***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              : Tipor;
    A1             : TipoValorAresta;
    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
      A1 := Aux mod Grafo.NumArestas;
      if ArestasIguais (Aresta.Vertices, A1, Grafo)
      then EncontrouAresta := true;
      Aux := Grafo.Prox[Aux];
      end;
    end;
  ExisteAresta := EncontrouAresta;
end; { ExisteAresta }

function RetiraAresta (var Aresta: TipoAresta;
                       var Grafo : TipoGrafo): TipoAresta;
var Aux, Prev, i: integer;
    A1          : TipoValorAresta;
    v           : Tipor;
begin
  for v := 0 to Grafo.r - 1 do
    begin
    Prev := INDEFINIDO;
    Aux := Grafo.Prim[Aresta.Vertices[v]];
    A1 := Aux mod Grafo.NumArestas;
    while (Aux >= 0) and
      not ArestasIguais (Aresta.Vertices, A1, Grafo) do
      begin
      Prev := Aux;
      Aux := Grafo.Prox[Aux];
      A1 := Aux mod Grafo.NumArestas;
      end;
    if Aux >= 0
    then begin  { Achou }
         if Prev = INDEFINIDO
         then Grafo.Prim[Aresta.vertices[v]] := Grafo.Prox[Aux]
         else Grafo.Prox[Prev] := Grafo.Prox[Aux];
         end;
  { else writeln ('Nao existe aresta ou foi retirada antes'); }
  end;
  RetiraAresta := Grafo.Arestas[A1];
  for i := 0 to Grafo.r-1 do Grafo.Arestas[A1].Vertices[i]:=INDEFINIDO;
  Grafo.Arestas[A1].Peso := INDEFINIDO;
end; { RetiraAresta }
```