Funktionell programmering

Funktionell programmering(FP) innebär att programmet inte befinner sig i ett visst tillstånd. Samma sak utförs varje gång och det finns inget minne av vad som hänt tidigare gånger. Programmet är en matematiskt funktion i sig som består av en eller flera matematiska funktioner. Programmet tar ett ingångsvärde och räknar ut ett utgångsvärde. Man ändrar inte heller värden på variablerna som man tex gör i objektorienterad programmering. Istället skapar man en ny variabel om det behövs. I FP fokuserar man på vad som ska ske och inte hur det ska ske.

Fördelarna med FP är att det blir mer stabilt program då det inte finns sidoeffekter och tillstånd som påverkar. Det är enklare att avgöra om programmet gör det som man vill och det lämpar sig bra i automatiserade sammanhang. Det är lätt att förstå koden om man jämför med objektorienterade språk där det är svårare att följa koden.

Nackdelarna är att det är svårt att få en bra överblick i ett stort program. Det blir också mer ineffektivt när man använder datastrukturer. Det kan vara svårare att lära sig programmeringsstilen.

Två kända språk som använder FP är Lisp och Haskell och på senare tid F#. Många tror att FP kommer användas mer och mer i framtiden speciellt i dagens samhälle då mycket hela tiden är uppkopplat till molnet och buggar måste elimineras. Och då FP är säkrare är ju detta bra ur den aspekten. Värt att nämna är att många objektorienterade språk har stöd för funktionell programmering och vice versa.