

Programa G.8 Programa teste para operadores do tipo grafo de dados grafo

```

/*-- Entra aqui estrutura do tipo grafo do Programa G.2 --*/
/*-- ou do Programa G.4 ou do Programa G.6 --*/
TipoApontador Aux;
int i;
TipoValorVertice V1, V2, Adj;
TipoPeso Peso;
TipoGrafo Grafo, Grafot;
TipoValorVertice NVVertices;
short NArestas;
short FimListaAdj;
/*-- Entram aqui os operadores correspondentes do Programa G.3 --*/
/*-- ou do Programa G.5 ou do Programa G.7 --*/
int main()
{
    /*-- Programa principal --*/
    /*-- NumVertices: definido antes da leitura das arestas --*/
    /*-- NumArestas: inicializado com zero e incrementado a --*/
    /*-- cada chamada de InsereAresta --*/
    printf("Leitura do grafo\n");
    printf("No. vertices:"); scanf("%d%*c\n", &NVVertices); getchar();
    printf("No. arestas:"); scanf("%d%*c\n", &NArestas); getchar();
    Grafo.NumVertices = NVVertices; Grafo.NumArestas = 0; FGVazio(&Grafo);
    for (i = 0; i <= NArestas - 1; i++)
    {
        printf("Insere V1 -- V2 -- Peso:");
        scanf("%d%d%*c\n", &V1, &V2, &Peso); getchar();
        Grafo.NumArestas++;
        InsereAresta(&V1, &V2, &Peso, &Grafo); /* 1 chamada g-direcionado */
        /* InsereAresta(V2, V1, Peso, Grafo); */ /* 2 g-naodirecionado */
    }
    ImprimeGrafo(&Grafo); getchar();
    printf("Insere V1 -- V2 -- Peso:");
    scanf("%d%d%*c\n", &V1, &V2, &Peso); getchar();
    if (ExisteAresta(V1, V2, &Grafo))
        printf("Aresta ja existe\n");
    else {
        Grafo.NumArestas++;
        InsereAresta(&V1, &V2, &Peso, &Grafo);
        /* InsereAresta(V2, V1, Peso, Grafo); */ /* g nao direcionado */
    }
    ImprimeGrafo(&Grafo); getchar();
    printf("Lista adjacentes de:");
    if (!ListaAdjVazia(&V1, &Grafo))
    {
        Aux = PrimeiroListaAdj(&V1, &Grafo); FimListaAdj = FALSE;
        while (!FimListaAdj)
        {
            ProxAdj(&V1, &Grafo, &Adj, &Peso);
            printf("%d (%d)", Adj, &Peso);
            putchar('\n'); getchar();
        }
        printf("Retira aresta V1 -- V2:");
        scanf("%d %d %*c\n", &V1, &V2); getchar();
    }
}

```

Continuação do Programa G.8

```

if (ExisteAresta(V1, V2, &Grafo))
{
    Grafo.NumArestas--;
    RetiraAresta(&V1, &V2, &Peso, &Grafo);
    /* RetiraAresta(V2, V1, Peso, Grafo); */
}
else printf("Aresta nao existe\n");
ImprimeGrafo(&Grafo); getchar();
printf("Existe aresta V1 -- V2:");
scanf("%d%d%*c\n", &V1, &V2); getchar();
if (ExisteAresta(V1, V2, &Grafo)) printf("Sim\n");
else printf("Nao\n");
LiberaGrafo(&Grafo); return 0;
}

```

Programa G.9 Busca em profundidade

```

void VisitaDfs(TipoValorVertice u, TipoGrafo *Grafo,
               TipoValorTempo* Tempo, TipoValorTempo* d,
               TipoValorTempo* t, TipoCor* Cor, short* Antecessor)
{
    char FimListaAdj; TipoValorAresta Peso; TipoApontador Aux;
    TipoValorVertice v; Cor[u] = cinza; (*Tempo)++; d[u] = (*Tempo);
    printf("Visita %d Tempo descoberta: %d cinza\n", u, d[u]); getchar();
    if (!ListaAdjVazia(&u, Grafo))
    {
        Aux = PrimeiroListaAdj(&u, Grafo);
        FimListaAdj = FALSE;
        while (!FimListaAdj)
        {
            ProxAdj(&u, &v, &Peso, &Aux, &FimListaAdj);
            if (Cor[v] == branco)
            {
                Antecessor[v] = u;
                VisitaDfs(v, Grafo, Tempo, d, t, Cor, Antecessor);
            }
        }
    }
    Cor[u] = preto; (*Tempo)++; t[u] = (*Tempo);
    printf("Visita %d Tempo termino: %d preto\n", u, t[u]); getchar();
}

void BuscaEmProfundidade(TipoGrafo *Grafo)
{
    TipoValorVertice x; TipoValorTempo Tempo;
    TipoValorTempo d[MAXNUMVERTICES + 1], t[MAXNUMVERTICES + 1];
    TipoCor Cor[MAXNUMVERTICES + 1];
    short Antecessor[MAXNUMVERTICES + 1];
    Tempo = 0;
    for (x = 0; x <= Grafo->NumVertices - 1; x++)
    {
        Cor[x] = branco; Antecessor[x] = -1;
    }
    for (x = 0; x <= Grafo->NumVertices - 1; x++)
    {
        if (Cor[x] == branco)
            VisitaDfs(x, Grafo, &Tempo, d, t, Cor, Antecessor);
    }
}

```