Fundamentals of functional programming

ian.mcloughlin@gmit.ie

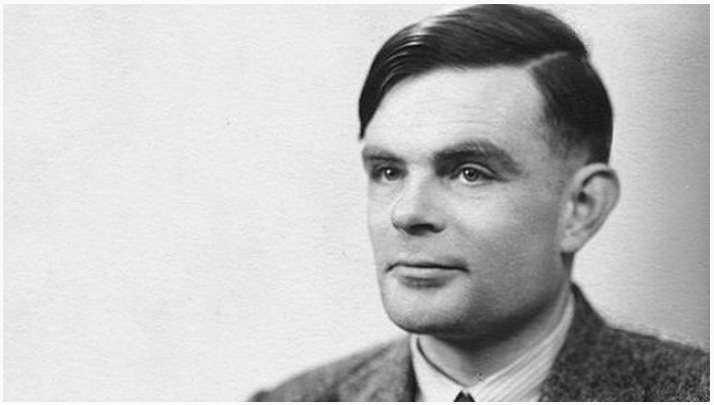
Functional programming

- Functional programming is a programming paradigm.
- Alonzo Church introduced lambda calculus in 1932:

$$\lambda f.(\lambda x.f(xx))(\lambda.f(xx))$$

- Functional programming is based on lambda calculus.
- Church was Turing's supervisor.
- Church–Turing thesis: lambda calculus **is** computation.

Imperative programming

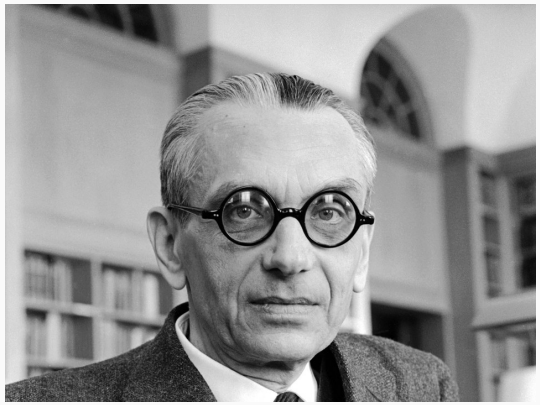


- Functional contrasts with imperative programming.
- C, Java, JavaScript are largely imperative.
- Programs have **state**, modified by **statements**.
- Origins in the Turing machine, a conceptual model created by Alan Turing.
- Turing machines and lambda calculus are equivalent.

von Neumann architecture

- Modern computers largely based on the von Neumann architecture
- Designed in the main by John von Neumann.
- Key facet of von Neumann architecture is that data (state) and instructions (statements) are stored in the same memory space.
- Begs the question: *what's the difference between the 0's and 1's that mean stuff and the 0's and 1's that do stuff?*

Unsolvable problems



- Some problems are not solvable by any computer.
- Turing showed this with Turing machines, and Church with lambda Calculus.
- Worked on what's come to be known as the Entscheidungsproblem, German for **decision problem**.
- No algorithm exists to check if another algorithm always finishes in a finite time, usually known as the **halting problem**.

Lisp

- Church's lambda calculus was based on simple rules.
- In the 1950's, John McCarthy created *Lisp*, the first functional programming language.
- McCarthy is also famous for coining the term *artificial intelligence*.
- Easy to write an interpreter – many dialects similar to Lisp, like Scheme and Racket.

Expressions in λ-calculus

Programs in Lisp are like λ-calculus expressions. Every expression is either:

x	a variable
(MN)	an application
$\lambda x.M$	an abstraction

Why study functional programming?

- Functional programming languages and ideas have had a resurgence in the last few years.
- One reason is that functional programming lends itself to parallel programming.
- Another reason is that artificial intelligence has its roots in functional programming.
- The main reason to learn it is because it makes you think about what programming is, or even better, what computation is.
- Note that a lot of imperative programming languages have recently reverse engineered functional ideas.

Some concepts for further discussion

- Anonymous functions
- First class functions
- Higher order functions
- Side effects
- Recursion
- Map and reduce