gained some light understanding of how Algol and Pascal evolved over the past 50 years. The Pascal is the descendent of Algol done by Niklaus Wirth. The C was developed by Dennis Ritchie in the early 1970s and standardized by ANSI in 1989.

Besides the traditional imperative languages, there are data-oriented languages like Fortran,

Lisp that focuses on list structure. APL focuses on the vector.

Those object-orientated languages that we are familiar with today are just one of the branches of the programming languages at that time.

During the working out of the discussion assignment, I gained knowledge of non-imperative languages like data-oriented languages and logic-orientated languages. Including the functional programming that is good performing at language.

Then I exposed the classical von Neuman computer architecture.

That connects the CPU, memory, and IO with the bus for communications.

I also learned how they manage the performance issue by introducing a cache system. This is resolved by the hierarchy of memory that combines the registers, cache, and main memory with the disk. That layer can load the late before its actual execution.

I think the most important concept is the computability thesis which supports the development of computers and programs. It answers the most basic question of what can be computed.

According to the Church-Turing Thesis, any computation that can be described can be programmed in a turning machine.

Overall, I gained some understanding and intuitive understanding of different programming paradigms and their use cases.