

Programa 5.25 Implementação da função de transformação  $h$  de Zobrist

```

function h (Chave: TipoChave; p: TipoPesos): TipoIndice;
var i, Soma: integer; { Funcao h do Zobrist }
begin
  Soma := 0;
  for i := 1 to N do Soma := Soma + p[i, ord(Chave[i])];
  h := Soma mod M;
end; { h }

```

### 5.5.2 Listas Encadeadas

Uma das formas de resolver as **colisões** é simplesmente construir uma lista linear encadeada para cada endereço da tabela. Assim, todas as chaves com mesmo endereço são encadeadas em uma lista linear.

Se a  $i$ -ésima letra do alfabeto é representada pelo número  $i$  e a função de transformação  $h(\text{Chave}) = \text{Chave} \bmod M$  é utilizada para  $M = 7$ , então a Figura 5.14 mostra o resultado da inserção das chaves *P E S Q U I S A* na tabela. Por exemplo,  $h(A) = h(1) = 1$ ,  $h(E) = h(5) = 5$ ,  $h(S) = h(19) = 5$  etc.

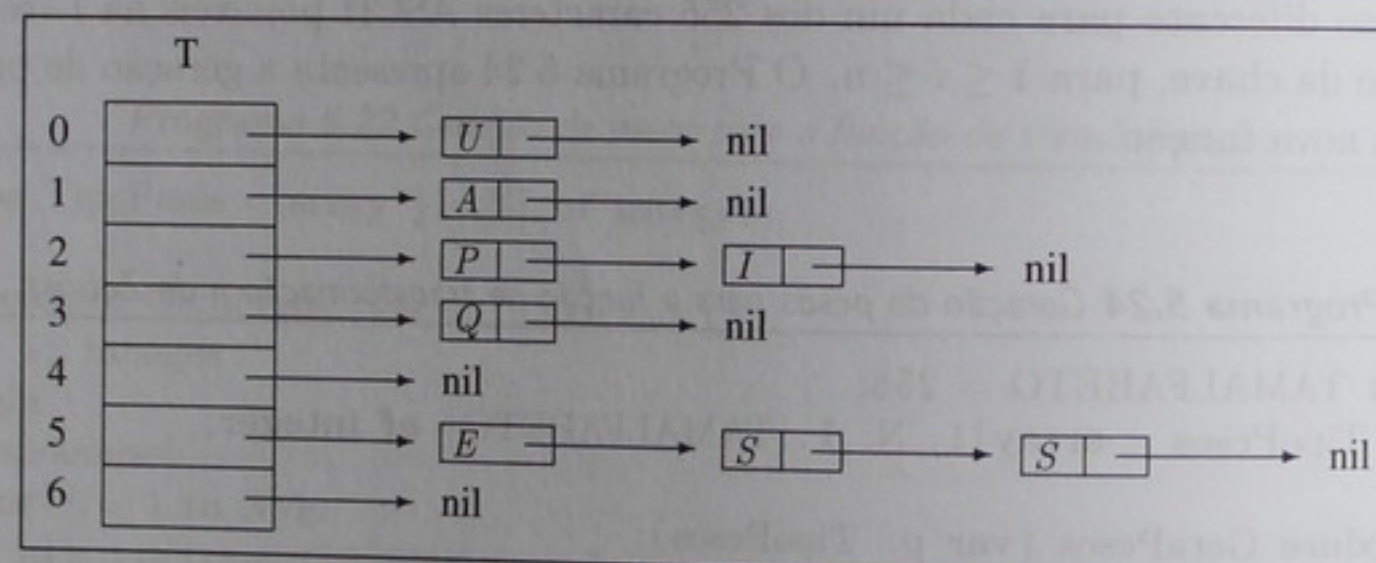


Figura 5.14 Lista encadeada em separado.

A estrutura de dados lista encadeada em separado será utilizada para implementar o tipo abstrato de dados Dicionário, com as operações Inicializa, Pesquisa, Insere, Retira. A estrutura do dicionário é apresentada no Programa 5.26.

A implementação das operações sobre o Dicionário são mostradas no Programa 5.27. As operações FLVazia, Insere e Retira, definidas sobre o TipoLista, mostradas no Programa 3.4 do Capítulo 3, podem ser utilizadas para manipular as listas encadeadas. Entretanto, será necessário alterar os nomes dos procedimentos Insere e Retira do Programa 3.4 para Ins e Ret respectivamente, para não haver conflito com os nomes dos procedimentos Insere e Retira do Dicionário (vide procedimentos Insere e Retira no Programa 5.27).

Programa 5.26 Estrutura do dicionário para listas encadeadas

```

type TipoChave = packed array [1..N] of char;
    TipoItem = record
        Chave: TipoChave;
        { outros componentes }
    end;
    TipoIndice = 0..M-1;
    TipoApontador = ^TipoCelula;
    TipoCelula = record
        Item: TipoItem;
        Prox: TipoApontador
    end;
    TipoLista = record
        Primeiro: TipoApontador;
        Ultimo: TipoApontador
    end;
    TipoDicionario = array [TipoIndice] of TipoLista;
{— Entra aqui TipoPesos do Programa 5.22 ou do Programa 5.24 —}

```

Programa 5.27 Operações do Dicionário usando listas encadeadas

```

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

function Pesquisa (Ch: TipoChave; var p: TipoPesos;
                  var T: TipoDicionario): TipoApontador;
{— Obs.: Apontador de retorno aponta para o item anterior da lista —}
var i: TipoIndice;
    Ap: TipoApontador;
begin
  i := h (Ch, p);
  if Vazia (T[i])
  then Pesquisa := nil { Pesquisa sem sucesso }
  else begin
    Ap := T[i].Primeiro;
    while (Ap^.Prox^.Prox <> nil) and (Ch <> Ap^.Prox^.Item.Chave) do
      Ap := Ap^.Prox;
    if Ch = Ap^.Prox^.Item.Chave
    then Pesquisa := Ap
    else Pesquisa := nil { Pesquisa sem sucesso }
    end
  end;
end; { Pesquisa }

```