Continuação do Programa E.27

```
TipoApontador Pesquisa(TipoChave Ch, TipoPesos p, TipoDicionario T)
{ /* TipoApontador de retorno aponta para o item anterior da lista */
  TipoIndice i;
  TipoApontador Ap;
  i = h(Ch, p);
  if (Vazia(T[i])) return NULL;  /* Pesquisa sem sucesso */
  else
  { Ap = T[i].Primeiro;
    while (Ap->Prox->Prox != NULL &&
           strncmp(Ch, Ap->Prox->Item.Chave, sizeof(TipoChave)))
      Ap = Ap->Prox;
    if (!strncmp(Ch, Ap->Prox->Item.Chave, sizeof(TipoChave)))
    return Ap;
    else return NULL;  /* Pesquisa sem sucesso */
  }
}
```

Programa E.28 *Estrutura do dicionário usando endereçamento aberto*

```
#define VAZIO   "!!!!!!!!!!"
#define RETIRADO   "**********"
#define M  7
#define N  11   /* Tamanho da chave */

typedef unsigned int TipoApontador;
typedef char TipoChave[N];
typedef unsigned TipoPesos[N];
typedef struct TipoItem {
  /* outros componentes */
  TipoChave Chave;
} TipoItem;
typedef unsigned int TipoIndice;
typedef TipoItem TipoDicionario[M];
```

Programa E.29 *Operações do dicionário usando endereçamento aberto*

```
void Inicializa(TipoDicionario T)
{ int i;
 for (i = 0; i <M; i++) memcpy(T[i].Chave, VAZIO, N);
}

TipoApontador Pesquisa(TipoChave Ch, TipoPesos p, TipoDicionario T)
{ unsigned int  i = 0; unsigned int  Inicial;
  Inicial = h(Ch, p);
  while (strcmp(T[(Inicial + i) % M].Chave,VAZIO) != 0 &&
         strcmp (T[(Inicial + i) % M].Chave, Ch) != 0 && i <M)
    i++;
```


Continuação do Programa E.29

```
  if (strcmp( T[(Inicial + i) % M].Chave, Ch) == 0)
  return ((Inicial + i) % M);
  else return M;  /* Pesquisa sem sucesso */
}

void Insere(TipoItem x, TipoPesos p, TipoDicionario T)
{ unsigned int i = 0; unsigned int Inicial;
  if (Pesquisa(x.Chave, p, T) <M)
  { printf("Elemento ja esta presente\n"); return; }
  Inicial = h(x.Chave, p);
  while (strcmp(T[(Inicial + i) % M].Chave,VAZIO) != 0 &&
         strcmp(T[(Inicial + i) % M].Chave, RETIRADO) != 0 && i <M)
    i++;
  if (i <M)
  { strcpy(T[(Inicial + i) % M].Chave, x.Chave);
    /* Copiar os demais campos de x, se existirem */
  }
  else printf(" Tabela cheia\n");
}

void Retira(TipoChave Ch, TipoPesos p, TipoDicionario T)
{ TipoIndice i;
  i = Pesquisa(Ch, p, T);
  if (i <M)
  memcpy(T[i].Chave, RETIRADO, N);
  else printf("Registro nao esta presente\n");
}
```

Programa E.30 *Rotula grafo e atribui valores para o arranjo g*

```
void Atribuig (TipoGrafo *Grafo,
               TipoArranjoArestas L,
               Tipog g)
{ int i, u, Soma;
  TipoValorVertice v;  TipoAresta a;
  for (i = Grafo->NumVertices - 1; i >= 0; i--) g[i] = INDEFINIDO;
  for (i = Grafo->NumArestas - 1; i >= 0; i--)
    { a = L[i];  Soma = 0;
      for (v = Grafo->r - 1; v >= 0; v--)
        { if (g[a.Vertices[v]] == INDEFINIDO)
          { u = a.Vertices[v]; g[u] = Grafo->NumArestas; }
          else Soma += g[a.Vertices[v]];
        }
        g[u] = a.Peso - Soma;
        if (g[u] < 0) g[u] = g[u]+(Grafo->r-1)*Grafo->NumArestas;
    }
}
```