Hoje somos inundados por diversos dispositivos digitais executando tarefas de forma automatizada. Embora seja muito intuitivo uma tarefa é uma sequência ordenada de passos finitos para resolver um determinado problema, sistematizar esse processo é denominado de algoritmo. Um algoritmo é uma ferramenta utilizada em várias áreas do conhecimento, na matemática alguns algoritmos são responsáveis por padronizar uma operação matemática qualquer e essa padronização pode ser facilmente repetida sempre que necessário. Na ciência da computação ele é fundamental para a construção de sistemas, de forma pragmática, um aplicativo é o algoritmo que foi programado em uma determinada linguagem de programação e foi transformado em linguagem de máquina podendo ser executado em computador. Vamos entender o processo de criação de um algoritmo, algumas padronizações, sua complexidade e eficiência, bem como entender como os tipos abstratos de dados vetores e matrizes funcionar.

Podemos construir algoritmos para uma infinidade de coisas, na verdade, a todo momento usamos e colocamos em prática algoritmos, seja de forma intuitiva quando queremos descobrir como resolver um dado problema, seja de forma explicita por meio de ferramentas especializadas que aplica um algoritmo já determinado. Vejamos, se você é um chefe de cozinha e cria receitas novas, você está construindo algoritmos, se você compra seu café da manhã por meio de um aplicativo que seleciona uma refeição seja a partir de uma dieta especifica ou não, você está usando algoritmos. Compreender o seu conceito é base para a construção de conhecimento mais solido, afinal desenvolver algoritmos estimula o raciocínio lógico e a cognição das pessoas. E quanto mais especializado um algoritmo é, mais eficiente ele se torna.

A Teoria da Computação busca determinar quais problemas podem ser aplicados em um modelo de computação, esse modelo formalmente pode ser definida como a solução de um problema por meio de um algoritmo, assim chegamos a conclusão que um algoritmo para ser computado deve obedecer a arquitetura de Von Newmann. Isso significa que a partir de entradas um algoritmo pode ser processado e produzir saídas resultantes. Em uma calculadora a função soma, é resultado de um algoritmo que ao receber valores retorna o valor resultante da sua operação. É importante salientar que não existe um algoritmo único, cada pessoa pode ver de forma diferente como resolver um dado problema. Alguns algoritmos são de fato muito simples enquanto outros muito difíceis, o nível de complexidade entre algoritmos pode ser classificado como, Fáceis (P), Razoáveis (NP), Difíceis (NP-Completo).

Algoritmos são classificados pelo tipo de problema que eles podem resolver e o tempo usado para tal, um fato importante é que o tempo de processamento de um algoritmo pode mudar de acordo com o tamanho da entrada que será utilizada, formalmente o tempo de um algoritmo é expresso pelo polinômio O(n^^3). Podemos separa-los como, Classe P, aqui estão vários problemas naturais, como, estruturas de decisão de programação linear, cálculo de Máximo divisor comum e são problemas que podem ser resolvidos em tempo polinomial, ao generalizarmos P, obtemos a Classe NP que é o conjunto de todos os problemas que podem ser resolvidos por algoritmos não-

deterministicos em tempo polinomial usando uma máquina de Turing não-deterministica - em uma máquina de estados um autômato determinístico demonstra que para cada estado existe apenas uma transição possível, já em um autômato não determinístico pode existir estados que podem levar a outros estados. Os problemas NP mais difíceis são os da classe NP-completo, estes por sua vez são problemas de decisão do tipo "sim ou não" e ainda existem problemas de otimização chamados de NP-Difícil, contudo sua dificuldade não é menor que a dificuldade de decisão dos problemas NP-Completo. Outra características de problemas NP-Completo é que, se qualquer um deles puder ser resolvido em tempo polinomial então todos os outros problema NP-completo também terão uma solução com tempo polinomial.

O estudo da complexidade dos algoritmos permite observar o quanto de empenho existe na busca de algoritmos mais eficientes. Para alcançar essa eficiência é importante obedecer algumas premissas, algoritmos devem ser simples e não podem ter ambiguidade, as ações devem ser ordenadas e estabelecer todas as sequencias de ações em passos finitos. Ele deve ser capaz de ler e escrever dados, avaliar expressões algébricas, relacionais e lógicas, ser capaz de tomar decisões com base em expressões avaliadas e repetir esse conjunto de ações de acordo com a necessidade. Relembrando, um software é um algoritmo que foi programado, à vista disso, o requisito básico é que a especificação deve fornecer uma descrição precisa do procedimento computacional a ser seguido.

Os algoritmos são independentes de linguagem contudo a técnica de representação por meio de pseudo-código utiliza um conjunto restrito de palavras-chaves como, 1) leia e escreva, responsáveis pelo processo de entrada e saída; 2) se, então, senão, senão-se, são estruturas de controle e decisão; 3) para, faça, enquanto é uma estrutura de loop ou laço; e 4) fim-se; geralmente essas palavras estão em uma linguagem nativa ao desenvolvedor. Esse tipo de representação também não possui a rigidez do formalismo das linguagens de programação propriamente dita, permitindo ser fácil de interpretar e de codificar. Contudo, o pseudo-código permite aplicar tanto conceitos lógicos como, verdadeiro e falso, quanto conceitos matemáticos como, conjuntos numéricos, algebra entre outros. Um conceito fundamental é o da variável, elas permitem realizar as mais diversas operações com os mais diversos tipos de dados diferentes.

As variáveis são espaços reservados na memória RAM do computador para guardar informações que serão utilizadas durante o código do programa. Pode-se armazenar nestas variáveis diversos valores de diversos tipos e tamanhos, tais como números inteiros, números reais, caracteres, frases, enfim, diversas coisas. A diversidade de tipos de dados que uma variável pode assumir é muito grande e não conhecer o tipo de dado que está armazenado no momento de uma determinada operação pode ocasionar erros, por isso, existem alguns tipos básicos em todas as linguagens de programação como: int (valores inteiros), float (valores decimais), e string (representa cadeias de caracteres, seja números ou letras). O tipo de um dado define o conjunto de valores ao qual o dado pertence, bem como o conjunto de todas as operações que podem atuar

sobre qualquer valor daquele conjunto de valores. Uma variável possui nome, tipo e conteúdo e pode ser representada em pseudo-código assim: var nome : string; var media : float; Por ver, quando é necessário mostrar o valor de uma variável em um pseudo-código, use o comando "imprima (conjunto de variáveis)", assim para construir a mensagem "Caro <aluno> sua média é <media>" comando é " imprima 'Caro' + nome + 'sua média é' + str(media)". Veja que é possível construir textos interpolando com algumas variáveis utilizando a operação de concatenação, que usa o mesmo conceito e sinal simbólico da operação aritmética soma. Contudo essa operação só é permitida para o tipo String, qualquer variável de outro tipo, deverá ser convertida para string atravéz do comado str(variavel) que converte outros tipos para script. Em muitos casos precisamos armazenar um numero considerável de dados uma variável, como uma lista de pessoas que participam de um concurso. Para esse caso, podemos usar uma estrutura de dado denominado de Vetor ou Matriz.

Nos algoritmos foi introduzido recursos e técnicas de programação que são comuns a todas as linguagens de programação. Umas delas pode ser conhecida por diferentes nomes e isso depende da linguagem, em Java chama-se de métodos, em C/C++ de funções e em Pascal é chamada de Função ou Procedimento. Todas estas linguagens aplicam de forma correta o termo, tanto procedimentos quantos funções basicamente fazem a mesma coisa, alguns autores diferem da seguinte forma, procedimento é uma função (void) que não retorna nada, já uma função retornar algum valor. Tanto uma quanto a outra são blocos de códigos nomeados e que podem ser reutilizados sempre que necessário a ideia é dividir o programa em subprogramas, é útil para deixar mais fácil de depurá-lo e de se reutilizar código.

Quando um procedimento ou função é chamado durante a execução de um programa, a execução é desviada para o subprograma, após seu término, a execução continua a partir do ponto onde foi chamado. A simples operação soma em uma calculadora qualquer, possui uma estrutura simples para somar o mínimo de dois números (2+4 ou 5+3 ou 77+123), isso me garante que a operação soma, recebe parâmetros que serão enviados para a função. A função para somar dois números pode ser escrita assim, "func soma(valor1, valor2) { retorna valor1 + valor2 }, para executar essa no programa bastar chama-la e passar os parâmetros ou melhor, valores que serão somados, soma(2,7), assim a função retorna o valor 9. O parâmetro é um valor que é fornecido à função quando ela é chamada para que a função execute um determinada operação. A lista de parâmetros é delimitada pelos parênteses no cabeçalho do procedimento, sua declaração é feita semelhante à declaração de variáveis.

Outra característica destas estruturas é o seu escopo, acarretando na existência de dois tipos de variáveis, Global e Local. As variáveis Globais fazem parte de toda a base do código podendo ser executada em qualquer extensão do código, já as variáveis Locais elas só tem funcionamento exatamente dentro do escopo das funções ou procedimentos que os cria. Existem diversos tipos de funções, para resolver os mais diversos tipos de problemas. Alguns tipos de funções são

considerados especiais, como funções recursivas, funções anônimas. Uma função recursiva ocorre quando dentro de um procedimento (ou função) existe uma chamada (direta ou indireta) para o mesmo procedimento (ou função). A grande vantagem da recursão está na possibilidade de usar um programa de computadorfinito para definir, analisar ou produzir um estoque potencialmente infinito de sentenças. Algumas linguagens de programação implementam as funções anônimas, que seria uma função que não tem um identificador, devem ser usadas conforme a experiência, e o objetivo do programador, tendo em conta que, normalmente, estas soluções são consideradas mais elegantes e fáceis de ler.

Podemos concluir que um algoritmo é uma ferramenta metodológica tanto para aprendizado da lógica de programação quanto é utilizada por desenvolvedores experientes enxergar de forma clara a solução de determinados problemas. O estudo de suas estruturas permite um amplo entendimento do funcionamento, bem como compreender a aplicação de Procedimentos e Funções para resolver problemas de forma mais organizada e elegante, não esquecendo do fato que estas estruturas permitem a reutilização de códigos dos blocos de funções durante toda a extensão do programa.