**矩阵的压缩存储实现**

#include <stdio.h>

#include <memory> //C语言内存操作头文件

#define ElemType int

#define MAXSIZE 100

typedef struct Triple { //三元组结构体

int i;

int j;

ElemType e;

}Triple;

typedef struct SMatrix {

Triple data[MAXSIZE]; //定义了一个数组，元素类型为三元组 相当于（int a[10]）

int mu; //行数

int nu; //列数

int tu; //非零元素的个数

}SMatrix;

//这里是C语言读取文件的方法 后续要用C++尝试重写

void CreatMatrix(SMatrix\* M) {

FILE\* fp;

errno\_t err=fopen\_s(&fp,"D://Matrix.txt", "r"); //将矩阵的数据写入txt文档 然后创建文件指针 以read的方式打开

if (fp == NULL) { //文件打开失败

exit(0); //C++里不用引入头文件就可以用exit函数

}

fscanf\_s(fp, "%d %d", &M->mu, &M->nu); //读第一行数据，从指针所指的文件当前位置开始，读取行和列两个数据并放入mu,nu中

int value;

int k = 0;

for (int i = 0; i < M->mu; i++) {

for (int j = 0; j < M->nu; j++) {

fscanf\_s(fp, "%d", &value);

if (value != 0) {

M->data[k].e = value;

M->data[k].i = i;

M->data[k].j = j;

k++;

}

}

}

M->tu = k;

fclose(fp);

}

void PrintMatrix(SMatrix\* M) {

printf("row=%d , col=%d\n", M->mu, M->nu);

for (int i = 0; i < M->tu; i++) {

printf("(%d %d %d)\n", M->data[i].i,M->data[i].j,M->data[i].e);

}

}

//用M矩阵拷贝一个T矩阵

void CopyMatrix(SMatrix\* M, SMatrix\* T) {

T->mu = M->mu;

T->nu = M->nu;

T->tu = M->tu;

for (int i = 0; i < M->tu; i++) {

T->data[i].i = M->data[i].i;

T->data[i].j = M->data[i].j;

T->data[i].e = M->data[i].e;

}

}

//转置

void TransposeMatrix(SMatrix\* M, SMatrix\* T) {

T->mu = M->nu;

T->nu = M->mu;

T->tu = M->tu;

int q = 0;

if (M->tu != 0) {

for (int col = 0; col < M->nu; col++) {

for (int p = 0; p < M->tu; p++) {

if (M->data[p].j == col) {

T->data[q].i = M->data[p].j;

T->data[q].j = M->data[p].i;

T->data[q].e = M->data[p].e;

q++;

}

}

}

}

}

void main() {

SMatrix sm,sm1;

memset(&sm,0,sizeof(sm)); // memset 初始化内存 //第一个参数 : 初始化内存的首地址 //第二个参数 : 将内存初始化成什么值

//第三个参数: 从首地址开始多少个值设置为0

CreatMatrix(&sm);

PrintMatrix(&sm);

//CopyMatrix(&sm,&sm1);

//PrintMatrix(&sm1);

TransposeMatrix(&sm, &sm1);

PrintMatrix(&sm1);

}