

Programa E.31 Programa para obter função de transformação perfeita

```

int main()
{
    Ler um conjunto de N chaves;
    Escolha um valor para M;
    do
    {
        Gera os pesos  $p_1[i]$  e  $p_2[i]$  para  $1 \leq i \leq \text{MAXTAMCHAVE}$ ;
        Gera o grafo  $G = (V, A)$ ;
        Atribui  $(G, g, \text{GrafoRotulavel})$ ;
    } while (!GrafoRotulavel);
    Retorna  $p_1[i]$  e  $p_2[i]$  e  $g$ ;
}

```

Programa E.32 Estruturas de dados

```

#define MAXNUMVERTICES 100000 /*—No. maximo de vertices—*/
#define MAXNUMARESTAS 100000 /*—No. maximo de arestas—*/
#define MAXR 5
#define MAXTAMPROX MAXR*MAXNUMARESTAS
#define MAXIAM 1000 /*—Usado Fila—*/
#define MAXTAMCHAVE 6 /*—No. maximo de caracteres da chave—*/
#define MAXNUMCHAVES 100000 /*—No. maximo de chaves lidas—*/
#define INDEFINIDO -1
/*—Tipos usados em GrafoListaInc do Programa 7.25 —*/
typedef int TipoValorVertice;
typedef int TipoValorAresta;
typedef int Tipor;
typedef int TipoMaxTamProx;
typedef int TipoPesoAresta;
typedef TipoValorVertice TipoArranjoVertices[MAXR];
typedef struct TipoAresta {
    TipoArranjoVertices Vertices;
    TipoPesoAresta Peso;
} TipoAresta;
typedef TipoAresta TipoArranjoArestas[MAXNUMARESTAS];
typedef struct TipoGrafo {
    TipoArranjoArestas Arestas;
    TipoValorVertice Prim[MAXNUMVERTICES];
    TipoMaxTamProx Prox[MAXTAMPROX];
    TipoMaxTamProx ProxDisponivel;
    TipoValorVertice NumVertices;
    TipoValorAresta NumArestas;
    Tipor r;
} TipoGrafo;
/*—Tipos usados em Fila do Programa 3.17 —*/
typedef int TipoApontador;
typedef struct {
    TipoValorVertice Chave;
    /* outros componentes */
} TipoItem;

```

Continuação do Programa E.32

```

typedef struct {
    TipoItem Item[MAXIAM + 1];
    TipoApontador Frente, Tras;
} TipoFila;
typedef int TipoPesos[MAXTAMCHAVE];
typedef TipoPesos TipoTodosPesos[MAXR];
typedef int Tipog[MAXNUMVERTICES];
typedef char TipoChave[MAXTAMCHAVE];
typedef TipoChave TipoConjChaves[MAXNUMCHAVES];
typedef TipoValorVertice TipoIndice;
static TipoValorVertice M;
static TipoValorAresta N;

```

Programa E.33 Gera um grafo sem arestas repetidas e sem self-loops

```

/*—Entram aqui Programa E.22 (GeraPesos), Programa E.23 (funcao h) —*/
/*—Programa G.5 (operadores do tipo abstrato de dados Grafo) —*/
void GeraGrafo (TipoConjChaves ConjChaves,
                TipoValorAresta N,
                TipoValorVertice M,
                Tipor r,
                TipoTodosPesos Pesos,
                int *NGrafosGerados,
                TipoGrafo *Grafo)
{
    /* Gera um grafo sem arestas repetidas e sem self-loops */
    int i, j; TipoAresta Aresta; int GrafoValido;

    inline int VerticesIguais (TipoAresta *Aresta)
    {
        int i, j;
        for (i = 0; i < Grafo->r - 1; i++)
        {
            for (j = i + 1; j < Grafo->r; j++)
            {
                if (Aresta->Vertices[i] == Aresta->Vertices[j])
                    return TRUE;
            }
        }
        return FALSE;
    }

    do
    {
        GrafoValido = TRUE; Grafo->NumVertices = M;
        Grafo->NumArestas = N; Grafo->r = r;
        FGVazio (Grafo); *NGrafosGerados = 0;
        for (j = 0; j < Grafo->r; j++) GeraPesos (Pesos[j]);
        for (i = 0; i < Grafo->NumArestas; i++)
        {
            Aresta.Peso = i;
            for (j = 0; j < Grafo->r; j++)
                Aresta.Vertices[j] = h (ConjChaves[i], Pesos[j]);
        }
    }
}

```