**Programa G.27** *Programa teste para operadores do tipo abstrato de dados hipergrafo*

```
{-- Entram aqui tipos do Programa G.23 ou do Programa G.25 --}
{-- Entram aqui operadores do Programa G.24 ou do Programa G.26 --}
int main() {
  TipoApontador Ap;
  int i, j;
  TipoValorAresta Inc;
  TipoValorVertice V1;
  TipoAresta Aresta;
  TipoPesoAresta Peso;
  TipoGrafo Grafo;
  short FimListaInc;
  printf ("Hipergrafo r: "); scanf("%d*[^\n]", &Grafo.r);
  printf ("No. vertices: "); scanf("%d*[^\n]", &Grafo.NumVertices);
  printf ("No. arestas: ");  scanf("%d*[^\n]", &Grafo.NumArestas);
  getchar();
  FGVazio (&Grafo);
  for (i = 0; i < Grafo.NumArestas; i++)
  { printf ("Insere Aresta e Peso: ");
    for (j=0; j < Grafo.r; j++) scanf("%d*[^\n]",&Aresta.Vertices[j]);
    scanf("%d*[^\n]", &Aresta.Peso);
    getchar();
    InsereAresta (&Aresta, &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");
  ImprimeGrafo(&Grafo);
  getchar();
  printf ("Lista arestas incidentes ao vertice: ");
  scanf("%d*[^\n]", &V1);

  if (!ListaIncVazia(&V1, &Grafo))
  { Ap = PrimeiroListaInc(&V1, &Grafo);
    FimListaInc = FALSE;
    while (!FimListaInc)
      { ProxArestaInc (&V1, &Grafo, &Inc, &Peso, &Ap, &FimListaInc);
        printf ("%2d (%d)", Inc % Grafo.NumArestas, Peso);
      }
    printf("\n"); getchar();
  }
  else printf ("Lista vazia\n");

  printf ("Existe aresta: ");
  for (j = 0; j < Grafo.r; j++) scanf("%d*[^\n]", &Aresta.Vertices[j]);
  getchar();
```



*Continuação do Programa G.27*

```
  if (ExisteAresta(&Aresta, &Grafo))
    printf ("Sim\n");
  else printf ("Nao\n");
  printf ("Retira aresta: ");
  for (j = 0; j < Grafo.r; j++) scanf("%d*[^\n]", &Aresta.Vertices[j]);
  getchar();
  if (ExisteAresta(&Aresta, &Grafo))
  { Aresta = RetiraAresta(&Aresta, &Grafo);
    printf ("Aresta retirada:");
    for (i = 0; i < Grafo.r; i++) printf ("%3d", Aresta.Vertices[i]);
    printf ("%4d\n", Aresta.Peso);
  }
  else printf ("Aresta nao existe\n");
  ImprimeGrafo(&Grafo);
  return 0;
}
```

**Programa G.28** *Primeiro refinamento do algoritmo de Kruskal*

```
void Kruskal();
{
1. S = ∅;
2. for(v=0;v < Grafo.NumVertices) CriaConjunto (v);
3. Ordena as arestas de A pelo peso;
4. for (cada (u,v) de A tomadas em ordem ascendente de peso)
5.     if (EncontreConjunto (u) != EncontreConjunto (v) )
6.     { S = S+{(u,v)};
7.       Uniao (u,v);
       }
}
```