Desde que Allan Turing em 1936 formalizou uma maquina de estados capaz de resolver problemas computáveis, ou melhor, um algoritmo que processa entradas e gera uma saída como resultado, que a computação não parou mais de evoluir. A sua máquina funcionava como um algoritmo genérico e que possuía como memória uma fita separada que armazenava as entradas e instruções, e sempre que o problema mudava era necessário mudar a fita para o novo problema em particular. Nove anos Anos depois, o matemático Jhon von Newmann propôs algumas alterações a máquina de Turing para torná-la mais prática. A proposta incluia adicionar um dispositivo de entrada, integrar uma memória fixa para guardar dados e instruções, adicionar uma unidade de processamento, um sistema operacional e um dispositivo de saída a qual a solução era apresentada. Este modelo posteriormente ficou conhecido como arquitetura de Von Newmann e desde então, tornou-se um padrão e permitiu aos computadores evoluirem em um curto intervalo de tempo. Se antes eram máquinas que operavam apenas uma tarefa, agora são computadores capazes de trabalhar com um elevado volumes de dados. Contudo, para que esse trabalho seja realizado com eficiência é necessário que os dados sejam organizados na memória de modo a permitir aos programas acessa-los rapidamente, essa organização é chamada de Estrutura de Dados.

Para a computação, todo dispositivo que permite ao computador guardar dados de forma temporária ou permanente é denominado de Memória, elas podem ser divididas em Memória, principal e secundária. A principal (RAM, Randomic) é aquela onde os dados são armazenados para serem processados pela unidade central de processamento (CPU), para tal possui acesso de alta velocidade, embora armazena volumes pequenos e sempre que o computador é desligado esta é perdida. Já a memória secundária possui capacidade de guardar grandes volumes de dados de forma permanente porém o acesso aos seus dados é muito lento comparado com a memória principal.

O caráter Randômico da memória RAM é responsável pela velocidade desta memória, e isso faz com que um dado seja simplesmente jogado em um qualquer bloco de memória. Como a memória principal é feita por semicondutores de circuitos digitais, a uníca informação que se tem do dado é um valor em hexadecima que representa a posição do dado na memória, imagine a memória como uma planilha eletrônica onde cada célula é uma posição de memória e pode ser identificada e localizada rapidamente. Algumas linguagens de programação permitem o acesso direto a memória por meio de uma estrutura denominada de ponteiro. Na analogia a uma planilha eletrônica, o ponteiro é capaz de acessar uma ou várias células especificas e explorar seu conteúdo.

Para que as operações sejam executadas corretamente é desejável que linguagens de programação estabeleçam uma estrutura para tratar os diferentes tipos de dados, alguns desses tipos são considerados primitivos como, char, int, float, double e outros são

considerados abstratos, como, listas, filas, pilhas. Os tipos abstratos de dados (TAD) são estruturas complexas capazes de armazenar variáveis ou conjunto de variáveis e usam estruturas de dados para as representarem. De forma geral uma estrutura de dados é uma forma de se implementar um tipo abstrato de dados. Os TADs permitem inclusões e remoções de dados e operações particulares como percorrer entre seus elementos, realizar buscas e ordenação. Um ponteiro é uma estrutura de dados e comportamento de um array (vetor) definido direto na memória principal. Deve-se ter cuidado com ponteiros, uma vez definido o tamanho do ponteiro (tamanho do vetor) se o seu tamanho for violado pode causar um erro chamado de "buffer overflow", geralmente causando um congelamento da aplicação.

Estrutura de dados é o ramo da ciência da computação que estuda os diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento. Manipular dados na memórias dentro de um processo de computação, possui um custo muito elevado, porem, vários processos possuem padrões que se repetem, permitindo elaborar algoritmos previamente testados e definidos para aqueles problemas. Uma estrutura muito comum é a lista. É formada por um conjunto de dados e preserva a relação de ordem linear entre eles, pode representar ordem de precedência entre os elementos ou não. Seu algoritmo de formação, ou melhor, sua estrutura de dados permite inserir elementos seja no fim da lista, seja em uma posição especifica, selecionar ou remover um elemento qualquer, verificar se um elemento existe e informa seu tamanho. Algumas linguagens de programação fornecem tipos abstratos de dados lineares, como é o caso do Array e Listas, que possuem uma estrutura de dados similar, contudo algumas outros TADs são implementados a partir dessa estruturas já existentes e diferem apenas na estrutura de dados que caracteriza sua regra de formação e uso. As listas podem ainda ter características diversas, como, lista encadeada, lista duplamente encadeadas, lista circular. Na lista encadeada cada elemento da lista possui um identificador que aponta para o próximo elemento da lista, já na duplamente encadeada a mudança é sutil, cada nó alem de identificar o seu próximo, também possui o endereço identificador do nó anterior. A lista circular é semelhante as demais, contudo o ultimo elemento da lista aponta para o primeiro da mesma lista, dando um caráter cíclico a estrutura.

Basicamente toda estrutura de dados linear é implementada a partir de um array, de forma que cada lista difere em seu algoritmo, isso pode ser mais notório quando olhamos para as estruturas de Fila e Pilha. Estas estruturas são comumente utilizadas em diversos tipos de problemas, desde controle de filas de atendimentos até empilhamento de containers em portos marítimos.

Em uma fila de pessoas para atendimento em um banco, independente de ser uma fila preferencial ou não, ela obedece as mesmas regras, que é, a primeira pessoa a chegar na fila será a primeira pessoa a sair, caso contrário houve um fura fila. O algoritmo da estrutura de

Fila é definido como FIFO (First in, First Out) o primeiro que entra é o primeiro que sai. A implementação do algoritmo da fila permite realizar as operações que satisfazem a estrutura FIFO, como, Add (adiciona elemento no início da fila), Pop (remove o elemento no fim da fila), Len (retorna o tamanho da fila). Tanto o primeiro quanto o ultimo elemento da fila recebe uma identificação que os classifica, de forma que quando a fila tiver apenas um elemento, ele terá tanto a marca de fim como de início, estas marcações são facilmente implementadas com ponteiros. Normalmente em uma fila sempre que os dados são removidos, eles são perdidos. Uma solução é implementar Filas Circulares, pois não ocorre a perca dos dados, apenas são alterados os ponteiros indicadores de inicio e fim da fila. Outras aplicações das filas são, enfileiramento de documentos para impressão, fila de processos no processador, troca de mensagens entre computadores numa rede.

Agora vejamos um cenário para a estrutura em Pilha, em um porto cargueiro containers estão empilhados um em cima do outro de forma que para remover o containers que está na base da pilha é necessário remover todos os que estão por cima, assim, o algoritmo de formação de uma pilha é o LIFO (Last in, First out) o ultimo que entra é o primeiro que sai. De forma prática, imagine um navio carregado com containers e o roteiro de viagem, Brasil, Angola, e com destino final Portugal. Para o sucesso da viagem, os containers devem ser empilhados de forma que os últimos destino são os primeiros a serem acomodados. Da mesma forma que uma fila, as pilhas também são estruturas lineares e podem ser implementadas a partir de uma lista ou array e podem compartilhar comportamentos. Pode-se até mesmo dizer que, tais estruturas são vetores com algoritmos específicos. Numa pilha, a manipulação dos elementos é realizada em apenas uma das extremidades, chamada de topo, em oposição a outra extremidade, chamada de base. As operações em uma pilha são, add (empilhar, adicionando elemento ao topo da pilha), pop (remove elemento do topo da pilha), Len (verifica se a pilha está vazia ou cheia) e permite implementar métodos que mostra o elemento do topo. Da mesma forma da fila, sempre que um elemento é adicionado na pilha ele recebe uma informação que identifica se o elemento é o Topo ou a Base da pilha, se tiver apenas um elemento na pilha, este recebe a identificação tanto do topo como a base. Outra aplicação da estrutura de Pilha são, funções recursivas em compiladores, mecanismo de desfazer/refazer dos editores de texto, Navegação entre páginas Web.

Compreender diversas estruturas de dados diferentes e um aprofundamento no conhecimento das estruturas de memória disponível, permite refletir sobre como diminuir o custo computacional ao organizar os dados a serem armazenados. A estrutura de dados é uma área da ciência da computação que pesquisa as mais diversas estruturas realizando testes e otimizações das estruturas já existentes e a cada dia novas estruturas de dados tem seus axiomas definidos. Processos onerosas como ordenação e busca de elementos podem ser drasticamente melhorados quando se utiliza uma estrutura de dados adequada, embora

algumas estruturas sejam bem definidas para determinados problemas, em muitos casos elas podem ser combinadas, por exemplo, em uma fila de banco podemos ter um conjunto de regras de prioridade, como, Ordem de chagada, Idosos, Gestantes, Lactantes, Deficientes cada uma obedecendo sua regra de formação. Dessa forma podemos concluir que a escolha sobre a estrutura pode ser a própria natureza do problema, e por isso deve ser analiticamente observada e testada pelo desenvolvedor ou equipe de desenvolvimento.