

Continuação do Programa 7.26

```

procedure ImprimeGrafo (var Grafo : TipoGrafo);
var i, j : integer;
begin
  writeln (' Arestas: Num Aresta, Vertices, Peso ');
  for i := 0 to Grafo.NumArestas - 1 do
    begin
      write (i:2);
      for j := 0 to Grafo.r - 1 do write (Grafo.Arestas[i].Vertices[j]:3);
      writeln (Grafo.Arestas[i].Peso:3);
    end;
  writeln ('Lista arestas incidentes a cada vertice: ');
  for i := 0 to Grafo.NumVertices - 1 do
    begin
      write (i:2);
      j := Grafo.Prim[i];
      while j <> INDEFINIDO do
        begin
          write (j mod Grafo.NumArestas:3);
          j := Grafo.Prox[j];
        end;
      writeln;
    end;
  end; { ImprimeGrafo }

{--Operadores para obter a lista de arestas incidentes a um vertice--}
function ListaIncVazia (var Vertice: TipoValorVertice;
var Grafo: TipoGrafo): boolean;
begin
  ListaIncVazia := Grafo.Prim[Vertice] = -1;
end;

function PrimeiroListaInc (var Vertice: TipoValorVertice;
var Grafo: TipoGrafo): TipoApontador;
begin
  PrimeiroListaInc := Grafo.Prim[Vertice];
end;

procedure ProxArestaInc (var Vertice: TipoValorVertice;
var Grafo: TipoGrafo;
var Inc: TipoValorAresta;
var Peso: TipoPesoAresta;
var Prox: TipoApontador;
var FimListaInc: boolean);
{--Retorna Inc apontado por Prox--}
begin
  Inc := Prox mod Grafo.NumArestas;
  Peso := Grafo.Arestas[Inc].Peso;
  if Grafo.Prox[Prox] = INDEFINIDO
  then FimListaInc := true
  else Prox := Grafo.Prox[Prox];
end; { ProxArestaInc }

```

O programa para testar os operadores do tipo abstrato de dados hipergrafo pode ser visto no Programa 7.27. Observe que o procedimento *InserereAresta* agora é chamado apenas uma vez para inserir uma aresta no grafo não direcionado em vez de duas vezes, como nas Seções 7.2.1, 7.2.2 e 7.2.3.

Programa 7.27 Programa teste para operadores do tipo abstrato de dados hipergrafo

```

program TestaOperadoresTADHipergrafo;
{-- Entram aqui tipos do Programa 7.23 ou do Programa 7.25 --}
var
  Ap : TipoApontador;
  i, j : integer;
  Inc : TipoValorAresta;
  Al : TipoValorAresta;
  V1 : TipoValorVertice;
  Aresta : TipoAresta;
  Peso : TipoPesoAresta;
  Grafo : TipoGrafo;
  FimListaInc: boolean;
{-- Entram aqui operadores do Programa 7.24 ou do Programa 7.26 --}
begin {-- Programa principal --}
  write ('Hipergrafo r: '); readln (Grafo.r);
  write ('No. vertices: '); readln (Grafo.NumVertices);
  write ('No. arestas: '); readln (Grafo.NumArestas);
  FGVazio (Grafo);
  for i := 0 to Grafo.NumArestas-1 do
    begin
      write ('Inserere Aresta e Peso: ');
      for j := 0 to Grafo.r - 1 do read (Aresta.Vertices[j]);
      readln (Aresta.Peso);
      InserereAresta (Aresta, Grafo);
    end;
  ImprimeGrafo (Grafo);
  readln;
  write ('Lista arestas incidentes ao vertice: '); read (V1);
  if not ListaIncVazia (V1, Grafo)
  then begin
    Ap := PrimeiroListaInc (V1, Grafo);
    FimListaInc := false;
    while not FimListaInc do
      begin
        ProxArestaInc (V1, Grafo, Inc, Peso, Ap, FimListaInc);
        write (Inc mod Grafo.NumArestas:2, ' (', Peso, ')');
        end;
        writeln; readln;
      end
    else writeln ('Lista vazia');
    write ('Existe aresta: ');
    for j := 0 to Grafo.r - 1 do read (Aresta.Vertices[j]);
    readln;
  end;

```