Haskell presentation

Skye & Ollie

CCMI

December 1, 2025

Contents

Introduction

Haskell is a purely functional programming language.

- Lazy evaluation
- Strong type system
- Declarative style

This makes it ideal for critical applications.

Sytnax & types

Yeah no idea - you do what you think is best here.

How to use

- **Glasgow Haskell Compiler**- Most widely used Haskell complier which supports optimisations and interactive REPL (ghci)
- **Linting** - Tools like hlint suggest idiomatic improvements
- **Hackage** - Central package repository for Haskell libraries which is managed via cabal or stack

Landin’s knot

Landin’s knot is a technique for implementing recursive structures without explicit recursion which is often used when implementing cyclic data structures or self-referential closures.

- **Manual Memory management** - In Haskell Garbage collection automatically handles cycles safely whereas in C++ cyclic structures risk memory leaks
- **Dangling pointers** - References are managed by the runtime system as Haskell uses no raw pointers. In C++, incorrect pointer handling can cause faults!
- **Complexity of implementing laziness**. Laziness is built into Haskell whereas manual tricks are needed in C++ which makes it error prone.

In Haskell, this is safe however in C++, it can fail at runtime!

Landin’s knot - code snippet

We show the syntax for creating two mutually dependent lists.
Syntax:

```
-- Create two lists that refer to each other
let xs = 0 : ys
    ys = 1 : xs

take 10 xs
-- Output: [0,1,0,1,0,1,0,1,0,]
```

Applications

Applications 2 if needed

Conclusion