O Paradigma Funcional de Programação

Aspectos básicos da linguagem Haskell

Programação funcional

- Programa = conjunto de definições (de funções)
- Programar = construir funções para resolver um dado problema
- Funções que obdecem aos princípios matemáticos (embora possam não ser funções totais).
- O interpretador (da linguagem funcional) actua como uma calculadora: lê uma expressão, calcula o seu valor e mostra o resultado (*read – evaluate – print loop*).
- As expressões com ocorrências de nomes de funções definidas no programa, são calculadas usando as definições como regras de cálculo.

Definições

- Uma definição associa um *nome* a um *valor*.
- Um conjunto de associações *nome-valor* dá-se o nome de ambiente ou contexto.
- As expressões são calculadas no âmbito de um contexto e podem conter ocorrências dos nomes definidos nesse contexto.
- O interpretador usará as definições associadas a esses nomes como regras para simplificar (calcular) o valor da espressão.

Um exemplo de um programa:

```
dobro x = x + x
triplo y = 3 * y
a = 25
quadruplo x = dobro (dobro x)
```

Após carregar este programa no interpretador Haskell, podemos fazer os seguintes testes:

```
Main> dobro 3
6
Main> triplo a
75
Main> dobro (triplo 4)
24
Main> quadruplo 8
32
```

Transparência referencial

- No paradigma funcional, as expressões:
  - são a representação concreta da informação;
  - podem ser associadas a nomes (definições);
  - denotam valores que são determinados pelo interpretador da linguagem.

- No âmbito de um dado contexto, todos os nomes que ocorrem numa expressão têm um valor único e imotável.

- O valor de uma expressão depende unicamente dos valores das sub-expressões que a constituem, e essas podem ser substituidas por outras que possuam o mesmo valor.

A esta caracteristica dá-se o nome de transparência referencial.

Linguagens funcionais

- O nome de linguagens funcionais advém do facto de estas terem como operações básicas definição de funções e a aplicação de funções.

- Nas linguagens funcionais as funções são entidades de *1^a classe*, isto é, podem ser usadas como qualquer outro objecto: passadas como parametro, retornadas como resultado, ou mesmo armazenadas em estruturas de dados. Isto dá às linguagens funcionais uma grande flexibilidade, capacidade de abstração e modularização do processamento de dados.

- As linguagens funcionais fornecem um alto nivel de abstração, o que faz com que os programas funcionais sejam mais pequenos, mais claros e mais rápidos de desenvolver do que programas imperativos.

- No entanto, os programas funcionias podem ser menos eficientes.

Um pouco de história ...

1960s **Lisp** *(untyped, not pure)*

1970s **ML** *(strongly typed, type inference, polymorphism)*

1980s **Miranda** *(strongly typed, type inference, polymorphism, lazy evaluation)*

1990s **Haskell** *(strongly typed, type inference, polymorphism, lazy evaluation, ad-hoc polymorphism, monadic IO)*

Haskell

O Haskell é uma linguagem puramente funcional, fortemente tipada, e com um sistema de tipos extremamente evoluido.

A linguagem usada nesta disciplina é o **Haskell 98**.

O interpretador/compilador de Haskell 98 usado será o **GHC** Glasgow Haskell Compiler

www.haskell.org/ghc/