```
GRF_tpCondRet GRF_EsvaziarGrafo(void)
{
AE PRINCIPAL
	if (pGrafo == NULL) {
		return GRF_CondRetGrafoNaoExiste;
	} else {
	AE1
		if (pGrafo->pVerCorr == NULL) {
			return GRF_CondRetGrafoVazio;
		} else {
		AE2
			int i = CHAR_MIN; /* id para percorrer */
		AI0
			/* remover todas as arestas (o id da aresta é um char) */
			while (i <= CHAR_MAX) {
				GRF_RemoverAresta((unsigned char) i);
			AIF
				i++;
			} /* if */
		AI1
			LIS_DestruirLista(pGrafo->pListaOr);
		AI2
			LIS_DestruirLista(pGrafo->pListaVer);
		AI3
			VER_DestruirVertice(&pGrafo->pVerCorr);
		AI4
			return GRF_CondRetOK;
		AS2
		} /* if */
	AS1
	} /* if */
AS PRINCIPAL
} /* Fim da função: GRF Esvaziar Grafo */
```

SEQUÊNCIA PRINCIPAL

AE Principal:

- Foi recebido um ponteiro que aponta uma área de memória onde existe um grafo ou NULL.

AS Principal:

- O grafo existia e não era vazio, foi esvaziado e foi retornado GRF_CondRetOK, ou o grafo não existia e foi retornado GRF_CondRetGrafoNaoExiste, ou o grafo estava vazio e foi retornado GRF_CondRetGrafoVazio.

SELEÇÃO DO PRIMEIRO IF

AE: Principal

AS: Principal

AE && (c == T) + B1 => AS

Pela assertiva de entrada, o ponteiro recebido ou aponta um grafo ou é NULL. Como a condição é verdadeira o ponteiro é NULL, ou seja, o grafo não existe. Executando o bloco, foi retornado GRF_CondRetGrafoNaoExiste. Vale AS.

AE && (c == F) + B2 => AS

Pela assertiva de entrada, o ponteiro recebido ou aponta um grafo ou é NULL. Como a condição é falsa o ponteiro não é NULL e o grafo existe. Neste caso o bloco é executado e ou o grafo está vazio e retorna-se GRF_CondRetGrafoVazio ou o grafo é esvaziado e retorna-se GRF_CondRetOK. Vale AS.

SEQUÊNCIA DENTRO DO ELSE DO PRIMEIRO IF

AE1:

- O grafo existe

AS1: Principal

SELEÇÃO DO SEGUNDO IF

AE: AE1

AS: Principal

AE && (c == T) + B1 => AS

Pela assertiva de entrada, o ponteiro recebido aponta um grafo existente. Como a condição é verdadeira o grafo está vazio. Executando o bloco, foi retornado GRF_CondRetGrafoVazio. Vale AS.

AE && (c == F) + B2 => AS

Pela assertiva de entrada, o ponteiro recebido aponta um grafo existente. Como a condição é falsa o grafo recebido não é vazio. Neste caso o bloco é executado, o grafo é esvaziado e retorna-se GRF_CondRetOK. Vale AS.

SEQUÊNCIA DENTRO DO ELSE DO SEGUNDO IF

AE2:

- O grafo existe e não é vazio

AI0:

- i define uma possível aresta a ser removida. i tem o valor inicial de 0.

AI1:

- Todas as arestas do grafo foram destruídas.

AI2:

- As origens e a lista de origens do grafo foram destruídas.

AI3:

- Os vértices e a lista de vértices do grafo foram destruídos.

AI4:

- O ponteiro corrente agora é NULL e o grafo pode ser considerado vazio.

AS2: Principal

REPETIÇÃO

AE: AE2

AS: AI1

AINV:

- Existem 2 conjuntos que separam as 256 arestas possíveis (máximo de um char): Já liberados e a liberar. i define uma possível aresta a ser liberada.

1

AE => AINV

Pela AE, i define a primeira possível aresta a ser liberada e todas as possíveis arestas estão em a liberar. O conjunto já liberados está vazio, vale AINV.

2

AE && (c == F) => AS

Não é possível o caso que não entre ou não complete o primeiro ciclo.

3

AE && (c == T) + B => AINV

Pela AE, i define a primeira possível aresta a ser liberada. Como (c == T), i ainda é menor do que o valor máximo que ele pode atingir. A aresta definida por ele passa do conjunto a liberar para o já liberado e i é reposicionado para outro elemento de a liberar. Vale AINV.

4

AINV && (c == T) + B => AINV

O bloco garante que uma aresta passe de a liberar para já liberado e i seja reposicionado. Vale AINV.

5

AINV && (c == F) => AS

condição falsa: pela AINV, i ultrapassou o limite lógico e todas as arestas estão em já liberado, ou seja, já foram destruídas. Vale AS.

6

Término

Como a cada ciclo, o bloco executado garante que uma aresta passe de a liberar para já liberado e o conjunto a liberar possui um número finito de arestas, a repetição termina em um número finito de passos.

SEQUÊNCIA DENTRO DO BLOCO DO FOR

AIF:

- A aresta definida por i ou não existia ou existia e foi destruída.