Universidade Federal de Uberlândia
Algoritmos e Programação - GGI

Roteiro para Laboratório - GRAFOS PONDERADOS DIRIGIDOS

PARTE 2 - Grafos - Implementação de uma matriz de adjacências para um grafo PONDERADO direcionado usando estruturas dinâmicas (alocação dinâmica de memória).

TAD para o Grafo:

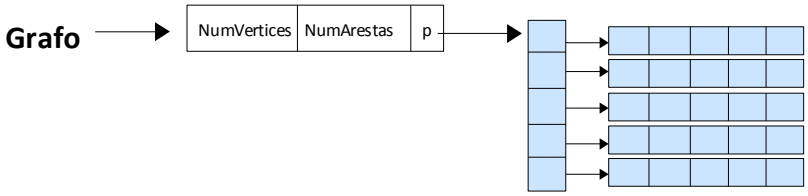
```
typedef int TipoPeso;

struct grafo {
                  int NumVertices;
                  int NumArestas;
                  TipoPeso **Mat;
                  };

typedef struct grafo TipoGrafo;
```

Função de alocação de memória para o Grafo:

- A função devolve a posição de memória de início para esta alocação.
- A função recebe a quantidade de vértices do grafo (N) para que possa ser feita a alocação de memória para a matriz (N x N)




```
TipoGrafo* Cria_grafo(int NVertices)
{
  int i, k;
  TipoGrafo *Grafo ;
  if ( NVertices <= 0) return NULL;
  Grafo = (TipoGrafo*) malloc(sizeof(TipoGrafo));
  if (Grafo == NULL)  return NULL;
  Grafo->Mat = (TipoPeso **) malloc(NVertices*sizeof(TipoPeso*));
  if (Grafo->Mat == NULL) {
        free(Grafo);
        return NULL;
  }
  for(i=0; i<NVertices; i++)  {
     Grafo->Mat[i] = (TipoPeso*) calloc(NVertices,sizeof(TipoPeso));
     if (Grafo->Mat[i]  == NULL)    {
            for (k=0; k<i; k++)
                  free(Grafo->Mat[k]);
            free(Grafo) ;
            return NULL;
     }
  }
  Grafo->NumVertices = NVertices;
  Grafo->NumArestas = 0;
  return Grafo;
}
```

Exercícios:

1) Monte um projeto usando CodeBlocksC++ para manter as funções de manipulação de um grafo ponderado direcionado utilizando matriz de adjacências e alocação dinâmica, de acordo com o tipo de dado (TAD) visto. São as funções:
(a) Preenchimento das arestas
(b) Dados dois vértices A e B, verificar se existe um caminho entre de A para B.

2) Faça uma função para percorrer a matriz de adjacências.

3) Faça uma função para armazenar os arcos na matriz de adjacências.