UNIVERSIDADE ESTADUAL DE PONTA GROSSA

SETOR DE CIÊNCIAS AGRÁRIAS E SUAS TÉCNOLOGIAS

DEPARTAMENTO DE INFORMÁTICA

MAYSA LOVATTO LOPES  
RENANN RODRIGUES DA SILVA

ALGORITIMOS: ESTRUTURA DE DADOS

PONTA GROSSA

2014

MAYSA LOVATTO LOPES

RENANN RODRIGUES DA SILVA

ALGORITIMOS: ESTRUTURA DE DADOS

Trabalho apresentada na disciplina de

Estrutura de dados como requisito

de avaliação parcial referente ao

4º Bimestre. 2º Ano de Informática.

Professor Márcio Augusto de Souza

PONTA GROSSA

2014

//Trabalho de Estrutura de dados Quarto Bimestre

//Grafos usando lista de adjacências com alocação dinâmica

//Acadêmicos: Maysa Lovatto Lopes RA: 13015123

// Renann R. da Silva RA: 13106523

#include<conio.h>

#include<stdio.h>

#define TAM 10

int vertices[TAM],cont=0;

struct no{

int destino;

no \*prox;

};

no \*arestas[TAM];

void inicia\_arestas(){

for(int i=0;i<TAM;i++){

arestas[i] = NULL;

vertices[i]= 0;

}

}

//funçao para inserir arestas

void insere\_aresta(int orig, int dest){

int i=0,j=0;

no \*atual, \*anterior, \*aux;

while(i<cont && vertices[i]!=orig)

i++;

if(i==cont){

printf("\n\tERRO: Vertice origem [%d] nao encontrado", orig);

return;

}

while(j<cont && vertices[j]!=dest)

j++;

if(j==cont){

printf("\n\tERRO: Vertice destino[%d] nao encontrado", dest);

return;

}

anterior = NULL;

atual = arestas[i];

while(atual != NULL){

if(atual->destino == dest){

printf("\n\tERRO: Aresta repetida");

return;

}

anterior = atual;

atual = atual->prox;

}

aux = new(struct no);

aux->destino = dest;

aux->prox = NULL;

if(anterior == NULL){

arestas[i] = aux;

}else

anterior->prox = aux;

printf("\nAresta inserida com sucesso [%d->%d]", orig, dest);

}

//função para inserir vertices

void insere\_vertice(int valor){

if(cont >= TAM){

printf("\n\tERRO: vertice nao inserido:");

printf("\n\tExcedido o numero de vertices");

return;

}

vertices[cont] = valor;

cont++;

printf("\nVertice inserido [%d]", valor);

}

//função de remoção

void remove\_aresta(int orig, int dest){

int i=0,j=0;

no \*anterior, \*atual;

while(i<cont && vertices[i]!=orig)

i++;

if(i==cont){

printf("\n\tERRO: Vertice origem [%d] nao encontrado", orig);

return;

}

while(j<cont && vertices[j]!=dest)

j++;

if(j==cont){

printf("\n\tERRO: Vertice destino [%d] nao encontrado", dest);

return;

}

atual = arestas[i];

anterior = NULL;

while(atual != NULL){

if(atual->destino == dest){

if(atual->prox == NULL && anterior == NULL){//caso tenha apenas 1 elemento da lista de adjacencias

arestas[i] = NULL;

delete(atual);

printf("\nAresta removida com sucesso!");

return;

}

if(atual->prox != NULL && anterior == NULL){//caso seja o elemento do meio da lista

arestas[i] = atual->prox;

delete(atual);

printf("\nAresta removida com sucesso!!");

return;

}

else{

anterior->prox = atual->prox; //caso seja o ultimo elemento da lista

delete(atual);

printf("\nAresta removida com sucesso!!!");

return;

}

}

anterior = atual;

atual = atual->prox;

}

printf("\n\tERRO: Aresta nao existente [%d]->[%d]",orig,dest);

}

//função de impressão

void imprime(){

int i;

no \*aux;

for(i=0;i<TAM && vertices[i] != NULL;i++){

printf("\n%d ->", vertices[i]);

aux = arestas[i];

while(aux != NULL){

printf("%d, ", aux->destino);

aux = aux->prox;

}

}

}

main(){

insere\_vertice(1);

insere\_vertice(10);

insere\_vertice(100);

insere\_vertice(1000);

insere\_vertice(10000);

insere\_vertice(2);

printf("\n\n");

insere\_aresta(1, 10);

insere\_aresta(1, 100);

insere\_aresta(1,1000);

insere\_aresta(1,10000);

insere\_aresta(10,10);

insere\_aresta(10,100);

insere\_aresta(10,1000);

insere\_aresta(100,10000);

insere\_aresta(10000,1);

insere\_aresta(1000,1);

printf("\n\n");

remove\_aresta(1000,1);

remove\_aresta(1,10);

remove\_aresta(10,1000);

remove\_aresta(1,2);

printf("\n\n");

imprime();

getch();

}