Conceito de Algoritmos e estrutura de dados

Definições de algoritmo:

- **Processo sistemático** para a resolução de um problema.
- Usado para descrever, de forma lógica, os passos a serem executados no cumprimento de determinada tarefa.
- Ferramenta para resolver um ***problema computacional*** bem especificado. A partir de um valor ou conjunto de valores como *entrada*, produz algum valor ou conjunto de valores como *saída*. Durante o processo de computação, o algoritmo manipula dados, gerados a partir de sua entrada.
- Descreve soluções de problemas do **nosso mundo (ilimitado)** utilizando os recursos do **mundo computacional (limitação de hardware/software)**.
- O conceito de um algoritmo foi formalizado em 1936 pela Máquina de Turing de Alan Turing e pelo cálculo lambda de Alonzo Church, que formaram as primeiras fundações da Ciência da computação.

Exemplos de problemas a serem resolvidos com algoritmos:

- Calcular média aritmética / ponderada.
- Ordenar uma sequência de números.
- Internet – gerenciar e manipular grandes quantidades de informações. Exemplos localização de boas rotas, uso de mecanismos de pesquisa para encontrar com rapidez páginas que residem informações específicas.
- Comércio eletrônico. Criptografia de chave pública e assinaturas digitais (números de cartão de crédito, senhas...)
- Mapa rodoviário. Como encontrar a rota mais curta.

Um programa de computador é essencialmente um algoritmo que informa ao computador os passos específicos, e em que ordem eles devem ser executados, como por exemplo, os passos a serem tomados para calcular as notas que serão impressas nos boletins dos alunos.

Quando os procedimentos de um algoritmo envolvem o processamento de dados, a informação é lida de uma fonte de entrada, processada e retornada sob novo valor após processamento, o que geralmente é realizado com o auxílio de uma ou mais estrutura de dados.

Para qualquer processo computacional, o algoritmo precisa estar rigorosamente definido, especificando a maneira que ele se comportará em todas as circunstâncias.

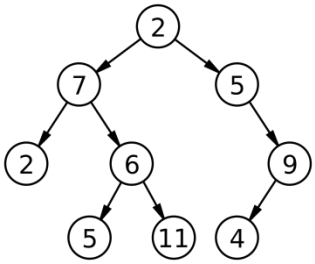
A corretividade do algoritmo pode ser provada matematicamente, bem como a quantidade assintótica de tempo e espaço (complexidade) necessários para a sua execução. Estes aspectos dos algoritmos são alvo da **análise de algoritmos**.

A análise de algoritmos é um ramo da ciência da computação que estuda as técnicas de projeto de algoritmos e os algoritmos de forma abstrata, sem estarem implementados em uma linguagem de programação em particular ou implementadas de algum outro modo.

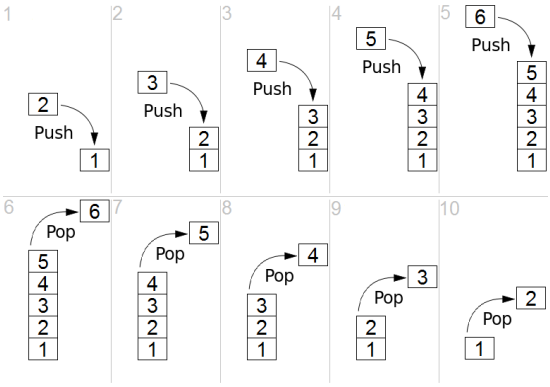
Preocupa-se com os **recursos necessários** para a execução do algoritmo, tais como o **tempo de execução e o espaço de armazenamento de dados.**

Para um dado algoritmo, pode-se ter diferentes quantidades de recursos alocados de acordo com os parâmetros passados na entrada. Por exemplo, se definirmos que o fatorial de um número natural é igual ao fatorial de seu antecessor multiplicado pelo próprio número, fica claro que a execução de fatorial(10) consome mais tempo que a execução de fatorial(5).

A Estrutura de dados (ED) é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente. Uma árvore binária e uma pilha são exemplos de ED:



Árvore Binária



Pilha

Exercício: Pesquisar sobre exemplos de algoritmos e de diferentes estruturas de dados.

Tipos abstratos de dados

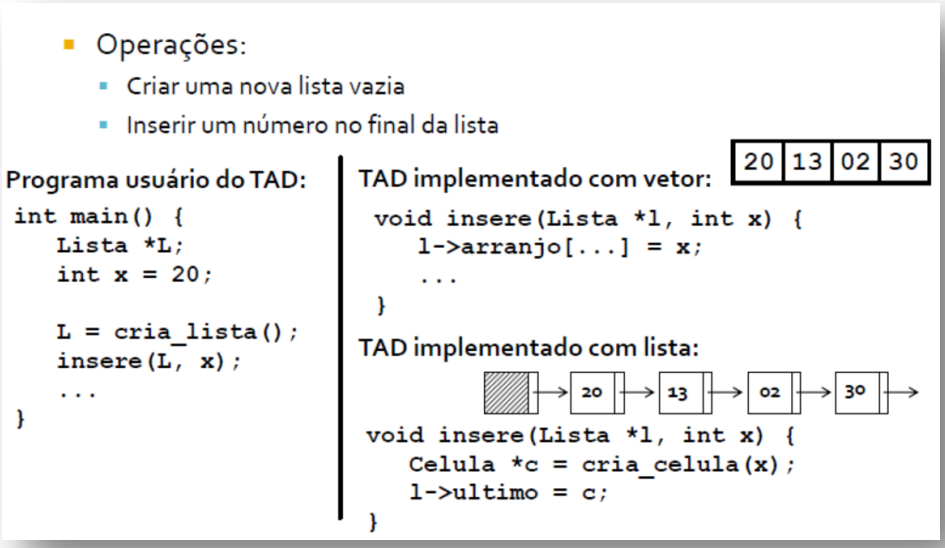
A **fase de delineamento** do programa contendo seus **requisitos** deve **preceder** o **processo de codificação**. Quanto maior e mais complexo o projeto, mais detalhada deveria ser esta fase. Os detalhes de implementação deveriam ser adiados para estágios posteriores do projeto, bem como os detalhes das estruturas de dados particulares a serem utilizadas na implementação não deveriam ser especificados no início.

Desde o início, é importante **especificar cada tarefa** em **termos de entrada e saída** e focar nossa atenção no **que o programa deveria fazer** (e não em como). Se um item é necessário para realizar algumas tarefas, ele é especificado em termos das operações nele realizadas, em vez de sua estrutura interna. A implementação do programa pode começar depois das operações terem sido especificadas precisamente. Define-se então qual estrutura de dados deveria ser usada para tornar a execução mais eficiente em relação a tempo e espaço. Estrutura de dados é uma implementação concreta de um tipo abstrato de dados (TAD).

Um item especificado em termos de operações é denominado *tipo abstrato de dados* (TAD). Uma linguagem orientada a objetos (LOO), como C++, tem vínculo direto com os tipos de dados, implementando-os como uma **classe**.

O TAD pode ser definido como um modelo matemático por meio de um par (v,o) em que: *v* é um conjunto de valores e *o* é um conjunto de operações sobre esses valores. Exemplo: Tipo *real*. v=$\mathcal{R}$ e o={+,-,*, /, =, <, >, <=, >=}.

Assim, os TAD encapsulam a representação dos dados e as operações que podem ser realizadas sobre eles. Os usuários de um TAD só tem acesso às operações disponibilizadas sobre os dados. Exemplo de TAD: lista de números:



Exercício: Pesquisar sobre exemplos de diferentes tipos de TAD;