**System Analysis and Design**

**Homework Assignment 3（期中）**

**Due April 30**

Examine the source code of a mesh application in the folder Airfoil in the Repository homework. According to your understanding of mesh applications, try to complete the following tasks:

1. What is the mesh file format in file new\_grid.dat? Write out your answer in C structures.

（崔子潇）

这是一个二维图形的网格表示，具体数据结构如下：

typedef struct node{

double x,y;//position(x,y)

}Node;

typedef struct cell{

int n1,n2,n3,n4;//four points to construct a cell

}Cell;

typedef struct edge{

Node n1,n2;

Cell c1,c2;

}Edge;

typedef struct bedge{

int n1,n2;

int c;

int b;//bound

}Bedge;

typedef struct graph{

Node\* nodes;

Cell\* cells;

Edge\* edges;

Edge\* bedges;

}Graph;

1. How to load in a mesh from a mesh file like new\_grid.dat? Write out your answer as a C function.

（丁维力）

以上的数据结构仅供参考，只是清楚一点表示出来，老师的代码中并没有使用上述数据结构。因此这里分析老师的代码并将载入部分修改为一个函数：

int Load(double \*x, int \*cell, int \*edge, int \*ecell, int \*bedge, int \*becell, int \*bound)

{

printf("reading in grid \n");

FILE \*fp;

if ((fp = fopen("./new\_grid.dat", "r")) == NULL)

{ //open the file

printf("can't open file new\_grid.dat\n");

return 0;

}

int nnode, ncell, nedge, nbedge;

if (fscanf(fp, "%d %d %d %d \n", &nnode, &ncell, &nedge, &nbedge) != 4)//get the total num

{

printf("error reading from new\_grid.dat\n");

return 0;

}

x = (double \*)malloc(2 \* nnode \* sizeof(double));

cell = (int \*)malloc(4 \* ncell \* sizeof(int));

edge = (int \*)malloc(2 \* nedge \* sizeof(int));

ecell = (int \*)malloc(2 \* nedge \* sizeof(int));

bedge = (int \*)malloc(2 \* nbedge \* sizeof(int));

becell = (int \*)malloc(nbedge \* sizeof(int));

bound = (int \*)malloc(nbedge \* sizeof(int));

//set variables for graph coloring

int \*cell2edge = (int \*)malloc(4 \* ncell \* sizeof(int));

for (int i = 0; i < 4 \* ncell; ++i)

{

cell2edge[i] = -1;

}

for (int n = 0; n < nnode; n++)//read the nodes

{

if (fscanf(fp, "%lf %lf \n", &x[2 \* n], &x[2 \* n + 1]) != 2)

{

printf("error reading from new\_grid.dat\n");

return 0;

}

}

for (int n = 0; n < ncell; n++)//read the cells

{

if (fscanf(fp, "%d %d %d %d \n", &cell[4 \* n], &cell[4 \* n + 1],

&cell[4 \* n + 2], &cell[4 \* n + 3]) != 4)

{

printf("error reading from new\_grid.dat\n");

return 0;

}

}

for (int n = 0; n < nedge; n++)//read the edeges

{

if (fscanf(fp, "%d %d %d %d \n", &edge[2 \* n], &edge[2 \* n + 1],

&ecell[2 \* n], &ecell[2 \* n + 1]) != 4)//just read

{

printf("error reading from new\_grid.dat\n");

return 0;

}

//deal with the two edge cells of the edge:

for (int i = 0; i < 4; ++i)//set the remained edge of this cell as this edge

{

if (cell2edge[ecell[2 \* n] + i] == -1)

{

cell2edge[ecell[2 \* n] + i] = n;

break;

}

}

for (int i = 0; i < 4; ++i)

{

if (cell2edge[ecell[2 \* n + 1] + i] == -1)

{

cell2edge[ecell[2 \* n + 1] + i] = n;

break;

}

}

}

for (int n = 0; n < nbedge; n++)//read the edge and bound

{

if (fscanf(fp, "%d %d %d %d \n", &bedge[2 \* n], &bedge[2 \* n + 1],

&becell[n], &bound[n]) != 4)

{

printf("error reading from new\_grid.dat\n");

return 0;

}

}

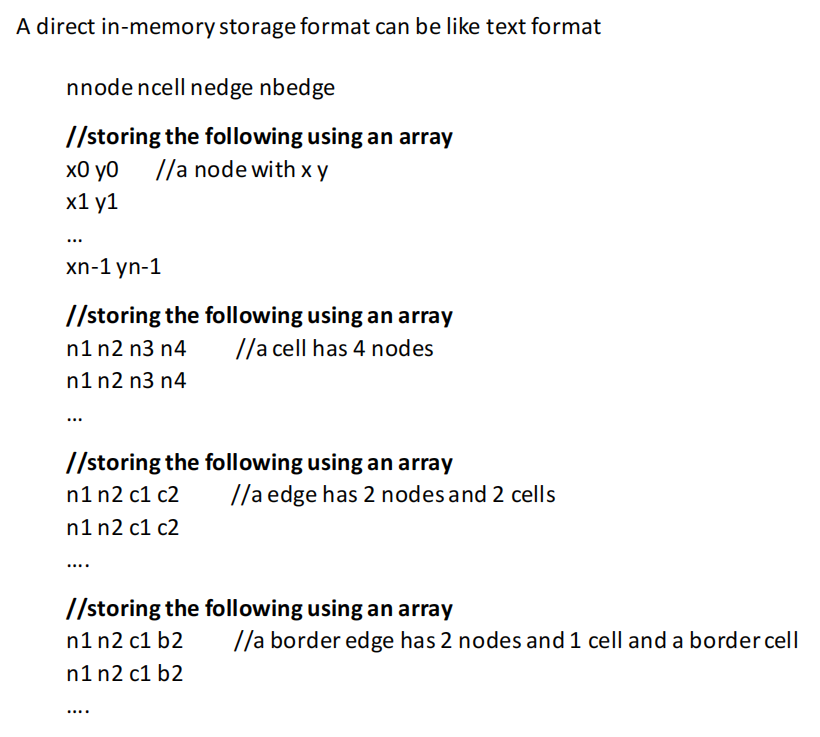
fclose(fp);

return 1;//successful flag

}

1. What is the in-memory storage structure of a mesh? Write out your answer as a fragment of C program.

（胡若晴）



存储点的时候，使用一个大小是点的个数的两倍的一维数组int node[2\*nnode]，数组每两个元素为一组，存放每个点的x坐标和y坐标，比如下标为0和1的数组元素分别存放第0个点的x坐标和y坐标，下标为2和3分别存放第1个点的x坐标和y坐标，以此类推：

int node[2\*nnode];

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

node[i \* 2] = xi;

node[i \* 2 + 1] = yi;

}

存储网格单元的时候，使用一个大小是网格单元的个数的四倍的一维数组int cell[4\*ncell]，数组每四个元素为一组，存放每个网格的四个点：

int cell[4\*ncell];

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

cell[i \* 2] = ni1;

cell[i \* 2 + 1] = ni2;

cell[i \* 2 + 2] = ni3;

cell[i \* 2 + 3] = ni4;

}

存储边的时候，使用两个一维数组，存放每个边的的两个点和两个网格：

int edge[2\*nedge];

int ecell[2\*nedge];

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

edge[i \* 2] = ni1;

edge[i \* 2 + 1] = ni2;

ecell[i \* 2 ] = ci1;

ecell[i \* 2 + 1] = ci2;

}

存储边界边的时候，使用两个一维数组，存放每个边的的两个点、一个网格和一个边界网格：

int ebdge[2\*nbedge];

int becell[nbedge];

int bound[nbedge];

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

ebdge[i \* 2] = ni1;

ebdge[i \* 2 + 1] = ni2;

becell[i] = ci1;

bound[i] = bi2;

}

1. What are the kernel functions as the operations over mesh elements in this mesh application? Write out your answers as C functions.

（李浩天）

根据文档,mesh application的kernel function 包括：

1. 对核心的初始化和退出

2. 非结构化Mesh:Set/Map/Aarg/Plan的创建和删除

3. 输出结果的打印和错误日志

4. Arg/Map结构的校验

5. 定时器

6. 将Dat格式文件输出为特定存储格式的文件（VTK）

7. 并行计算操作

在例程中,kernel函数为

1. WriteMeshToVTKBinary()

将mesh结构写到vtk 二进制文件中.

/\*

\* Write simulation output to binary file

\*/

void WriteMeshToVTKBinary(const char\* filename, double\* nodeCoords\_data, int nnode, int\* cellsToNodes\_data, int ncell, double \*values\_data) {

printf("Writing OutputSimulation to binary file: %s \n",filename);

FILE\* fp;

fp = fopen(filename, "w");

if(fp == NULL) {

printf("can't open file for write %s\n",filename);

exit(-1);

}

// write header

char s[256];

strcpy(s, "# vtk DataFile Version 2.0\n Output from OP2 Volna.\n"); fwrite(s, sizeof(char), strlen(s), fp);

strcpy(s, "BINARY \nDATASET UNSTRUCTURED\_GRID\n\n"); fwrite(s, sizeof(char), strlen(s), fp);

// write vertices

sprintf(s,"POINTS %d double\n", nnode); fwrite(s, sizeof(char), strlen(s), fp);

//double\* nodeCoords\_data;

//nodeCoords\_data = (double\*)nodeCoords->data;

double tmp\_double;

int i = 0;

for (i = 0; i < nnode; ++i) {

tmp\_double = swapEndiannesDouble(nodeCoords\_data[i\*MESH\_DIM ]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

tmp\_double = swapEndiannesDouble(nodeCoords\_data[i\*MESH\_DIM+1]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

tmp\_double = swapEndiannesDouble(0.0);

fwrite(&tmp\_double, sizeof(double), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

// write cells

sprintf(s, "CELLS %d %d\n", ncell, 5\*ncell); fwrite(s, sizeof(char), strlen(s), fp);

int three = 3;

int tmp\_int;

for ( i = 0; i < ncell; ++i ) {

tmp\_int = swapEndiannesInt(three);

fwrite(&tmp\_int, sizeof(int), 1, fp);

tmp\_int = swapEndiannesInt(cellsToNodes\_data[i\*N\_NODESPERCELL ]);

fwrite(&tmp\_int, sizeof(int), 1, fp);

tmp\_int = swapEndiannesInt(cellsToNodes\_data[i\*N\_NODESPERCELL+1]);

fwrite(&tmp\_int, sizeof(int), 1, fp);

tmp\_int = swapEndiannesInt(cellsToNodes\_data[i\*N\_NODESPERCELL+2]);

fwrite(&tmp\_int, sizeof(int), 1, fp);

tmp\_int = swapEndiannesInt(cellsToNodes\_data[i\*N\_NODESPERCELL+3]);

fwrite(&tmp\_int, sizeof(int), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

// write cell types (5 for triangles)

sprintf(s, "CELL\_TYPES %d\n", ncell); fwrite(s, sizeof(char), strlen(s), fp);

int five=9; //five triangles 9 quads

for ( i=0; i<ncell; ++i ) {

tmp\_int = swapEndiannesInt(five);

fwrite(&tmp\_int, sizeof(int), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

//double\* values\_data;

//values\_data = (double\*) values->data;

sprintf(s, "CELL\_DATA %d\n"

"SCALARS q0 double 1\n"

"LOOKUP\_TABLE default\n",

ncell); fwrite(s, sizeof(char), strlen(s), fp);

for ( i=0; i<ncell; ++i ) {

tmp\_double = swapEndiannesDouble(values\_data[i\*N\_STATEVAR]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

strcpy(s, "SCALARS q1 double 1\nLOOKUP\_TABLE default\n"); fwrite(s, sizeof(char), strlen(s), fp);

for ( i=0; i<ncell; ++i ){

tmp\_double = swapEndiannesDouble(values\_data[i\*N\_STATEVAR+1]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

strcpy(s, "SCALARS q2 double 1\nLOOKUP\_TABLE default\n"); fwrite(s, sizeof(char), strlen(s), fp);

for ( i=0; i<ncell; ++i ) {

tmp\_double = swapEndiannesDouble(values\_data[i\*N\_STATEVAR+2]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

strcpy(s, "SCALARS q3 double 1\nLOOKUP\_TABLE default\n"); fwrite(s, sizeof(char), strlen(s), fp);

for ( i=0; i<ncell; ++i ) {

tmp\_double = swapEndiannesDouble(values\_data[i\*N\_STATEVAR+3]);

fwrite(&tmp\_double, sizeof(double), 1, fp);

}

strcpy(s, "\n"); fwrite(s, sizeof(char), strlen(s), fp);

if(fclose(fp) != 0) {

printf("can't close file %s\n",filename);

exit(-1);

}

}

2. WriteMeshToVTKAscii()

将mesh结构写到vtk ASCII文件中.

/\*

\* Write simulation output to ASCII file

\*/

//void WriteMeshToVTKAscii(const char\* filename, op\_dat nodeCoords, int nnode, op\_map cellsToNodes, int ncell, op\_dat values) {

void WriteMeshToVTKAscii(const char\* filename, double\* nodeCoords\_data, int nnode, int\* cellsToNodes\_data, int ncell, double \*values\_data) {

printf("Writing OutputSimulation to ASCII file: %s \n",filename);

FILE\* fp;

fp = fopen(filename, "w");

if(fp == NULL) {

printf("can't open file for write %s\n",filename);

exit(-1);

}

// write header

fprintf(fp,"# vtk DataFile Version 2.0\n Output from OP2 Volna.\n");

fprintf(fp,"ASCII \nDATASET UNSTRUCTURED\_GRID\n\n");

// write vertices

fprintf(fp,"POINTS %d double\n", nnode);

// double\* nodeCoords\_data;

// nodeCoords\_data = (double\*)nodeCoords->data;

int i = 0;

for (i = 0; i < nnode; ++i) {

fprintf(fp, "%g %g %g \n",

(double)nodeCoords\_data[i\*MESH\_DIM ],

(double)nodeCoords\_data[i\*MESH\_DIM+1],

0.0);

}

fprintf(fp, "\n");

fprintf(fp, "CELLS %d %d\n", ncell, 5\*ncell);

for ( i = 0; i < ncell; ++i ) {

fprintf(fp, "4 %d %d %d %d\n",

cellsToNodes\_data[i\*N\_NODESPERCELL ],

cellsToNodes\_data[i\*N\_NODESPERCELL+1],

cellsToNodes\_data[i\*N\_NODESPERCELL+2],

cellsToNodes\_data[i\*N\_NODESPERCELL+3]);

}

fprintf(fp, "\n");

// write cell types (5 for triangles, 9 for quads)

fprintf(fp, "CELL\_TYPES %d\n", ncell);

for ( i=0; i<ncell; ++i )

fprintf(fp, "9\n");

fprintf(fp, "\n");

// double\* values\_data;

// values\_data = (double\*) values->data;

fprintf(fp, "CELL\_DATA %d\n"

"SCALARS q0 double 1\n"

"LOOKUP\_TABLE default\n",

ncell);

double tmp = 0.0;

for ( i=0; i<ncell; ++i ) {

tmp = values\_data[i\*N\_STATEVAR];

fprintf(fp, "%10.20g\n", values\_data[i\*N\_STATEVAR]);

}

fprintf(fp, "\n");

fprintf(fp, "SCALARS q1 double 1\n"

"LOOKUP\_TABLE default\n");

for ( i=0; i<ncell; ++i )

fprintf(fp, "%10.20g\n", values\_data[i\*N\_STATEVAR+1]);

fprintf(fp, "\n");

fprintf(fp, "SCALARS q2 double 1\n"

"LOOKUP\_TABLE default\n");

for ( i=0; i<ncell; ++i )

fprintf(fp, "%10.20g\n", values\_data[i\*N\_STATEVAR+2]);

fprintf(fp, "\n");

fprintf(fp, "SCALARS q3 double 1\n"

"LOOKUP\_TABLE default\n");

for ( i=0; i<ncell; ++i )

fprintf(fp, "%10.20g\n", values\_data[i\*N\_STATEVAR+3]);

fprintf(fp, "\n");

if(fclose(fp) != 0) {

printf("can't close file %s\n",filename);

exit(-1);

}

}

1. What is the whole user algorithm of this mesh application? Write out the algorithm as a C function.

（杨宗恺）

根据文档，mesh application的user algorithm有：

1.设置 Set/Arg

2.使用1到20个参数在网格应用上使用一些用户自定义的算法进行循环

3.输入输出网格

4.有限元计算

分析老师的代码，将使用到用户算法改写成函数，有：

//用户可以通过设置文件路径名来读取网格文件，老师的代码中，pathname为"./new\_grid.dat"

FILE \*fp;

void readGrid(FILE \*fp, char \*pathname) {

if ( (fp = fopen(pathname,"r")) == NULL) {

printf("can't open file new\_grid.dat\n"); exit(-1);

}

}

//设置变量来对图进行染色

void setColoring(real \*x, real \*q, real \*qold, real \*res, real \*adt) {

int\* cell2edge = (int \*) malloc(4\*ncell\*sizeof(int));

for(int i = 0; i< 4\*ncell; ++i){

cell2edge[i] = -1;

}

int\* edge2color = (int\*) malloc(nedge\*sizeof(int));

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

edge2color[i] = -1;

}

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

if (fscanf(fp,"%lf %lf \n",&x[2\*n], &x[2\*n+1]) != 2) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

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

if (fscanf(fp,"%d %d %d %d \n",&cell[4\*n ], &cell[4\*n+1],

&cell[4\*n+2], &cell[4\*n+3]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

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

if (fscanf(fp,"%d %d %d %d \n",&edge[2\*n], &edge[2\*n+1],

&ecell[2\*n],&ecell[2\*n+1]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

for(int i = 0; i < 4;++i){

if(cell2edge[ecell[2\*n]+i] == -1){

cell2edge[ecell[2\*n]+i] = n;

break;

}

}

for(int i = 0; i < 4;++i){

if(cell2edge[ecell[2\*n+1]+i] == -1){

cell2edge[ecell[2\*n+1]+i] = n;

break;

}

}

}

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

if (fscanf(fp,"%d %d %d %d \n",&bedge[2\*n],&bedge[2\*n+1],

&becell[n], &bound[n]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

fclose(fp);

int max = 0;

std::vector<std::vector<int> > color2edge;

for(int edge\_ind = 0; edge\_ind < nedge; ++edge\_ind){

int color = 0;

while(1){

bool valid\_color = true;

for(int i = 0; i < 4; ++i){

if(edge2color[cell2edge[ecell[edge\_ind]+i]] == color ||

edge2color[cell2edge[ecell[edge\_ind+1]+i]] == color){

valid\_color = false;

}

}

if(valid\_color){

edge2color[edge\_ind] = color;

if(color2edge.size() == color){

color2edge.push\_back(std::vector<int>(1,edge\_ind));

} else if(color < color2edge.size()){

color2edge[color].push\_back(edge\_ind);

} else {

printf("ismet para van\n");

}

if(color > max){

max = color;

printf("%d\n",max);

}

break;

}

++color;

}

}

printf("%d %d %d %d %d %d", color2edge[0].size(),color2edge[1].size(),color2edge[2].size(),

color2edge[3].size(),color2edge[4].size(),color2edge[5].size());

free(cell2edge);

free(edge2color);

}

//设置常数并初始化流场和残差

void setConInitFFRes(real gam, real gm1, real cfl, real eps， read \*qinf, real \*q, read \*res) {

real mach = 0.4f;

real alpha = 3.0f\*atan(1.0f)/45.0f;

real p = 1.0f;

real r = 1.0f;

real u = sqrt(gam\*p/r)\*mach;

real e = p/(r\*gm1) + 0.5f\*u\*u;

qinf[0] = r;

qinf[1] = r\*u;

qinf[2] = 0.0f;

qinf[3] = r\*e;

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

for (int m=0; m<4; m++) {

q[4\*n+m] = qinf[m];

res[4\*n+m] = 0.0f;

}

}

}

1. In this mesh application, the computed mesh finally is stored in a VTK format text file or a binary file. Describe in some detail what is the storage structure of the VTK text format of mesh?

（胡洁珏）

vtk文件可分为六个部分：

1.数据版本声明：

​ # vtk DataFile Version 3.0

​ 这句话是强制的，说明文件的版本，这里说明使用的版本是 vtk 的3.0版本；

2.标题：

​ 这个标题是自定义的，最多256个字符，以回车作为结束；

3.文件格式声明：

​ 文件格式分为两种，一种是ASCII,另一种是BINARY，表示这份vtk是使用标准ASCII码字符集还是BINARU

4.几何拓扑结构：

​ format表示数据的格式，包括：

STRUCTURED\_POINTS

STRUCTURED\_GRID

UNSTRUCTURED\_GRID

POLYDATA

STRUCTURED\_POINTS

RECTILINEAR\_GRID

FIELD

当选择的format是UNSTRUCTURED\_GRID时，需要指定节点和单元：

POINTS [n] [dataType] (#n\_point#域填入实际的点数, dataType为double float等类型)

[x\_0] [y\_0] [z\_0]

[x\_1] [y\_1] [z\_1]

[x\_0] [y\_0] [z\_0]

接着单元的声明：

CELLS [n\_cells] [n\_list] #n\_cells填入单元个数，n\_list填入列表的数字的总数)

[单元0顶点数量] [单元0顶点0的编号] [单元0顶点1的编号] [单元0顶点2的编号] ...

[单元1顶点数量] [单元1顶点0的编号] [单元1顶点1的编号] [单元1顶点2的编号] ...

.

.

.

CELL\_TYPES [n\_cells] #n\_cells与前面CELLS那一块的n\_cells必须一样

[单元0类型] #表示几何类型的整数，参考vtk的文档，比如四面体是10，六面体是11

[单元1类型]

.

.

.

5.物理量：

​ 分为POINT\_DATA 和 CELL\_DATA,表示节点上的值和单元上的值。可以表示多个物理量：标量（scalar)，向量(vector)，张量(tensor)。如下所示：

POINT\_DATA [n]

[数据]

CELL\_DATA [n]

[数据]

数据可以是SCALAR，VECTOR或TENSOR。

SCALARS [dataName] [dataType] [numComp]

LOOKUP\_TABLE [tableName]

s\_0

s\_1

...

s\_n-1

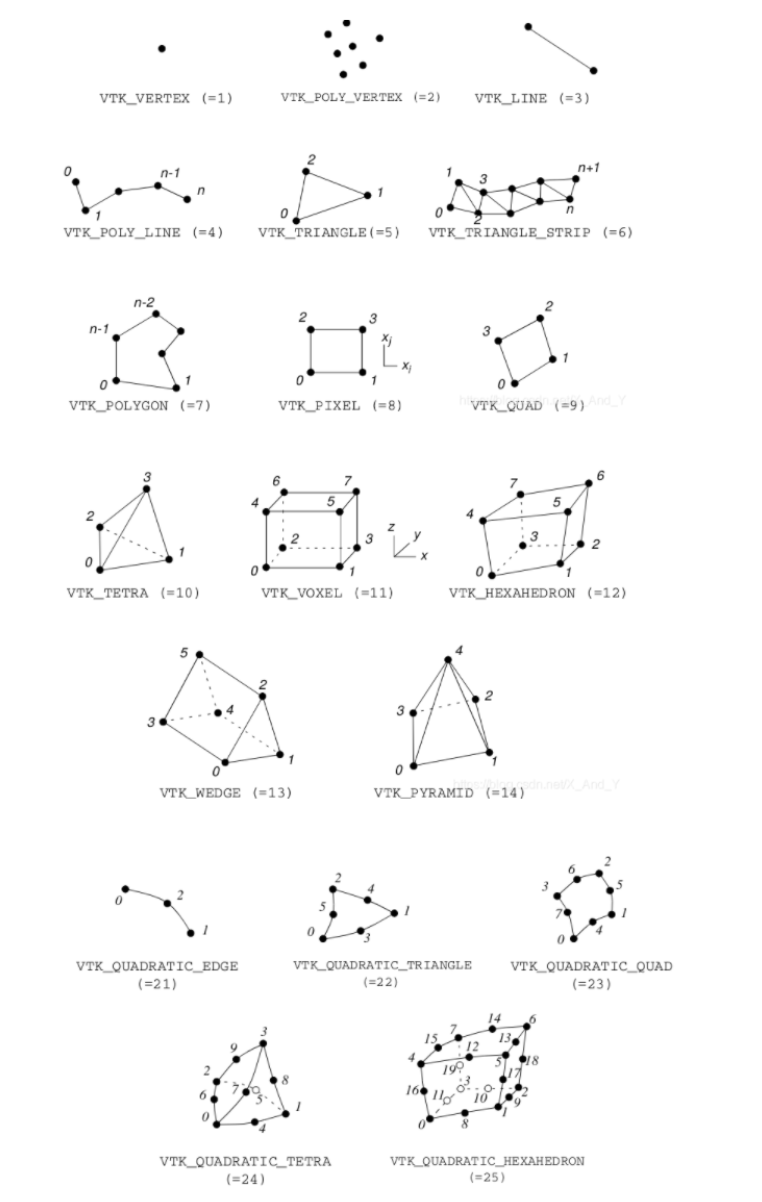
dataName：数据名字

dataType：double之类的

numComp：一般是1

tableName：表名

6.单元类型&&节点顺序：



单元类型为括号中的数字

1. In this mesh application, there is a fragment of code doing coloring over the mesh. Try to understand what this coloring does, describe in a paragraph to explain your understanding about this coloring algorithm.

（莫振威）

首先对需要着色的网格进行变量赋值，具体情况如下所示

//cell2edge是记录单元格cell与邻接边edge的数组，此处进行初始化

int\* cell2edge = (int \*) malloc(4\*ncell\*sizeof(int));

for(int i = 0; i< 4\*ncell; ++i){

cell2edge[i] = -1;

}

//edge2color是记录网格边edge着色情况的数组，此处进行初始化

int\* edge2color = (int\*) malloc(nedge\*sizeof(int));

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

edge2color[i] = -1;

}

//对网格节点进行赋值，数组x[]每两个元素记录一个节点的坐标属性（x，y）

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

if (fscanf(fp,"%lf %lf \n",&x[2\*n], &x[2\*n+1]) != 2) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

//对单元格cell进行赋值，数组cell[]中的元素代表一个节点下标，四个节点组成一个单元格

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

if (fscanf(fp,"%d %d %d %d \n",&cell[4\*n ], &cell[4\*n+1],

&cell[4\*n+2], &cell[4\*n+3]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

//对非处在结界bound的边进行赋值，数组edge[]记录与边邻接的两个节点，数组ecell[]记录与边邻接的单元格cell

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

if (fscanf(fp,"%d %d %d %d \n",&edge[2\*n], &edge[2\*n+1],

&ecell[2\*n],&ecell[2\*n+1]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

//对处于相同单元格的边进行初始化赋值

for(int i = 0; i < 4;++i){

if(cell2edge[ecell[2\*n]+i] == -1){

cell2edge[ecell[2\*n]+i] = n;

break;

}

}

for(int i = 0; i < 4;++i){

if(cell2edge[ecell[2\*n+1]+i] == -1){

cell2edge[ecell[2\*n+1]+i] = n;

break;

}

}

}

//对处于结界bound的边进行赋值，数组bedge[]记录与边邻接的两个节点，数组becell[]记录与边邻接的单元格，数组bound[]记录与边邻接的边界

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

if (fscanf(fp,"%d %d %d %d \n",&bedge[2\*n],&bedge[2\*n+1],

&becell[n], &bound[n]) != 4) {

printf("error reading from new\_grid.dat\n"); exit(-1);

}

}

fclose(fp);

对相关变量初始化后，进行网格着色，具体着色过程如下所示

int max = 0; //记录当前所用颜色的最大数量

std::vector<std::vector<int> > color2edge; //记录已着色的网格边

//遍历所有边，对边进行着色处理

for(int edge\_ind = 0; edge\_ind < nedge; ++edge\_ind){

int color = 0; //将颜色初始化

while(1){

bool valid\_color = true; //判断颜色是否合适的标志变量

for(int i = 0; i < 4; ++i){

//如果边所邻接的单元格有相同的颜色，则颜色不合适，重新选择新的颜色进行着色

if(edge2color[cell2edge[ecell[edge\_ind]+i]] == color ||

edge2color[cell2edge[ecell[edge\_ind+1]+i]] == color){

valid\_color = false;

}

}

if(valid\_color){

//如果颜色合适，则对该边进行着色

edge2color[edge\_ind] = color;

if(color2edge.size() == color){

//如果当前已使用颜色数量与当前颜色值color相同，说明还未使用过该种颜色，那么就将当前颜色加入

color2edge.push\_back(std::vector<int>(1,edge\_ind));

} else if(color < color2edge.size()){

//如果当前颜色值color比已使用颜色数量小，说明当前使用颜色值color在已使用颜色中，就将着色的边加入该颜色中

color2edge[color].push\_back(edge\_ind);

} else {

printf("ismet para van\n");

}

if(color > max){

//max记录当前所使用的最大颜色值，并输出到控制台

max = color;

printf("%d\n",max);

}

break;

}

++color; //使用新的颜色进行下一轮着色

}

}

着色所使用的是遍历算法，通过对网格中的所有边进行遍历，逐一判断着色条件，即相邻单元格是否存在相同颜色的边，来排除重复着色的情况，并选择性地对网格中的边进行着色，直至遍历完所有边结束算法。

1. Rewrite the source code in the folder Airfoil using your own functions you have done above.

见附件

Subimit your work with all source code and documents you have worked out.