1.Estruturas auto-referenciadas

Uma estrutura auto-referenciada contém um membro ponteiro que aponta para uma estrutura do mesmo tipo

```
struct node {

int data;

struct node *nextPtr;
```

A estrutura acima define um tipo, ***struct node***. Uma estrutura do tipo ***struct node*** tem dois membros: um membro inteiro data e um membro ponteiro ***nextPtr***. O membro ***nextPtr*** aponta para uma estrutura do tipo ***struct node***. O membro ***nextPtr*** é chamado de link e pode ser usado para "ligar" uma estrutura do tipo ***struct node*** com outras estruturas do mesmo tipo. As estruturas auto-referenciadas podem ser ligadas entre si para formar estruturas úteis de dados como listas, filas, pilhas e árvores.

2.Listas encadeadas

Uma lista encadeada é um conjunto linear de estruturas auto-referenciadas, chamadas nós, conectadas por links de ponteiros. Uma lista encadeada é acessada por meio de um ponteiro para o primeiro nó da lista. Os nós subsequentes são acessados por meio do membro ponteiro de ligação armazenado em cada nó. Os dados são armazenados dinamicamente em uma lista encadeada. Um nó pode conter dados de qualquer tipo, incluindo outras structs. Pilhas e filas também podem ser estruturas lineares. As árvores são estruturas não-lineares de dados.

As listas de dados podem ser armazenadas em ***arrays***, mas as listas encadeadas apresentam várias vantagens. Uma lista encadeada é recomendável quando o número de elementos de dados a serem representados na estrutura de dados não pode ser previsto imediatamente.

As listas encadeadas são dinâmicas, portanto, o comprimento de uma lista pode aumentar ou diminuir quando necessário. Entretanto, o tamanho de um ***array*** não pode ser alterado porque a memória do ***array*** é alocada em tempo de compilação. Os ***arrays*** podem ficar cheios. As listas encadeadas ficarão cheias quando o sistema não tiver memória suficiente para satisfazer as exigências de alocação dinâmica do armazenamento.

Normalmente os nós de listas encadeadas não são armazenados contiguamente na memória. Entretanto, logicamente os nós de uma lista encadeada parecem estar contíguos.

As duas funções principais das listas encadeadas são ***insert*** e ***delete***. Uma função ***isEmpty*** é chamada função predicada, ela não altera a lista de nenhuma

maneira, mas sim, determina se a lista está vazia. Se sim, 1 é retornado, caso contrário, 0 é retornado.

3.Pilhas/Stacks

Uma pilha (stack) é uma versão limitada de uma lista encadeada. Os novos nós só podem ser adicionados no topo da pilha e também só podem ser removidos os nós do topo de uma pilha. Por esse motivo, a pilha é conhecida como uma estrutura de dados último a entrar, primeiro a sair (LIFO).

A referência a uma pilha é feita por meio de um ponteiro para o elemento do topo da pilha. O membro de ligação no último nó da pilha é definido com NULL para indicar a base da pilha.

As principais funções utilizadas para manipular uma pilha são ***push*** e ***pop***. A função ***push*** cria um novo nó e o coloca no topo da pilha. A função ***pop*** remove um nó do tipo de uma pilha, libera a memória que estava alocada ao nó removido e retorna o valor removido.

As pilhas possuem muitas aplicações interessantes. Por exemplo, sempre que for feita a chamada de uma função, a função chamada deve saber como retornar à função chamadora, portanto, o endereço de retorno é colocado em uma pilha. Se ocorrer a chamada de uma série de funções, os sucessivos valores de retorno são colocados na pilha, na ordem último a entrar, primeiro a sair, de forma que cada função possa retornar a quem a chamou. As pilhas suportam chamadas recursivas de funções da mesma forma que chamadas convencionais não-recursivas.

As pilhas contêm o espaço criado para variáveis automáticas em cada chamada de uma função. Quando a função retorna a quem a chamou, o espaço daquelas variáveis automáticas é removido da pilha e as variáveis deixam de ser conhecidas pelo programa.

4.Filas

Outra estrutura comum de dados é a fila (Queue). Uma fila funciona como um atendimento em uma mercearia, a primeira pessoa da fila é atendida em primeiro lugar e os outros clientes entram na fila somente no final e esperam ser servidos. Os nós são removidos apenas do início das filas e são inseridos apenas em seu final. Por esse motivo, uma fila é conhecida como uma estrutura de dados do tipo primeiro a entrar, primeiro a sair (FIFO). As operações de inserção e remoção são conhecidas como enfileirar e desenfileirar.

As filas possuem muitas aplicações em sistemas computacionais. Muitos computadores possuem apenas um único processador, portanto apenas um usuário pode ser servido de cada vez. As entradas de outros usuários são colocadas em uma

fila. Cada entrada avança gradualmente para a frente da fila à medida que outros usuários são servidos. A entrada da frente da fila é a próxima a ser servida.

As filas também são usadas para suportar armazenamento de dados (spooling) para impressão. Um ambiente multiusuário pode ter apenas uma única impressora. Muitos usuários podem estar gerando saídas impressas. Se a impressora estiver ocupada, outras saídas ainda podem estar sendo geradas. Estas são armazenadas (spooled) em disco onde esperam em uma fila até que a impressora fique disponível.

Os pacotes de informações também esperam filas em redes de computadores. Cada vez que um pacote chega a um nó da rede, tal pacote deve ser direcionado para o próximo nó da rede situado ao longo do trajeto do pacote até seu destino final. O nó de roteamento dirige um pacote por vez, portanto pacotes adicionais são enfileirados até que o roteador possa enviá-los.

5.Árvores

As listas encadeadas, pilhas e filas são estruturas lineares de dados. Uma árvore é uma estrutura de dados não-linear e bidimensional com propriedades especiais. Os nós da árvore contêm dois ou mais links. Esta seção analisa as árvores binárias, árvores cujos nós contêm dois links (nenhum, ou ambos dos quais podem ser NULL).

O nó raiz é o primeiro nó da árvore. Cada link do nó raiz se refere a um filho. O filho da esquerda é o primeiro nó na subárvore esquerda e o filho da direita é o primeiro nó na subárvore direita. Os filhos de um nó são chamados irmãos. Um nó sem filhos é chamado um nó folha. Normalmente, os cientistas computacionais desenham árvores do nó raiz para baixo.

Uma árvore de pesquisa binária apresenta a característica de que os valores em qualquer subárvore esquerda são menores do que o valor de seu nó pai e os valores de qualquer subárvore direita são maiores do que o valor em seu nó pai.

A árvore de pesquisa binária facilita a eliminação de valores duplicados. À medida que a árvore é criada, uma tentativa de inserir um valor duplicado será reconhecida porque a duplicata enfrentará as mesmas decisões que o valor original de "ir para a esquerda" ou "ir para a direita" em cada comparação. Dessa forma, posteriormente a duplicata será comparada com um nó que contém o mesmo valor. Nesse momento a duplicata pode ser simplesmente descartada.

Procurar em uma árvore binária um valor que corresponde a um valor-chave também é rápido. Se a árvore estiver montada corretamente, cada nível contém cerca de duas vezes o número de elementos do nível anterior. Portanto, uma busca binária com "n" elementos teria um máximo de log_2 níveis e assim precisariam ser feitas no máximo $log_2 n$ comparações para encontrar uma correspondência ou para determinar que ela não existe. Isso significa, por exemplo, que comparar uma árvore binária de

1000 elementos não exigiria mais de 10 comparações porque $2^{10} > 1000$. Para pesquisar uma árvore de pesquisa binária com 1.000.000 elementos não seriam necessários mais de 10 comparações porque $2^{20} > 1.000.000$