Introducao

Descricao do Objectivo

O objectivo deste trabalho e mostrar vantagens de Programação Funcional, explicar porque sao vantagens, como se podem aplicar ou implementar em `C++`, e por fim, comparar `C++` com uma linguagem funcional, `Haskell`.

Descricao do que vamos detalhar no relatorio

Os aspectos de Programação Funcional que vamos detalhar sao:

- Imutabilidade
- Lazy Evaluation
- Composicao
- ADTs

Destes, composicao e, arguably (discutivelmente?), o mais importante e talvez o unico inerente a Programação Funcional. A ideia central de Programação Funcional e que construindo pecas pequenas, faceis de entender, e de provar como correctas, e tambem facil construir um sistema, mesmo que complexo, correctamente.

De seguida, imutabilidade, a ideia em que objectos nao sao alterados, mas copiados, para implementar mudancas. Esta propriedade ajuda a evitar erros comuns em Programação Imperativa, causados pela partilha de memoria, e a nao especificacao da relacao entre estados.

Lazy Evaluation, nao sendo adoptada como estrategia de avaliacao, pode ser usada como estrategia de optimizacao, especialmente quando combinada com imutabilidade e partilha de memoria.

Finalmente, ADTs sao um forma de definir formalmente novos tipos de dados a partir de tipos ja existentes. Apesar de nao essenciais a Programação Funcional, e desejavel criar abstraccoes no sistema de tipos que nos ajudem a descrever o problema em maos, dando significado a valores, e tentando limitar o conjunto de valores possiveis aos estritamente validos.

Para cada um destes, vamos mostrar exemplos de como se faz em `Haskell`, e como se pode fazer em `C++`. A unica excepcao e Lazy Evaluation, visto que em `Haskell` e adoptada como estrategia de avaliacao; neste caso, nao ha necessidade de mostrar como se faz em Haskell.