The logic programming language \( \lambda \)Prolog extends Prolog by containing polymorphic types, higher-order programming, \( \lambda \)-terms as data structures, higher-order unification, modules, lexical scope, abstract data types, and implicational and universally quantified goals. These extensions are integrated into a single logical system and provide \( \lambda \)Prolog with expressive strengths not found in Prolog. After surveying various aspects of this language, we shall focus on its uses in specifying proof systems and implementing theorem provers. In particular, we shall discuss how the syntax and inference rules of object logics can be represented and on how tactic-style theorem provers can be written. We shall also illustrate how proofs can be built and manipulated in rather natural ways.