

**Programa 7.26** Operadores sobre grafos implementados como listas de incidência usando arranjos

```

function ArestasIguais (var V1 : TipoArranjoVertices;
var NumAresta: TipoValorAresta;
var Grafo : TipoGrafo): boolean;

var i, j: Tipor;
Aux: boolean;
begin
  Aux := true;
  i := 0;
  while (i < Grafo.r) and Aux do
    begin
      j := 0;
      while (V1[i] <> Grafo.Arestas[NumAresta].Vertices[j]) and
        (j < Grafo.r) do j := j + 1;
      if j = Grafo.r then Aux := false;
      i := i + 1;
    end;
  ArestasIguais := Aux;
end; { ArestasIguais }

procedure FGVazio (var Grafo: TipoGrafo);
var i: integer;
begin
  Grafo.ProxDisponivel := 0;
  for i := 0 to Grafo.NumVertices - 1 do Grafo.Prim[i] := -1;
end; { FGVazio }

procedure InsereAresta (var Aresta: TipoAresta;
var Grafo: TipoGrafo);
var i, Ind: integer;
begin
  if Grafo.ProxDisponivel = MAXNUMARESTAS + 1
  then writeln ('Nao ha espaco disponivel para a aresta')
  else begin
    Grafo.Arestas[Grafo.ProxDisponivel] := Aresta;
    for i := 0 to Grafo.r - 1 do
      begin
        Ind := Grafo.ProxDisponivel + i * Grafo.NumArestas;
        Grafo.Prox[Ind] :=
          Grafo.Prim[Grafo.Arestas[Grafo.ProxDisponivel].Vertices[i]];
        Grafo.Prim[Grafo.Arestas[Grafo.ProxDisponivel].Vertices[i]] := Ind;
      end;
    end;
    Grafo.ProxDisponivel := Grafo.ProxDisponivel + 1;
  end; { InsereAresta }

function ExisteAresta (var Aresta: TipoAresta;
var Grafo: TipoGrafo): boolean;

```

Continuação do Programa 7.26

```

var v
  Al : Tipor;
  Aux : integer;
  EncontrouAresta: boolean;
begin
  EncontrouAresta := false;
  for v := 0 to Grafo.r - 1 do
    begin
      Aux := Grafo.Prim[Aresta.Vertices[v]];
      while (Aux <> -1) and not EncontrouAresta do
        begin
          Al := Aux mod Grafo.NumArestas;
          if ArestasIguais (Aresta.Vertices, Al, Grafo)
          then EncontrouAresta := true;
          Aux := Grafo.Prox[Al];
        end;
      end;
      ExisteAresta := EncontrouAresta;
    end; { ExisteAresta }

function RetiraAresta (var Aresta: TipoAresta;
var Grafo: TipoGrafo): TipoAresta;
var Aux, Prev, i: integer;
  Al : TipoValorAresta;
  v : Tipor;
begin
  for v := 0 to Grafo.r - 1 do
    begin
      Prev := INDEFINIDO;
      Aux := Grafo.Prim[Aresta.Vertices[v]];
      Al := Aux mod Grafo.NumArestas;
      while (Aux >= 0) and
        not ArestasIguais (Aresta.Vertices, Al, Grafo) do
        begin
          Prev := Aux;
          Aux := Grafo.Prox[Al];
        end;
      Al := Aux mod Grafo.NumArestas;
      end;
      if Aux >= 0
      then begin { Achou }
          if Prev = INDEFINIDO
          then Grafo.Prim[Aresta.vertices[v]] := Grafo.Prox[Al];
          else Grafo.Prox[Prev] := Grafo.Prox[Al];
          end;
        { else writeln ('Nao existe aresta ou foi retirada antes'); }
      end;
      RetiraAresta := Grafo.Arestas[Al];
      for i := 0 to Grafo.r - 1 do Grafo.Arestas[Al].Vertices[i] := INDEFINIDO;
      Grafo.Arestas[Al].Peso := INDEFINIDO;
    end; { RetiraAresta }
  end;

```