

## Continuação do Programa E.33

```

    if (VerticesIguais (&Aresta) || ExisteAresta (&Aresta, Grafo))
    { GrafoValido = FALSE; break; }
    else InsereAresta (&Aresta, Grafo);
  }
  ++(*NGrafosGerados);
} while (!GrafoValido);
} /* Fim GeraGrafo */

```

## Programa E.34 Programa principal

```

/*— Entram aqui as estruturas de dados do Programa E.32 —*/
/*— Entram aqui os operadores do Programa C.18 —*/
/*— Entram aqui os operadores do Programa E.22 —*/
/*— Entra aqui a funcao hash universal do Programa E.23 —*/
/*— Entram aqui os operadores do Programa G.26 —*/
/*— Entram aqui VerticeGrauUm e GrafoAciclico do Programa G.10 —*/
int main(){
  Tipor r;
  TipoGrafo Grafo;
  TipoArranjoArestas L;
  short GAciclico;
  Tipog g;
  TipoTodosPesos Pesos;
  int i, j;
  int NGrafosGerados;
  TipoConjChaves ConjChaves;
  FILE *ArqEntrada;
  FILE *ArqSaida;
  char NomeArq[100];
  printf ("Nome do arquivo com chaves a serem lidas: ");
  scanf ("%s*[\n]", NomeArq);
  printf ("NomeArq = %s\n", NomeArq);
  ArqEntrada = fopen(NomeArq, "r");
  printf ("Nome do arquivo para gravar experimento: ");
  scanf ("%s*[\n]", NomeArq);
  printf ("NomeArq = %s\n", NomeArq);
  ArqSaida = fopen(NomeArq, "w");
  NGrafosGerados = 0; i = 0;
  fscanf (ArqEntrada, "%d %d %d*[\n]", &N, &M, &r);
  Ignore (ArqEntrada, '\n');
  printf ("N=%d, M=%d, r=%d\n", N, M, r);
  while ((i < N) && (!feof(ArqEntrada)))
  { fscanf (ArqEntrada, "%s*[\n]", ConjChaves[i]);
    Ignore (ArqEntrada, '\n');
    printf ("Chave[%d]=%s\n", i, ConjChaves[i]);
    i++;
  }
}

```

## Continuação do Programa E.34

```

if (i != N)
{ printf("Erro: entrada com menos do que ' , N, ' elementos.\n");
  exit(-1);
}
do
{ GeraGrafo (ConjChaves, N, M, r, Pesos, &NGrafosGerados, &Grafo);
  ImprimeGrafo (&Grafo);
  /*—Imprime estrutura de dados—*/
  printf ("prim: ");
  for (i = 0; i < Grafo.NumVertices; i++)
    printf ("%3d ", Grafo.Prim[i]);
  printf ("\n"); printf ("prox: ");
  for (i = 0; i < Grafo.NumArestas * Grafo.r; i++)
    printf ("%3d ", Grafo.Prox[i]);
  printf ("\n");
  GrafoAciclico (&Grafo, L, &GAciclico);
} while (!GAciclico);
printf ("Grafo aciclico com arestas retiradas:");
for(i = 0; i < Grafo.NumArestas; i++) printf ("%3d ", L[i].Peso);
printf ("\n");
Atribuig (&Grafo, L, g);
fprintf (ArqSaida, "%d (N)\n", N);
fprintf (ArqSaida, "%d (M)\n", M);
fprintf (ArqSaida, "%d (r)\n", r);
for (j = 0; j < Grafo.r; j++)
{ for (i = 0; i < MAXTAMCHAVE; i++)
  fprintf (ArqSaida, "%d ", Pesos[j][i]);
  fprintf (ArqSaida, " (%d)\n", j);
}
for (i = 0; i < M; i++) fprintf (ArqSaida, "%d ", g[i]);
fprintf (ArqSaida, " (g)\n");
fprintf (ArqSaida, "No. grafos gerados por GeraGrafo:%d\n",
  NGrafosGerados);
fclose (ArqSaida); fclose (ArqEntrada); return 0;
}

```

## Programa E.35 Função de transformação perfeita

```

TipoIndice hp (TipoChave Chave,
  Tipor r,
  TipoTodosPesos Pesos,
  Tipog g)
{ int i, v;
  v = 0;
  for (i = 0; i < r; i++) v += g[h(Chave, Pesos[i])];
  return (v % N);
} /* hp */

```