Teórica FP de testes 2012/13/14

Um **processo computacional** é um ente imaterial que existe dentro de um computador durante a execução de um programa, e cuja evolução ao longo do tempo é ditada pelo programa.

A **programação funcional** baseia-se no conceito de que os programas são funções que utilizam os valores produzidos por outras funções. A programação funcional não contém a instrução de atribuição nem estruturas explícitas de repetição.

Características dos paradigmas de programação:
Em **programação imperativa**, um programa é considerado como uma sequência de instruções, cada uma das quais produz um efeito. A programação imperativa depende da instrução de atribuição e da utilização de ciclos.
A **programação por objetos** baseia-se na utilização de objetos, entidades com estado interno associados a um conjunto de métodos que manipulam esse estado.
A **Programação funcional** baseia-se na utilização de funções que devolvem valores que são utilizados por outras funções. Em programação funcional as operações de atribuição e os ciclos podem não existir.

Um **algoritmo** é uma sequência finita de instruções bem definidas e não ambíguas, cada uma da qual pode ser executada mecanicamente num período de tempo finito com uma quantidade de esforço finita.
(É uma sequência de passos, bem definida e sem ambiguidades, que, sendo seguida mecanicamente, garante atingir um dado objetivo)

Com as seguintes características:

- Um algoritmo é rigoroso. Cada instrução do algoritmo deve especificar exata e rigorosamente o que deve ser feito, não havendo lugar para ambiguidade.
- Um algoritmo é eficaz. Cada instrução do algoritmo deve ser suficientemente básica e bem compreendida de modo a poder ser executada num intervalo de tempo finito, com uma quantidade de esforço finita.

- Um algoritmo deve terminar. O algoritmo deve levar a uma situação em que o objetivo tenha sido atingido e não existam mais instruções para ser executadas.

Um **objeto** é uma entidade computacional com estado interno e com um conjunto de operações, os métodos, que manipulam esse estado interno. Os objetos estão organizados numa hierarquia de classes, subclasses e instâncias, à qual se aplica o conceito de herança.
A herança consiste em transmitir o estado interno e os métodos associados a uma classe a todas as suas subclasses, exceto se esse estado interno ou esses métodos forem explicitamente alterados numa subclasse.

Desenvolvimento de um programa:

- ***Análise do problema:*** O programador, juntamente com o cliente, estuda o problema a resolver com o objetivo de determinar exatamente o que o programa deve fazer.
- ***Desenvolvimento de uma solução***: Determinação de como vai ser resolvido o problema. Desenvolvimento de um algoritmo e definição abstrata dos tipos de informação usados. Deve usar-se a metodologia do topo para a base.
- ***Codificação da solução***: Tradução do algoritmo para uma linguagem de programação, e implementação dos tipos de informação. Depuração, i.e., correção de erros sintáticos e semânticos.
- ***Testes:*** Definição de uma bateria de testes com o objetivo de "garantir" que o programa funciona corretamente em todas as situações possíveis.
- ***Manutenção:*** Fase que decorre depois do programa estar em funcionamento. A manutenção é necessária por dois tipos de razões: a descoberta de erros ou a necessidade de introduzir modificações e atualizações nas especificações do programa.

A **abstração de dados** consiste em separar a o modo como os dados são utilizados do modo como os dados são representados. Para isso definem-se camadas conceptuais lidando com cada um destes aspetos estando estas camadas separadas por "barreiras de abstração" que definem o modo como os programas acima da barreira podem comunicar com os programas que se encontram abaixo da barreira. Idealmente, os

programas que se encontram a um dado nível de abstração contêm toda a informação necessária para lidar com um certo tipo de dados, a informação está "encapsulada" dentro desta camada conceptual, e escondem das restantes partes do programa o modo como a informação está representada, o que é conhecido por "anonimato da representação".

A **abstração procedimental** corresponde a abstrair o modo como uma função realiza o seu trabalho, considerando apenas o que ela faz.
Ao desenvolver um programa, identificam-se os principais problemas que este tem que resolver, especificando-se funções que realizam esse trabalho e sem entrar nos detalhes do modo como elas realizam o seu trabalho. Depois de escrita uma primeira versão do programa recorrendo à abstração procedimental, aborda-se o desenvolvimento de cada uma das funções especificadas utilizando o mesmo método.

Metodologia dos tipos abstratos de informação:

- Identificação das operações básicas
- Axiomatização das operações básicas
- Escolha de uma representação para os elementos do tipo
- Realização das operações básicas para a representação escolhida

Esta metodologia garante a abstração de dados no sentido em que as operações básicas, que definem o comportamento do tipo de informação, são definidas antes da escolha da representação para o tipo.

Métodos de passagem de parâmetros:
Os parâmetros formais são os argumentos especificados na definição de uma função e os parâmetros concretos são os valores que são usados na invocação de uma função.

- Na **passagem por valor**, o parâmetro concreto é avaliado e o seu valor é associado com o respetivo parâmetro formal. A passagem por valor é um mecanismo unidirecional, do ponto de chamada para a função.
- Na **passagem por referência** a localização de memória da entidade correspondente ao parâmetro concreto é fornecida ao parâmetro formal. Na passagem por referência o parâmetro concreto e o parâmetro formal partilham a mesma entidade na memória do computador.

Os **erros sintáticos** correspondem ao facto de um programa não estar de acordo com as regras definidas para a sua sintaxe, por exemplo a utilização da expressão (+ x y) para somar os valores de x e de y.
Os **erros semânticos** correspondem ao facto de uma dada parte do programa, embora sintaticamente correta, não corresponder ao significado que o programador pretendia, por exemplo, escrever (x + y) quando se pretendia multiplicar os valores de x e y.