Lisp Evaluation

Riley Fiske – November 1st, 2023

***Comparison to other Languages***

In comparison to more traditional languages explored this semester, Lisp has certainly been a unique language to work with. The first thing to probably draw attention to is the way functions are called and their argument values given. In Java, Python, Fortran, or Pascal, when you call a function, you give the function name followed by parentheses that contain the arguments for that function. In Lisp, every statement goes inside a set of parentheses with the first item being the function name followed by arguments separated by spaces ending with a closing parenthesis. This very much removes any ambiguity of where statements begin and end, but when nesting statements, the parentheses can get quite jumbled and not as easily readable, just as when writing nested statements in mathematics.

This project had us look at arithmetic operators and creating an array reversal algorithm. Another way Lisp varies from standard programming languages is the way in which arithmetic expressions are written. Previously in this class, we have talked about prefix, postfix, and infix operators, and Lisp takes this to the extreme and treats the syntax of arithmetic operations the same as the syntax for calling functions, by prefixing operators. This means if a programmer wants to add two values, they must first declare addition, then provide the arguments following the plus sign. Incremental operators such as var++, var--, ++var, or --var are used in other languages as prefix-postfix operators, but Lisp does this for all arithmetic operators. It is a little unnatural to write at first, but one adjusts quite quickly. One advantage I have found through this exploration in terms of the arithmetic operators is that more than two arguments can be given in the statement, meaning it is possible to sum together more than two values with one sum operation, meaning it is quicker to write that sort of operation in Lisp than many other languages.

***Programs vs. Data***

In most languages, programs contain code that is executed and processes and manipulates data objects within it. Lisp is different in that programs are lists and lists are how every data object is stored in Lisp, meaning programs are data. This makes Lisp quite unique in that the same ways you can manipulate internal data structures can be used to manipulate entire programs. According to a [Stack Overflow exchange on this topic](https://stackoverflow.com/questions/5833033/in-lisp-code-is-data-what-benefit-does-that-provide), some ways this can be used is to have programs change themselves or other programs or to write compilers trees.

***Screenshot of Lisp Program Execution:***

***A screenshot of a computer program

Description automatically generated***