Continuação do Programa G.26

```
short ExisteAresta(TipoAresta *Aresta,
                   TipoGrafo *Grafo)
{ Tipor v;
  TipoValorAresta A1;
  int Aux;
  short EncontrouAresta;
  EncontrouAresta = FALSE;
  for(v = 0; v < Grafo->r; v++)
  { Aux = Grafo->Prim[Aresta->Vertices[v]];
    while (Aux != -1 && !EncontrouAresta)
      { A1 = Aux % Grafo->NumArestas;
        if (ArestasIguais(Aresta->Vertices, &A1, Grafo))
        EncontrouAresta = TRUE;
        Aux = Grafo->Prox[Aux];
      }
  }
  return EncontrouAresta;
}

TipoAresta RetiraAresta(TipoAresta *Aresta,
                        TipoGrafo *Grafo)
{ int Aux, Prev, i;
  TipoValorAresta A1;
  Tipor v;
  for (v = 0; v < Grafo->r; v++)
  { Prev = INDEFINIDO;
    Aux = Grafo->Prim[Aresta->Vertices[v]];
    A1 = Aux % Grafo->NumArestas;
    while(Aux >= 0 &&
          !ArestasIguais(Aresta->Vertices, &A1, Grafo))
    { Prev = Aux;
      Aux = Grafo->Prox[Aux];
      A1 = Aux % Grafo->NumArestas;
    }
    if (Aux >= 0)
    { if (Prev == INDEFINIDO)
      Grafo->Prim[Aresta->Vertices[v]] = Grafo->Prox[Aux];
      else Grafo->Prox[Prev] = Grafo->Prox[Aux];
    }
  }
  TipoAresta Resultado = Grafo->Arestas[A1];
  for (i = 0; i < Grafo->r; i++)
    Grafo->Arestas[A1].Vertices[i] = INDEFINIDO;
  Grafo->Arestas[A1].Peso = INDEFINIDO;
  return Resultado;
}
```


Continuação do Programa G.26

```
void ImprimeGrafo(TipoGrafo *Grafo)
{ int i, j;
  printf(" Arestas: Num Aresta, Vertices, Peso \n");
  for (i = 0; i < Grafo->NumArestas; i++)
  { printf ("%2d", i);
    for (j = 0; j < Grafo->r; j++)
      printf ("%3d", Grafo->Arestas[i].Vertices[j]);
    printf ("%3d\n", Grafo->Arestas[i].Peso);
  }
  printf ("Lista arestas incidentes a cada vertice:\n");
  for (i = 0 ; i < Grafo->NumVertices; i++)
  { printf ("%2d", i);
    j = Grafo->Prim[i];
    while (j != INDEFINIDO)
      { printf ("%3d", j % Grafo->NumArestas);
        j = Grafo->Prox[j];
      }
    printf("\n");
  }
}
/* operadores para obter a lista de arestas incidentes a um vertice */
short ListaIncVazia(TipoValorVertice *Vertice,
                    TipoGrafo *Grafo)
{ return Grafo->Prim[*Vertice] == -1; }

TipoApontador PrimeiroListaInc(TipoValorVertice *Vertice,
                               TipoGrafo *Grafo)
{ return Grafo->Prim[*Vertice]; }

void ProxArestaInc(TipoValorVertice *Vertice,
                   TipoGrafo *Grafo,
                   TipoValorAresta *Inc,
                   TipoPesoAresta *Peso,
                   TipoApontador *Prox,
                   short *FimListaInc)
/* Retorna Inc apontado por Prox */
{ *Inc = *Prox % Grafo->NumArestas;
  *Peso = Grafo->Arestas[*Inc].Peso;
  if (Grafo->Prox[*Prox] == INDEFINIDO)
  *FimListaInc = TRUE;
  else *Prox = Grafo->Prox[*Prox];
}
```