Programa 5.29 Operações do dicionário usando endereçamento aberto

```
procedure Inicializa (var T: TipoDicionario);
var i: integer;
begin
  for i := 0 to M - 1 do T[i].Chave := VAZIO
end; { Inicializa }

function Pesquisa (Ch: TipoChave; var p: TipoPesos;
                   var T: TipoDicionario): TipoApontador;
var i, Inicial: integer;
begin
  Inicial := h (Ch, p);
  i := 0;
  while (T[(Inicial + i) mod M].Chave <> VAZIO) and
        (T[(Inicial + i) mod M].Chave <> Ch) and
        (i < M) do i := i + 1;
  if T[(Inicial + i) mod M].Chave = Ch
  then Pesquisa := (Inicial + i) mod M
  else Pesquisa := M { Pesquisa sem sucesso }
end; { Pesquisa }

procedure Insere (x: TipoItem; var p: TipoPesos;
                  var T: TipoDicionario);
var i, Inicial: integer;
begin
  if Pesquisa (x.Chave, p, T) < M
  then writeln ('Elemento ja esta presente')
  else begin
       Inicial := h (x.Chave, p);
       i := 0;
       while ((T[(Inicial + i) mod M].Chave <> VAZIO) and
          (T[(Inicial + i) mod M].Chave <> RETIRADO)) and
          (i < M) do i := i + 1;
       if i < M
       then T[(Inicial + i) mod M] := x
       else writeln (' Tabela cheia')
       end;
end; { Insere }

procedure Retira (Ch: TipoChave; var p: TipoPesos;
                  var T: TipoDicionario);
var i: integer;
begin
  i := Pesquisa (Ch, p, T);
  if i < M
  then T[i].Chave := RETIRADO
  else writeln ('Registro nao esta presente')
end; { Retira }
```


Análise Considere $\alpha = N/M$ o fator de carga da tabela. Conforme demonstrado por Knuth (1973), o custo de uma pesquisa com sucesso é:

$$C(N) = \frac{1}{2}\left(1 + \frac{1}{1-\alpha}\right).$$

O ***hashing* linear** sofre de um mal chamado **agrupamento** (**clustering**; Knuth, 1973, p. 520–521). Esse fenômeno ocorre quando a tabela começa a ficar cheia, pois a inserção de uma nova chave tende a ocupar uma posição contígua a outras posições já ocupadas, o que deteriora o tempo necessário para novas pesquisas. Entretanto, apesar de o *hashing* linear ser um método relativamente pobre para resolver colisões, os resultados apresentados são bons. A tabela 5.2 mostra alguns valores para $C(N)$ para diferentes valores de α.

Tabela 5.2 Número de comparações em uma pesquisa com sucesso para hashing *linear*

α	$C(N)$
0,10	1,06
0,25	1,17
0,50	1,50
0,75	2,50
0,90	5,50
0,95	10,50

O aspecto negativo do método, seja listas encadeadas ou endereçamento aberto, está relacionado com o pior caso, que é $O(N)$. Se a função de transformação não conseguir espalhar os registros de forma razoável pelas entradas da tabela, então uma longa lista linear pode ser formada, deteriorando o tempo médio de pesquisa. O melhor caso, assim como o caso médio, é $O(1)$.

Como vantagens na utilização do método de transformação da chave citamos: (i) alta eficiência no custo de pesquisa, que é $O(1)$ para o caso médio; e (ii) simplicidade de implementação. Como aspectos negativos citamos: (i) o custo para recuperar os registros na ordem lexicográfica das chaves é alto, sendo necessário ordenar o arquivo; e (ii) o pior caso é $O(N)$.

5.5.4 Hashing *Perfeito com Ordem Preservada*

Uma função de transformação transforma um conjunto de chaves x_j, $1 \leq j \leq N$, em um conjunto de valores inteiros no intervalo $0 \leq h(x_j) \leq M - 1$ com colisões permitidas. Nos casos em que $h(x_i) = h(x_j)$ se e somente se $i = j$, então não há colisões, e a função de transformação é chamada **função de transformação perfeita** ou função *hashing* perfeita, denominada por *hp*.