simples e o mais barato de manter. A desvantagem é que ele ignora o fato de que a página mais antiga pode ser a mais referenciada.

Toda informação necessária ao algoritmo escolhido para remoção de páginas pode ser armazenada em cada Moldura de Página. Para registrar o fato de que uma página sofreu alteração no seu conteúdo (para sabermos se ela terá de ser reescrita na memória secundária), basta manter um *bit* na Moldura de Página correspondente.

Resumindo, em um sistema de memória virtual, o programador pode endereçar grandes quantidades de dados, deixando para o sistema a responsabilidade de transferir o dado endereçado da memória secundária para a memória principal. Essa estratégia funciona muito bem para os algoritmos que possuem uma **localidade de referência espacial** pequena, isto é, cada referência a uma localidade de memória tem grande chance de ocorrer em uma área que é relativamente próxima de outras áreas que foram recentemente referenciadas. Isso faz com que o número de transferências de páginas entre a memória principal e a memória secundária diminua muito. Por exemplo, a maioria das referências a dados no Quicksort ocorre perto de um dos dois apontadores que realizam a partição do conjunto, o que pode fazer com que esse algoritmo de ordenação interna funcione muito bem em um ambiente de memória virtual para uma ordenação externa.

6.1.2 Implementação de um Sistema de Paginação

A seguir, vamos descrever uma das formas possíveis de implementar um sistema de paginação. A estrutura de dados é mostrada no Programa 6.1. O Programa apresenta também a estrutura de dados para representar uma árvore binária de pesquisa, em que um apontador para um nó da árvore é representado pelo par: número da página (p) e posição dentro da página (b). Assumindo que a chave é constituída por um inteiro de 2 *bytes* e o endereço ocupa 2 *bytes* para p e 1 *byte* para b, o total ocupado por nó da árvore é de 8 *bytes*. Como o tamanho da página é de 512 *bytes*, então o número de itens (nós) por página é 64.

Em alguns casos pode ser necessário manipular mais de um arquivo ao mesmo tempo. Quando isso ocorre, uma página pode ser definida como no Programa 6.2, em que o usuário pode declarar até três tipos diferentes de páginas. Se o tipo TipoPaginaA for declarado

```
type TipoPaginaA = array[1..ItensPorPagina] of ItemTipo;
```

e a variável Pagina for declarada

```
var Pagina: TipoPagina;
```

então é possível a seguinte atribuição:

```
Pagina.Pa[1].Reg.Chave := 10;
```


Programa 6.1 *Estrutura de dados para o sistema de paginação*

```
const TAMANHODAPAGINA = 512;
      ITENSPORPAGINA = 64; { TamanhodaPagina/TamanhodoItem }
type Registro = record
                  Chave: TipoChave;
                  { outros componentes }
                end;
     TipoEndereco = record
                      p: integer;
                      b: 1..ITENSPORPAGINA;
                    end;
     TipoItem = record
                  Reg: TipoRegistro;
                  Esq, Dir: TipoEndereco;
                end;
     TipoPagina = array [1..ITENSPORPAGINA] of TipoItem;
```

Programa 6.2 *Diferentes tipos de páginas para o sistema de paginação*

```
type TipoPagina = record
                    case byte of
                    0 : (Pa : TipoPaginaA);
                    1 : (Pb : TipoPaginaB);
                    2 : (Pc : TipoPaginaC);
                  end;
```

A Tabela de Páginas para cada arquivo poderá ser declarada separadamente, mas a Fila de Molduras é única, bastando para isso ter em cada moldura a indicação do arquivo a que se refere aquela página.

A comunicação com o sistema de paginação poderá ser realizada com o uso dos seguintes procedimentos:

1. ObtemRegistro: Torna disponível um registro de um arquivo. O parâmetro de entrada é o endereço virtual $< p, b >$ e o parâmetro de saída é o apontador para a Moldura de Página ($< p\prime, b >$ na Figura 6.1).
2. EscreveRegistro: Permite criar ou alterar o conteúdo de um registro. O procedimento possui dois parâmetros de entrada: o registro e seu endereço virtual $< p, b >$.
3. DescarregaPaginas: Permite varrer a Fila de Molduras para atualizar na memória secundária todas as páginas que porventura tenham sofrido qualquer alteração no seu conteúdo (bit de alteração = **true**).

O diagrama da Figura 6.3 mostra a transformação do endereço virtual para o endereço real de memória do sistema de paginação, tornando disponível na memó-