Prolog Paradigm

Explanation of Prolog and logic programming

Prolog is an abbreviation of ‘Programming in Logic’. It’s logic programming language which is a paradigm and a language that is not often used in the modern day. The flow of logic programming defined by statements of code. Each line is either true or false and where false, Prolog looks for the next true statement. Where in traditional paradigms, you have functions and procedures, Prolog has predicates which again, equate to true or false. Predicates are facts and rules you define for the program and you can make as many instances of them as you want. If statements, looping, and other standard practices are not present in Prolog and so, you make a new version of your predicate that will equal true.

How my code conforms to the logic paradigm

My code implements logic programming to a professional degree with the use of predicates that fail and try the next, predicates that implement recursion, and the inclusion of asserts to memory. While I limited my use of built in Prolog predicates, of which there are plenty, I made use of some, notably ‘nth0/1’, ‘assert/1’, and ‘format/2’ that I considered necessary, from my knowledge and the help of the SWI Prolog documentation.

Why I chose Prolog

Bluntly, I chose Prolog due to my familiarity with the language from months of use during my placement year. Logic programming is a difficult challenge but one I knew I could implement for my game due to the paradigm’s step by step process, a process which makes sense for a game follow. Prolog has different implementations but I chose SWI Prolog due to it’s free and open source, feature rich approach.