

Programa 5.34 Programa principal

```

program RotulaGrafoAciclico;
{--- Entram aqui as estruturas de dados do Programa 5.32 ---}
var M
  N
  r
  Grafo
  L
  GAciclico
  g
  Pesos
  i, j, NGrafosGerados: integer;
  ConjChaves
  ArqEntrada
  ArqSaida
  NomeArq
  : TipoValorVertice;
  : TipoValorAresta;
  : Tipor;
  : TipoGrafo;
  : TipoArranjoArestas;
  : boolean;
  : Tipog;
  : TipoTodosPesos;
  : TipoConjChaves;
  : text;
  : text;
  : string [100];
{--- Entram aqui os operadores do Programa 3.18 ---}
{--- Entram aqui os operadores do Programa 5.22 ---}
{--- Entram aqui os operadores do Programa 5.23 ---}
{--- Entram aqui os operadores do Programa 7.26 ---}
{--- Entram aqui VerticeGrauUm e GrafoAciclico do Programa 7.10 ---}
begin {--- ObtemHashPerfeito ---}
  randomize; {--- Inicializa random para 2^32 valores ---}
  write ('Nome do arquivo com chaves a serem lidas: ');
  readln (NomeArq); assign (ArqEntrada, NomeArq);
  write ('Nome do arquivo para gravar experimento: '); readln (NomeArq);
  assign (ArqSaida, NomeArq); reset (ArqEntrada);
  rewrite (ArqSaida);
  NGrafosGerados := 0; i := 0;
  readln (ArqEntrada, N, M, r);
  while (i < N) and (not eof(ArqEntrada)) do
    begin readln (ArqEntrada, ConjChaves[i]); i := i + 1; end;
  if (i < N)
  then begin
    writeln ('Erro: entrada com menos do que ', N, ' elementos. ');
    exit;
  end;
  repeat
    GeraGrafo (ConjChaves, N, M, r, Pesos, NGrafosGerados, Grafo);
    ImprimeGrafo (Grafo);
    {--- Imprime estrutura de dados ---}
    write ('prim: '); for i:=0 to Grafo.NumVertices - 1 do
      write (Grafo.Prim[i]:3); writeln;
    write ('prox: '); for i:=0 to Grafo.NumArestas*Grafo.r-1 do
      write (Grafo.prox[i]:3); writeln;
    GrafoAciclico (Grafo, L, GAciclico);
    until GAciclico;
    write ('Grafo aciclico com arestas retiradas: ');
    for i := 0 to Grafo.NumArestas - 1 do write (L[i].Peso:3);
    writeln;
  
```

Continuação do Programa 5.34

```

  AtribuiG (Grafo, L, g);
  writeln (ArqSaida, N, ' (N) ');
  writeln (ArqSaida, M, ' (M) ');
  writeln (ArqSaida, r, ' (r) ');
  for j := 0 to Grafo.r - 1 do
    begin
      for i := 1 to MAXTAMCHAVE do write (ArqSaida, Pesos[j][i], ' ');
      for i := 1 to MAXTAMCHAVE do write (Pesos[j][i], ' ');
      writeln (ArqSaida, ' (p', j:1, ' ');
      writeln (' (p', j:1, ' ');
    end;
    for i := 0 to M - 1 do write (ArqSaida, g[i], ' ');
    for i := 0 to M - 1 do write (g[i], ' ');
    writeln (ArqSaida, ' (g) ');
    writeln (' (g) ');
    writeln(ArqSaida, 'No. grafos gerados por GeraGrafo: ', NGrafosGerados);
    close (ArqSaida);
    close (ArqEntrada);
  end. { ObtemHashPerfeito }

```

Programa 5.35 Função de transformação perfeita

```

function hp (Chave
  r
  : TipoChave;
  : Tipor;
  var Pesos: TipoTodosPesos;
  var g
  : Tipog): TipoIndice;
var i, v: integer;
begin
  v := 0;
  for i := 0 to r - 1 do v := v + g[h(Chave, Pesos[i])];
  hp := v mod N;
end; { hp }

```

Programa 5.36 Teste para a função de transformação perfeita

```

program HashingPerfeito;
const
  MAXNUMVERTICES = 100000; {---Numero maximo de vertices---}
  MAXNUMARESTAS = 100000; {---Numero maximo de arestas---}
  MAXR = 5;
  MAXTAMCHAVE = 6; {---Numero maximo de caracteres da chave---}
  MAXNUMCHAVES = 100000; {---Numero maximo de chaves lidas---}
type
  TipoValorVertice = -1..MAXNUMVERTICES;
  TipoValorAresta = 0..MAXNUMARESTAS;
  Tipor
  = 0..MAXR;
  TipoPesos
  = array [1..MAXTAMCHAVE] of integer;

```