编程题参考答案

评分标准

|  |  |  |  |
| --- | --- | --- | --- |
| 类目 | 细节 | A卷 | B卷 |
| 头文件  4分 | using namespace std; | 2 | 2 |
| #include <fstream> | 2 | 2 |
| 文件  14分 | 创建文件对象 | 2 | 2 |
| 文件路径格式是否正确 | 2 | 2 |
| 文件正确方式打开：1）采用读和写文件分别操作（A卷可能较多采用）或2）同时读写（B卷可能较多采用） | 6 | 6 |
| 关闭文件 | 2 | 2 |
| 读写过程对文件对象是否做过有效性的判断 | 2 | 2 |
| 数据  22分 | 读文件头得到矩阵数目（A卷）和矩阵行列数 | 4 | 4 |
| 正确动态创建各矩阵容纳空间（A卷比较难，需用到指针数组保存各个空间地址） | 6 | 6 |
| 正确读入各矩阵 | 4 | 4 |
| 定位并正确写入各矩阵数据，注意包括行列值（A卷） | 6 | 6 |
| 正确删除动态空间 | 2 | 2 |
| 计算  10分 | 矩阵转置或加权相加 | 10 | 10 |
|  |  |  |  |

bool actA(char \* file)//A卷

{

fstream matrices;

matrices.open (file, ios::in | ios::out | ios::binary);

bool result = true;

if (matrices.good ()) {

// 读矩阵数

int num;

matrices.read ((char \*)&num, 4);

cout<<num<<endl;

if (num <= 0) {

matrices.close ();

return false;

}

// 用于存放矩阵行列值

int \* rows = new int[num];

int \* cols = new int[num];

// 用于原矩阵与转置矩阵的动态指针数组

double \*\* bufs = new double \*[num];

double \*\* bufs2 = new double \*[num];

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

bufs[i] = NULL;

bufs2[i] = NULL;

// 读矩阵行列数

int row = 0, col = 0;

matrices.read ((char \*)&row, 4);

matrices.read ((char \*)&col, 4);

rows[i] = row;

cols[i] = col;

if (!matrices.eof () && row > 0 && col > 0) {

cout<<row<<'\t'<<col<<endl;

// 存放原矩阵与转置矩阵的缓冲区

bufs[i] = new double[row \* col];

bufs2[i] = new double[row \* col];

matrices.read ((char \*)bufs[i], 8 \*row\*col);

for (int m = 0; m < row; m++) {

for (int n = 0; n < col; n++) {

cout<<setw(8)<<bufs[i][m\*col + n];

}

cout<<endl;

}

}

}

// 转置矩阵

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

for (int m = 0; m < rows[i]; m++) {

for (int n = 0; n <cols[i]; n++) {

bufs2[i][n\*rows[i] + m] = bufs[i][m\*cols[i] + n];

}

}

}

// 输出转置后矩阵，用于验证

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

cout<<cols[i]<<'\t'<<rows[i]<<endl;

for (int n = 0; n < cols[i]; n++) {

for (int m = 0; m <rows[i]; m++) {

cout<<setw(8)<<bufs2[i][n\*rows[i] + m] ;

}

cout<<endl;

}

cout<<endl;

}

// 写转置后矩阵回文件，注意原行列值颠倒

matrices.clear (0);

matrices.seekp (4, ios::beg);

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

matrices.write ((char\*)&cols[i], 4);

matrices.write ((char\*)&rows[i], 4);

matrices.write ((char\*)bufs2[i], 8\*cols[i]\*rows[i]);

}

// 删除动态资源

delete []rows;

delete []cols;

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

delete []bufs[i];

delete []bufs2[i];

}

delete []bufs;

delete []bufs2;

}

else {

result = false;

}

matrices.close ();

return result;

}

bool actB(char \* file) //B卷

{

fstream images;

images.open (file, ios::in | ios::out | ios::binary);

bool result = true;

if (images.good ()) {

// 读矩阵数

int num;

images.read ((char \*)&num, 4);

cout<<num<<endl;

// 读矩阵行列数

int row = 0;

int col = 0;

images.read ((char \*)&row, 4);

images.read ((char \*)&col, 4);

cout<<row<<'\t'<<col<<endl;

if (num <= 0 || row <=0 || col <= 0) {

images.close ();

return false;

}

// 读加权系数

double \* cofs = new double[num];

images.read ((char \*)cofs, 8\*num);

for (int i = 0; i < num; i++)

cout<<cofs[i]<<endl;

// 用于原矩阵的动态指针数组

double \*\* bufs = new double \*[num];

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

bufs[i] = NULL;

if (!images.eof ()) {

// 存放原矩阵的缓冲区

bufs[i] = new double[row \* col];

images.read ((char \*)bufs[i], 8 \*row\*col);

for (int m = 0; m < row; m++) {

for (int n = 0; n < col; n++) {

cout<<setw(8)<<bufs[i][m\*col + n];

}

cout<<endl;

}

cout<<endl;

}

}

double \* add = new double[row \* col];

for (int i = 0; i < row \* col; i++) {

add[i] = 0;

for (int j = 0; j < num; j++)

add[i]+= (cofs[j] \* bufs[j][i]);

}

// 输出加权计算矩阵，用于验证

for (int m = 0; m < row; m++) {

for (int n = 0; n <col; n++) {

cout<<setw(8)<<add[m\*col + n] ;

}

cout<<endl;

}

// 写加权计算矩阵到文件

images.clear (0);

images.write ((char \*)add, 8\*row\*col);

// 删除动态资源

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

delete []bufs[i];

}

delete []bufs;

delete []cofs;

delete []add;

}

else {

result = false;

}

images.close ();

return result;

}