FUNDAMENTOS DA PROGRAMAÇÃO

→ Um **algoritmo** é uma sequência finita de instruções bem definidas e não ambíguas, a ser executadas num período de tempo finito com uma quantidade de esforço finita. Com isto, um algoritmo deve ter as seguintes características:
  - Ser rigoroso, devendo especificar exatamente o que deve ser feito;
  - Ser eficaz, devendo cada instrução ser suficientemente básica de modo a poder ser executada num espaço de tempo finito, com quantidade de esforço finita;
  - Ser finito, devendo levar a uma situação em que o objetivo é atingido e não haja mais instruções a executar.

→ A **representação externa** de uma entidade corresponde ao modo como nós visualizamos essa entidade, independentemente do modo como esta é representada internamente no computador – **representação interna**.

→ Uma **expressão composta** corresponde ao conceito de aplicação de uma operação a operandos, através de operações embutidas (pré-definidas/primitivas). Os operadores podem ser unários (se apenas têm um operando, por exemplo, o operador not ou o operador – (simétrico)) ou binários (se têm dois operandos, por exemplo, + ou *).

→ Os **tipos de informação** podem dividir-se em dois grupos:
  - Elementares – são caracterizados pelo facto de as suas constantes (os elementos do tipo) serem tomadas como indecomponíveis (ao nível da utilização do tipo). [Ex.: “verdadeiro”, “falso”].

    Como tipos elementares, existem, entre outros, o tipo inteiro, o tipo real e o tipo lógico.
    - Tipo inteiro – designado por *int*, são números sem parte decimal, podendo ser positivos, negativos ou zero.
    - Tipo real – designados por *float*, são números com parte decimal, podendo ser representados com notação decimal ou notação científica.
    - Tipo lógico – designado por *bool*, apenas pode assumir dois valores True (verdadeiro) ou False (falso). As operações que se podem efetuar sobre os valores lógicos, produzindo valores lógicos, são de dois tipos, as operações unárias e as operações binárias.

        - Operações unárias – produzem um valor lógico a partir de um valor lógico. Existe uma operação unária em Python, not. A operação not muda o valor lógico.
        - Operações binárias – aceitam dois argumentos do tipo lógico e produzem um valor do tipo lógico. Entras as operações encontram-se as operações lógicas tradicionais correspondentes à conjunção e à disjunção. A conjunção and tem o valor True, apenas se ambos os seus argumentos tiverem valor True. A disjunção or tem valor False, apenas se ambos os seus argumentos têm valor False.
    - Estruturados – são caracterizados pelo facto de as suas constantes serem constituídas por um agregado de valores.

→ Em **notação BNF**, um programa em Python, também é conhecido por um guião, é definido do seguinte modo:

```
<programa em Python> ::= <definição>* <instruções>
```

→ Um **programa** não contém diretamente expressões, aparecendo estas associadas a definições e a instruções.

→ As **linguagens de programação** fornecem estruturas que permitem especificar qual a ordem de execução das instruções do programa.

→ Ao nível da linguagem máquina, existem dois tipos de estruturas de controle: a **sequenciação** e o **salto**.

- A sequenciação especifica que as instruções de um programa são executadas pela ordem em que aparecem no programa.
- O salto especifica a transferência da execução para qualquer ponto do programa.

→ Os **parâmetros formais** são os argumentos especificados na definição de uma função e os **parâmetros concretos** são os valores utilizados na invocação de uma função.

→ Os **métodos de passagem de parâmetros** permitem:

- 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.
- Passagem por referência – a localização em memória da entidade correspondente ao parâmetro concreto é fornecida ao parâmetro formal. Na passagem por referencia o parâmetro concreto e o parâmetro formal partilham a mesma entidade na memoria do computador.

→ Um **processo computacional** é um ente imaterial que existe dentro de um computador durante a execução 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.

→ Em **programação imperativa**, um programa é constituído por uma série de ordens dadas ao computador. A programação imperativa depende da instrução de atribuição e da utilização de ciclos.

→ A **programação funcional** baseia-se na utilização de funções 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.

→ A **programação por objetos** baseia-se na utilização de objetos, entidades com estados associados a um conjunto de métodos que manipulam esse estado.

- Um **objeto** é uma entidade com estado interno cujo comportamento é ditado por um conjunto de métodos pertencentes ao objeto. Estes 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.

→ No uso de **tipos abstratos de dados** (TADs) não é necessário que o programador conheça a representação interna dos elementos do tipo, já que os TADs visam precisamente garantir a independência entre as funções básicas e a representação dos elementos tipo. Assim, a **abstração de dados** consiste em separar o modo de representação de dados da sua utilização.

→ A **abstração procedimental** corresponde à abstração do modo com uma função desenvolve o programa, focalizando-se, apenas, no resultado da função. Ao desenvolver um programa, identificam-se os principais problemas a resolver, são, então, especificadas funções que resolvem o problema, no entanto não é especificado o modo como a função corre. Depois de escrita a primeira versão do programa, recorrendo à abstração procedimental, aborda-se o desenvolvimento de cada uma das funções especificas utilizando o mesmo método.

Figura 3.3: O conceito de caixa preta.

- A **metodologia dos tipos abstratos de informação** 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. Consiste nos seguintes passos:
  1. Identificação das operações básicas;
  2. Axiomatização das operações básicas;
  3. Escolha de uma representação para os elementos do tipo;
  4. Realização das operações básicas para os elementos do tipo.

→ Existem diversas etapas para o **desenvolvimento de um programa**:

1. 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.
2. 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.
3. 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.
4. Testes: Definição de uma bateria de testes com o objetivo de “garantir" que o programa funciona corretamente em todas as situações possíveis.
5. 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.

→ Ao desenvolver programas, podemo-nos deparar com dois **tipos de 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 com y.
- 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 com y.