The paradigm of functional programming has branched off from type theory and lambda calculus. However, the latter offer a starting point for analyzing issues concerning design of functional programming languages, spanning efficiency and expressiveness as well as addressing decision problems within their convertibility relations. A novel idea in the lambda calculus is the notion of type isomorphism: Two types $A, B$ are isomorphic if and only if there is an invertible program $F$ such that $A =_F B$. Similarly, we make the following definition; two programs $P, Q$ are program isomorphic if and only if there is an invertible program $F$ such that $FP = Q$. We will explain how these relations provide lenient congruence relations on types and programs; specifically, they provide a suitable framework for program transformations. (Received September 28, 2005)