代码

#include <iostream>

#include<fstream>

#include<cstdio>

using namespace std;

#define MAX 100

class AMGlist;

typedef char headtype;

class AMGnode {

private:

int num;

AMGnode\* next=nullptr;

int cost;

public:

AMGnode() {

num = NULL;

next = nullptr;

cost = NULL;

}

AMGnode(int numm, AMGnode\* nextt,int costt) {

this->num = numm;

this->next = nextt;

this->cost = costt;

}

int Getnum() {

return this->num;

}

AMGnode\* Getnext() {

return this->next;

}

int Getcost() {

return this->cost;

}

void changenum(int numm) {

this->num = numm;

}

void changenext(AMGnode\* &nextt) {

this->next = nextt;

}

void changecost(int numm) {

this->cost = numm;

}

friend void InsertVex(AMGlist& G, headtype v);

friend void DeleteVex(AMGlist& G, headtype v);

friend void InsertArc(AMGlist& G, headtype v, headtype w);

friend void DeleteArc(AMGlist& G, headtype v, headtype w);

};

class AMGhead {

private:

headtype head;

AMGnode\* next=nullptr;

public:

AMGhead() {

head = NULL;

next = nullptr;

}

AMGhead(headtype headd, AMGnode\*& nextt) {

this->head = headd;

this->next = nextt;

}

AMGhead(headtype headd) {

this->head = headd;

}

headtype Gethead() {

return this->head;

}

AMGnode\* Getnext() {

return this->next;

}

void changehead(headtype numm) {

this->head = numm;

}

void changenext(AMGnode\* &nextt) {

this->next = nextt;

}

friend void InsertVex(AMGlist& G, headtype v);

friend void DeleteVex(AMGlist& G, headtype v);

friend void InsertArc(AMGlist& G, headtype v, headtype w);

friend void DeleteArc(AMGlist& G, headtype v, headtype w);

};

class AMGlist {

private:

AMGhead\* Head;

int vexnum, arcnum;

public:

~AMGlist() {

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

if (this->Head[i].Getnext()) {

AMGnode\* temp = this->Head[i].Getnext();

while (temp != nullptr) {

AMGnode\* temp1 = temp->Getnext();

delete temp;

temp = temp1;

}

AMGnode\* p1 = nullptr;

Head[i].changenext(p1);

}

}

delete[]Head;

}

int locatehead(headtype head1) {

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

if (Head[i].Gethead() == head1) { return i; break; }

}

return -1;

}

void creatAMGlist(int v,int a) {

this->vexnum = v;

this->arcnum = a;

this->Head = new AMGhead[vexnum];

headtype head1,head2;

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

cin >> head1;

Head[i].changehead(head1);

}

int m, n;

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

cin >> head1 >> head2;

m = this->locatehead(head1);

n = this->locatehead(head2);

AMGnode\* p1 = new AMGnode(n, Head[m].Getnext(), 0);

Head[m].changenext(p1);

AMGnode\* p2 = new AMGnode(m, Head[n].Getnext(), 0);

Head[n].changenext(p2);

}

}

void creatAMGlist(const char\* path) {

ifstream ifs;

ifs.open(path, ios::in);

if (ifs.is\_open()) {

int m, n;

headtype head1, head2;

ifs >> m >> n;

this->vexnum = m;

this->arcnum = n;

this->Head = new AMGhead[MAX];

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

ifs >> head1;

Head[i].changehead(head1);

}

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

ifs >> head1 >> head2;

m = this->locatehead(head1);

n = this->locatehead(head2);

AMGnode\* p1 = new AMGnode(n, Head[m].Getnext(), 0);

Head[m].changenext(p1);

AMGnode\* p2 = new AMGnode(m, Head[n].Getnext(), 0);

Head[n].changenext(p2);

}

ifs.close();

}

else cout << "wrong";

}

friend void InsertVex(AMGlist& G, headtype v);

friend void DeleteVex(AMGlist& G, headtype v);

friend void InsertArc(AMGlist& G, headtype v, headtype w);

friend void DeleteArc(AMGlist& G, headtype v, headtype w);

};

void DeleteArc(AMGlist& G, headtype v, headtype w) {

int n = G.locatehead(v);

int m = G.locatehead(w);

AMGnode\* temp1=nullptr, \* temp2;

temp1 = G.Head[n].Getnext();

if (temp1 && temp1->Getnum() == m) {

temp2 = temp1->Getnext();

G.Head[n].changenext(temp2);

delete temp1;

G.arcnum--;

}

while (temp1!=nullptr) {

temp2 = temp1;

temp1 = temp1->Getnext();

if (temp1->Getnum() == m) {

AMGnode\* t = temp1;

temp2->changenext(t);

delete temp1;

G.arcnum--;

}

}

}

void InsertArc(AMGlist& G, headtype v, headtype w) {

int n = G.locatehead(v);

int m = G.locatehead(w);

AMGnode\* p = new AMGnode(m, G.Head[m].Getnext(), 0);

G.Head[n].changenext(p);

}

void InsertVex(AMGlist& G, headtype v) {

if (G.vexnum + 1 > MAX) { cout << "wrong"; return; }

else {

G.Head[G.vexnum].changehead(v);

G.vexnum++;

}

}

void DeleteVex(AMGlist& G, headtype v) {

if (G.locatehead(v) == -1) { cout << "wrong"; return; }

else {

int n = G.locatehead(v);

if (G.Head[n].Getnext()) {

AMGnode\* temp = G.Head[n].Getnext();

while (temp != nullptr) {

DeleteArc(G,G.Head[temp->Getnum()].Gethead(),G.Head[n].Gethead());

AMGnode\* temp1 = temp->Getnext();

delete temp;

temp = temp1;

}

AMGnode\* p1 = nullptr;

G.Head[n].changenext(p1);

for (int i = n; i < G.vexnum - 1; i++) {

G.Head[i] = G.Head[i + 1];

}

}

G.vexnum--;

}

}