Panorama histórico

Gustavo Zambonin
Paradigmas de Programação (UFSC-INE5416)

Questão 1

- Seja uma função x e um argumento y. O resultado da função aplicada ao argumento retorna um elemento semelhante a uma árvore binária, onde cada folha é um combinador **S**, **K** ou **I**. Todas as operações neste tipo de cálculo-λ são expressadas dessa maneira.
  - Combinador **S**: leva três argumentos e retorna o primeiro argumento aplicado ao terceiro, e isto é aplicado ao segundo argumento aplicado ao terceiro. Em jargão matemático: $\mathbf{S}xyz = xz(yz)$.
  - Combinador **K**: retorna a função constante de um argumento, que uma vez aplicada a qualquer argumento, retorna seu próprio argumento. Em jargão matemático: $\mathbf{K}xy = x$.
  - Combinador **I**: é um combinador utilizado apenas para conveniência. Retorna a identidade da função. Em jargão matemático: $\mathbf{I}x = x$.

Questão 2

- A tese de Church-Turing define o conceito de funções computáveis. A noção de computabilidade, ignorando limitações físicas e temporais, denota que uma função é computável se, e apenas se, for computável por uma máquina de Turing. Este construto teórico é matematicamente semelhante ao cálculo-λ de Church e funções μ-recursivas de Gödel, outras maneiras de provar a computabilidade de funções. Formalmente, a tese não pode ser provada, pois estas coincidem com a maneira informal de uma função efetivamente calculável (termo utilizado antes da definição de função computável), que não apresenta uma prova formal.

Questão 3

- Muitos tipos de paradigmas existem e seria inviável descrevê-los todos. Presentes nesta lista estão alguns dos mais utilizados e conhecidos. Linguagens como Python podem apresentar características multi-paradigma e darão espaço a outras com paradigmas mais definidos.
- Programação **declarativa** expressa a lógica computacional sem descrever seu fluxo de controle. Analogamente, programas são teorias de lógica formal, enquanto computações são deduções neste espaço. Implementações deste paradigma geralmente preocupam-se no quê o programa deveria resolver, ou seja, descrever seus resultados desejados, ao invés de como ele deveria fazê-lo.
  - Programação **funcional** geralmente utiliza expressões para resolver os problemas, tratando computação como a resolução de funções matemáticas, ou seja, o resultado de uma função depende apenas de seus argumentos. Pode ser inferido que é difícil trabalhar com dados voláteis neste paradigma. Exemplos de linguagens: Erlang, Haskell, e dialetos de Lisp como Common Lisp e Clojure.
  - Programação **lógica** baseia-se, como o nome diz, em lógica formal. Um programa escrito com este paradigma geralmente é um conjunto de sentenças expressando fatos e regras sobre algum problema. Exemplos de linguagens: Prolog e suas diversas implementações.
- Programação **imperativa** descreve a computação em termos de sentenças que mudam o estado de um programa, analogamente ao modo imperativo de linguagens naturais em um diálogo. Foca em como o programa vai operar, quais caminhos ele tomará. Exemplos de linguagens: ALGOL, C (e suas derivações), COBOL, FORTRAN, Go, Java, Julia, Lua, MATLAB, Pascal, Perl, PHP e Ruby.
- Programação **paralela** é o paradigma dominante na área de arquitetura de computadores, especialmente na forma de processadores multi-*core*. O princípio se baseia na divisão de grandes problemas em problemas menores, resolvidos ao mesmo tempo, para ganho de desempenho e menor gasto de energia. Exemplos de linguagens: Scala, Smalltalk e Verilog.
- Programação **estruturada** foca na combinação de três estruturas de controle: sequência, onde a ordem das rotinas importa; seleção, que executa certas porções de código dependendo do estado do programa; e iteração, onde um bloco de código ou sentença é executada até que um estado mude. É possível escrever um programa estruturado em qualquer linguagem de programação.

Questão 4

- A lei de Moore observa que o número de transistores em um circuito tende a dobrar aproximadamente a cada dois anos. A indústria utiliza esta lei como planejamento de longo termo. Entretanto, a partir das células lógicas de 22 nanômetros, esta evolução tem tomado dois anos e meio, mostrando uma demora mais longa em desenvolver tecnologias menores, por conta do limite espacial dos elementos químicos que compõem as células.
- Richard Feynman, um grande físico do século XX, teve ideias muito à frente de seu tempo sobre nanotecnologia, e contribuiu conceptualmente, mas diretamente, para a evolução dessa ramificação da tecnologia, tornando-a muito mais comum nos tempos atuais.
- Peter Shor é conhecido primariamente por formular um algoritmo de fatoração de números inteiros utilizando computadores quânticos, que funciona exponencialmente mais rápido do que o melhor algoritmo de fatoração clássico existente.

Questão 5

- Um *qubit* é um sistema quântico de dois estados (ou dois níveis) que pode ser utilizado como elemento básico de memória, guardando até 2 bits de informação. Utilizando uma propriedade fundamental da mecânica quântica chamada superposição, um qubit pode estar nos dois estados lógicos, 0 e 1, ao mesmo tempo, processando informação de maneira mais rápida.

Questão 6

- O *D-Wave* é o primeiro computador quântico comercialmente disponível. Ele opera sobre uma célula de 128 qubits de extensão. Desde seu lançamento, novas versões com mais capacidade de processamento foram lançadas, utilizados por entidades como Google e NASA em seus laboratórios.