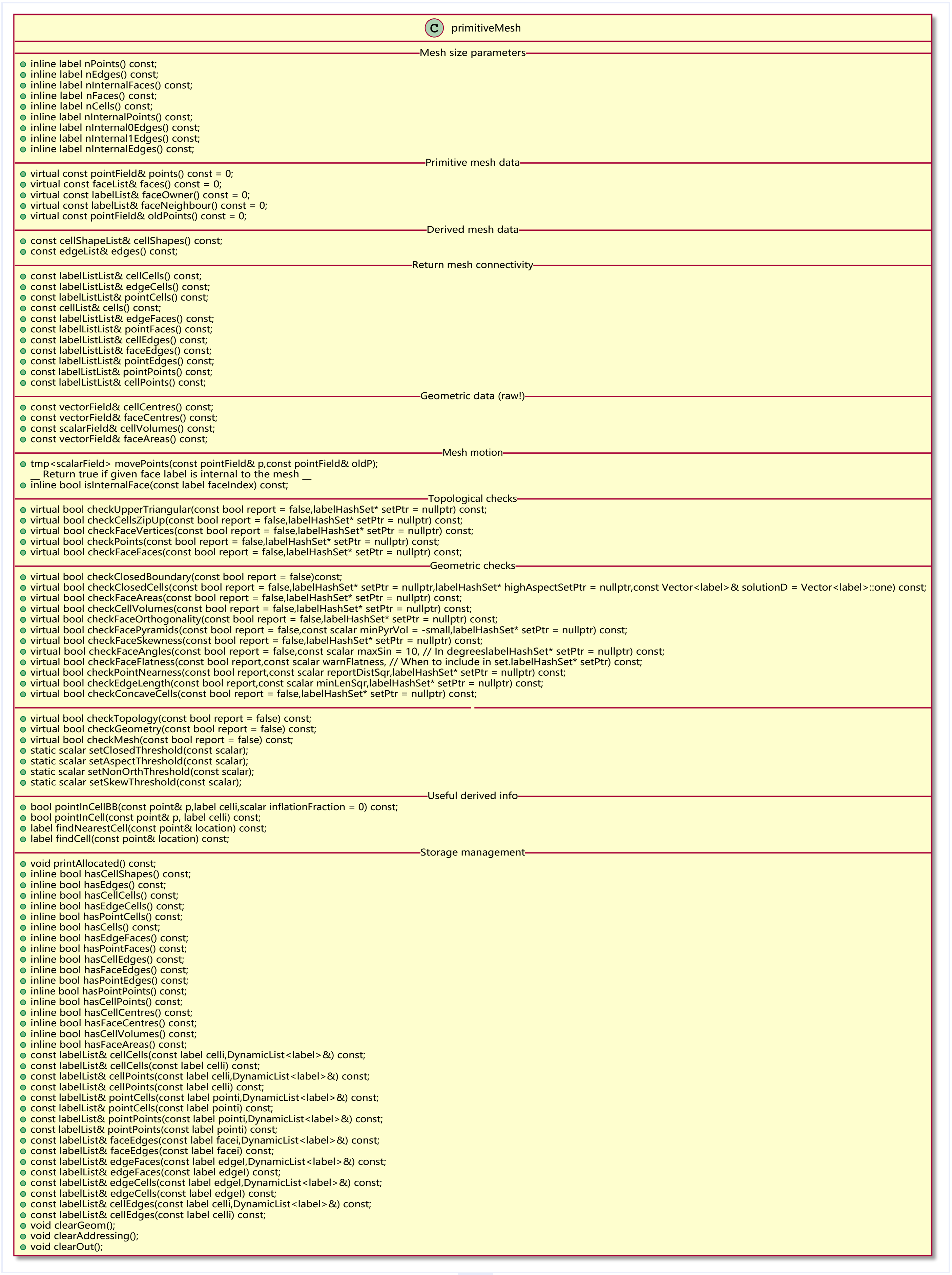
无摘要



```
primitiveMesh
--Mesh size parameters--
inline label nPoints() const;
inline label nEdges() const;
inline label nInternalFaces() const;
inline label nFaces() const;
inline label nCells() const;
inline label nInternalPoints() const;
inline label nInternal0Edges() const;
inline label nInternal1Edges() const;
inline label nInternalEdges() const;
--Primitive mesh data--
virtual const pointField& points() const = 0;
virtual const faceList& faces() const = 0;
virtual const labelList& faceOwner() const = 0;
virtual const labelList& faceNeighbour() const = 0;
virtual const pointField& oldPoints() const = 0;
--Derived mesh data--
const cellShapeList& cellShapes() const;
const edgeList& edges() const;
--Return mesh connectivity--
const labelListList& cellCells() const;
const labelListList& edgeCells() const;
const labelListList& pointCells() const;
const cellList& cells() const;
const labelListList& edgeFaces() const;
const labelListList& pointFaces() const;
const labelListList& cellEdges() const;
const labelListList& faceEdges() const;
const labelListList& pointEdges() const;
const labelListList& pointPoints() const;
const labelListList& cellPoints() const;
--Geometric data (raw!)--
const vectorField& cellCentres() const;
const vectorField& faceCentres() const;
const scalarField& cellVolumes() const;
const vectorField& faceAreas() const;
--Mesh motion--
tmp<scalarField> movePoints(const pointField& p,const pointField& oldP);
__ Return true if given face label is internal to the mesh __
inline bool isInternalFace(const label faceIndex) const;
--Topological checks--
virtual bool checkUpperTriangular(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkCellsZipUp(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkFaceVertices(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkPoints(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkFaceFaces(const bool report = false,labelHashSet* setPtr = nullptr) const;
--Geometric checks--
virtual bool checkClosedBoundary(const bool report = false)const;
virtual bool checkClosedCells(const bool report = false,labelHashSet* setPtr = nullptr,labelHashSet* highAspectSetPtr = nullptr,const Vector<label>& solutionD = Vector<label>::one) const;
virtual bool checkFaceAreas(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkCellVolumes(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkFaceOrthogonality(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkFacePyramids(const bool report = false,const scalar minPyrVol = -small,labelHashSet* setPtr = nullptr) const;
virtual bool checkFaceSkewness(const bool report = false,labelHashSet* setPtr = nullptr) const;
virtual bool checkFaceAngles(const bool report = false,const scalar maxSin = 10, // In degreeslabelHashSet* setPtr = nullptr) const;
virtual bool checkFaceFlatness(const bool report,const scalar warnFlatness, // When to include in set.labelHashSet* setPtr) const;
virtual bool checkPointNearness(const bool report,const scalar reportDistSqr,labelHashSet* setPtr = nullptr) const;
virtual bool checkEdgeLength(const bool report,const scalar minLenSqr,labelHashSet* setPtr = nullptr) const;
virtual bool checkConcaveCells(const bool report = false,labelHashSet* setPtr = nullptr) const;
----
virtual bool checkTopology(const bool report = false) const;
virtual bool checkGeometry(const bool report = false) const;
virtual bool checkMesh(const bool report = false) const;
static scalar setClosedThreshold(const scalar);
static scalar setAspectThreshold(const scalar);
static scalar setNonOrthThreshold(const scalar);
static scalar setSkewThreshold(const scalar);
--Useful derived info--
bool pointInCellBB(const point& p,label celli,scalar inflationFraction = 0) const;
bool pointInCell(const point& p, label celli) const;
label findNearestCell(const point& location) const;
label findCell(const point& location) const;
--Storage management--
void printAllocated() const;
inline bool hasCellShapes() const;
inline bool hasEdges() const;
inline bool hasCellCells() const;
inline bool hasEdgeCells() const;
inline bool hasPointCells() const;
inline bool hasCells() const;
inline bool hasEdgeFaces() const;
inline bool hasPointFaces() const;
inline bool hasCellEdges() const;
inline bool hasFaceEdges() const;
inline bool hasPointEdges() const;
inline bool hasPointPoints() const;
inline bool hasCellPoints() const;
inline bool hasCellCentres() const;
inline bool hasFaceCentres() const;
inline bool hasCellVolumes() const;
inline bool hasFaceAreas() const;
const labelList& cellCells(const label celli,DynamicList<label>&) const;
const labelList& cellCells(const label celli) const;
const labelList& cellPoints(const label celli,DynamicList<label>&) const;
const labelList& cellPoints(const label celli) const;
const labelList& pointCells(const label pointi,DynamicList<label>&) const;
const labelList& pointCells(const label pointi) const;
const labelList& pointPoints(const label pointi,DynamicList<label>&) const;
const labelList& pointPoints(const label pointi) const;
const labelList& faceEdges(const label facei,DynamicList<label>&) const;
const labelList& faceEdges(const label facei) const;
const labelList& edgeFaces(const label edgeI,DynamicList<label>&) const;
const labelList& edgeFaces(const label edgeI) const;
const labelList& edgeCells(const label edgeI,DynamicList<label>&) const;
const labelList& edgeCells(const label edgeI) const;
const labelList& cellEdges(const label celli,DynamicList<label>&) const;
const labelList& cellEdges(const label celli) const;
void clearGeom();
void clearAddressing();
void clearOut();
```

primitiveMesh没有更顶层的类了。常用的方法：

```c++
inline label nCells() const; //返回网格个数
const cellList& cells() const; //返回所有的 cell, cell 是由 faces 构成的
const labelList& cellCells(const label celli) const; //返回 celli 的 邻居们
const labelList& edgeCells(const label edgeI) const; //返回共用这条边的所有网格
const labelList& pointCells(const label pointi) const; //返回共用这个点的所有网格
```

```c++
Info<<"mesh.cells()[0]==="<<mesh.cells()[0]<<endl;
```

返回包围第0个cell的所有面的index：

```c++
mesh.cells()[0]===6(0 1 2 4379997 4382913 4391661)
```

surfaceInterpolation没有更顶层的类了。

lduMesh没有更顶层的类了。

fvMesh 常用的方法：

```c++
//- Return cell volumes
const DimensionedField<scalar, volMesh>& V() const;
//- Return old-time cell volumes
const DimensionedField<scalar, volMesh>& V0() const;
//- Return old-old-time cell volumes
const DimensionedField<scalar, volMesh>& V00() const;
//- Return sub-cycle cell volumes
tmp<DimensionedField<scalar, volMesh>> Vsc() const;
//- Return sub-cycl old-time cell volumes
tmp<DimensionedField<scalar, volMesh>> Vsc0() const;
//- Return cell face area vectors
const surfaceVectorField& Sf() const;
//- Return cell face area magnitudes
const surfaceScalarField& magSf() const;
//- Return cell face motion fluxes
const surfaceScalarField& phi() const;
//- Return cell centres as volVectorField
const volVectorField& C() const;
//- Return face centres as surfaceVectorField
const surfaceVectorField& Cf() const;
```

```c++
//- Internal face owner
const labelUList& owner() const
{
    return lduAddr().lowerAddr();
}
```

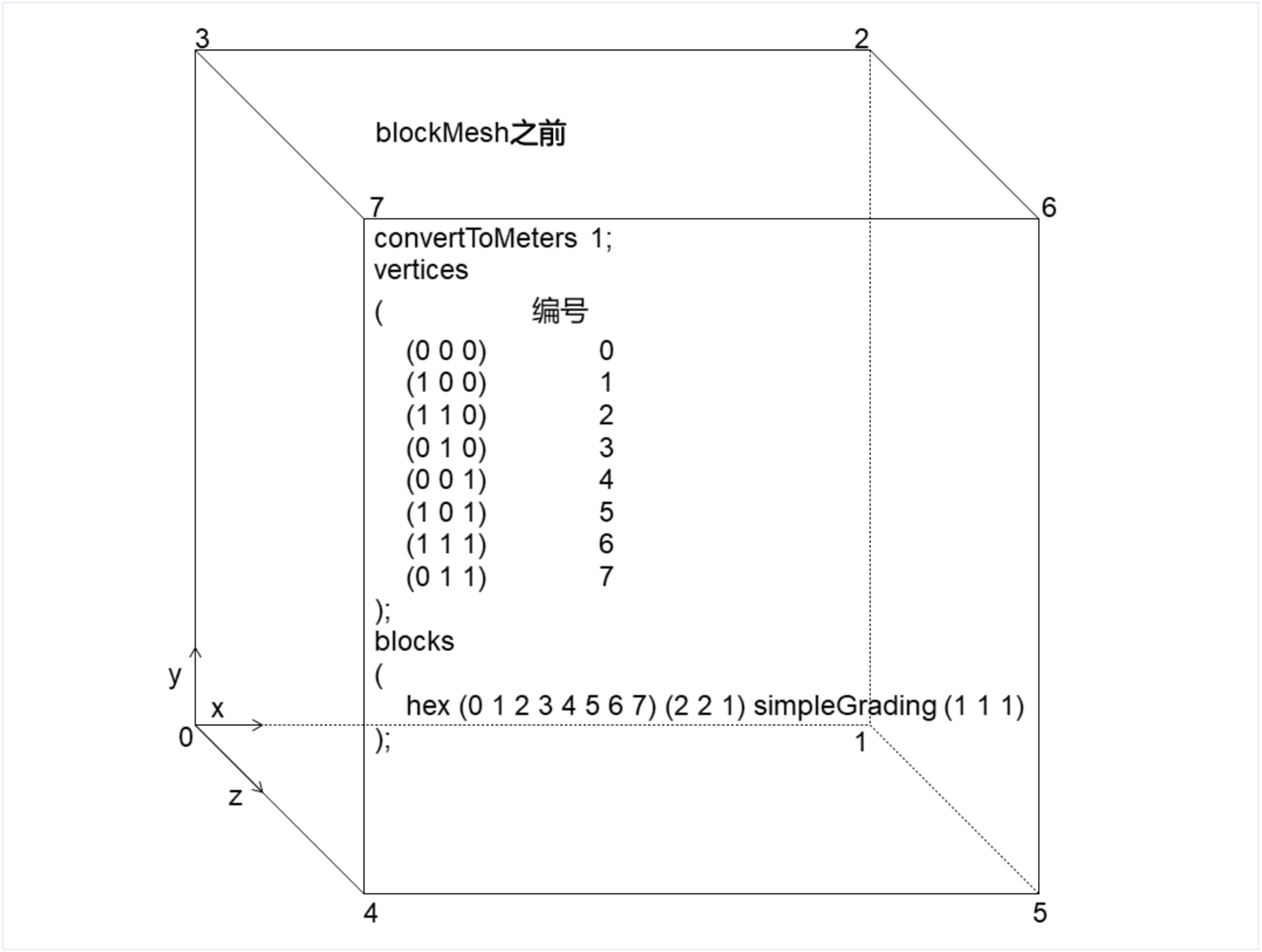
```
//- Internal face neighbour
const labelUList& neighbour() const
{
    return lduAddr().upperAddr();
}
```

画了一个 2*2*1 的网格：

plain

```
Mesh Information
----------------
  nPoints: 18
  nCells: 4
  nFaces: 20
  nInternalFaces: 4
```

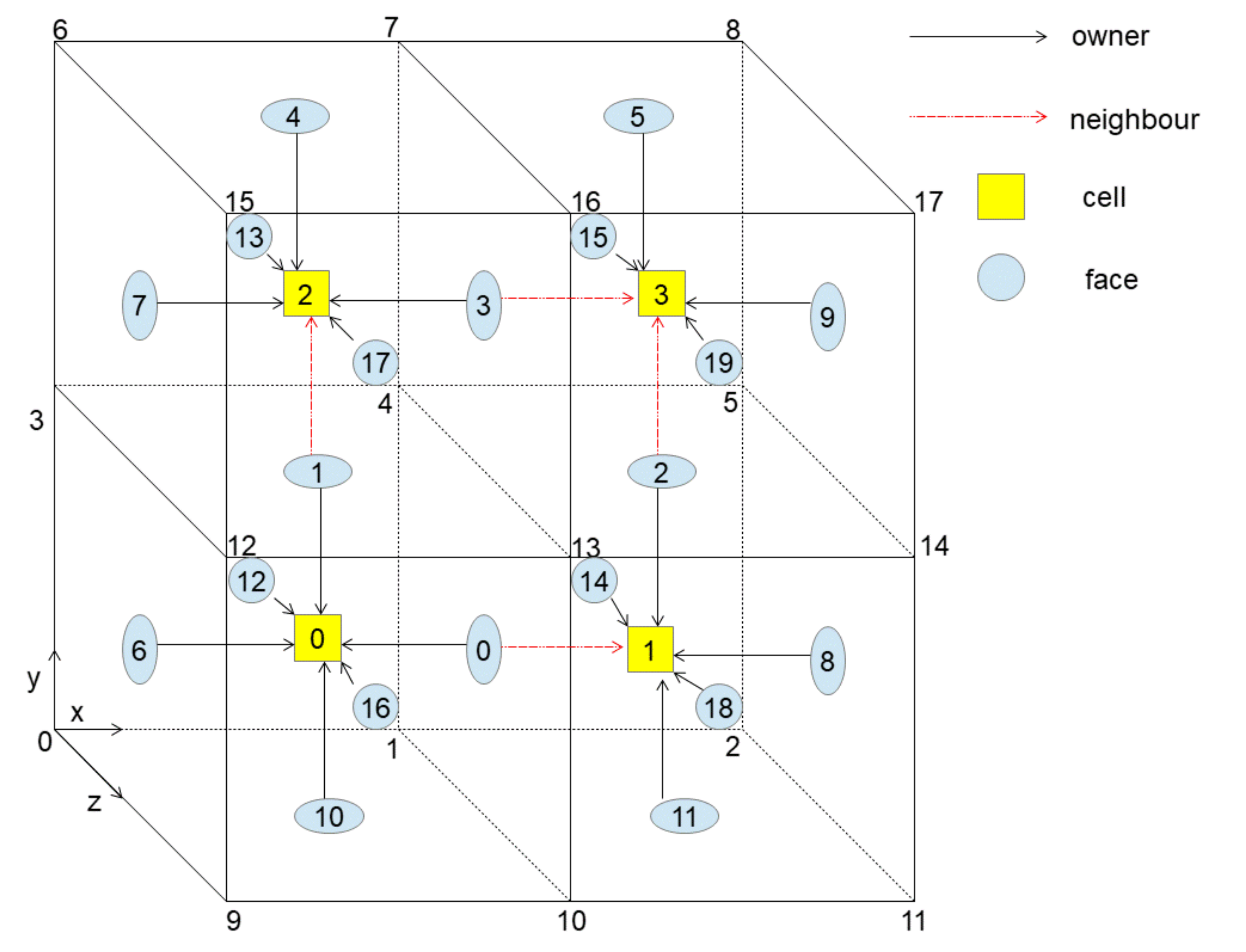
以下三张图为李建治绘制：



```
convertToMeters 1;
vertices
(                 编号
    (0 0 0)          0
    (1 0 0)          1
    (1 1 0)          2
    (0 1 0)          3
    (0 0 1)          4
    (1 0 1)          5
    (1 1 1)          6
    (0 1 1)          7
);
blocks
(
    hex (0 1 2 3 4 5 6 7) (2 2 1) simpleGrading (1 1 1)
);
```

blockMesh之后

```
points                  faces                        owner        neighbour
18                      20                           20           4
(               编号    (                    编号    (            (
(0 0 0)           0     4(1 4 13 10)          0      0            1
(0.5 0 0)         1     4(3 12 13 4)          1      0            2
(1 0 0)           2     4(4 13 14 5)          2      1            3
(0 0.5 0)         3     4(4 7 16 13)          3      2            3
(0.5 0.5 0)       4     4(6 15 16 7)          4      2            )
(1 0.5 0)         5     4(7 16 17 8)          5      3
(0 1 0)           6     4(0 9 12 3)           6      0
(0.5 1 0)         7     4(3 12 15 6)          7      2
(1 1 0)           8     4(2 5 14 11)          8      1
(0 0 1)           9     4(5 8 17 14)          9      3
(0.5 0 1)        10     4(0 1 10 9)          10      0
(1 0 1)          11     4(1 2 11 10)         11      1
(0 0.5 1)        12     4(0 3 4 1)           12      0
(0.5 0.5 1)      13     4(3 6 7 4)           13      2
(1 0.5 1)        14     4(1 4 5 2)           14      1
(0 1 1)          15     4(4 7 8 5)           15      3
(0.5 1 1)        16     4(9 10 13 12)        16      0
(1 1 1)          17     4(12 13 16 15)       17      2
)                       4(10 11 14 13)       18      1
                        4(13 14 17 16)       19      3
                        )                            )
```



下面这段代码来自 *CFD中文网*

c++

```c++
const labelUList& owner = mesh.owner();
//这里的owner只考虑了内部面?
Info<<"owner==="<<owner<<endl;
const labelUList& neighbour = mesh.neighbour();
Info<<"neighbour==="<<neighbour<<endl;

forAll(T,cellI)
{
Info<<"cellI==="<<cellI<<endl;
    forAll(mesh.cells()[cellI], faceI)
    //遍历当前 cellI 的所有面
    {
        if (mesh.isInternalFace(mesh.cells()[cellI][faceI]))
        //如果这个cellI由6个面包围，则faceI从0到5遍历
        {
            const label faceIndex = mesh.cells()[cellI][faceI];
            //faceIndex 是当前 faceI 真实 index
            Info<< " one of faceIndex is " << faceIndex;
            Info<< ", and this face's owner is "
                << owner[faceIndex] << ", its neighbour is "
                << neighbour[faceIndex] << nl;
        }
    }
}
```

plain

```
owner===4(0 0 1 2)
这里内部面只有 0 1 2 3，对应的owner是 0 0 1 2
neighbour===4(1 2 3 3)



cellI===0
 one of faceIndex is 0, and this face's owner is 0, its neighbour is 1
 one of faceIndex is 1, and this face's owner is 0, its neighbour is 2
cellI===1
 one of faceIndex is 2, and this face's owner is 1, its neighbour is 3
 one of faceIndex is 0, and this face's owner is 0, its neighbour is 1
cellI===2
 one of faceIndex is 3, and this face's owner is 2, its neighbour is 3
 one of faceIndex is 1, and this face's owner is 0, its neighbour is 2
cellI===3
 one of faceIndex is 2, and this face's owner is 1, its neighbour is 3
 one of faceIndex is 3, and this face's owner is 2, its neighbour is 3
```

c++

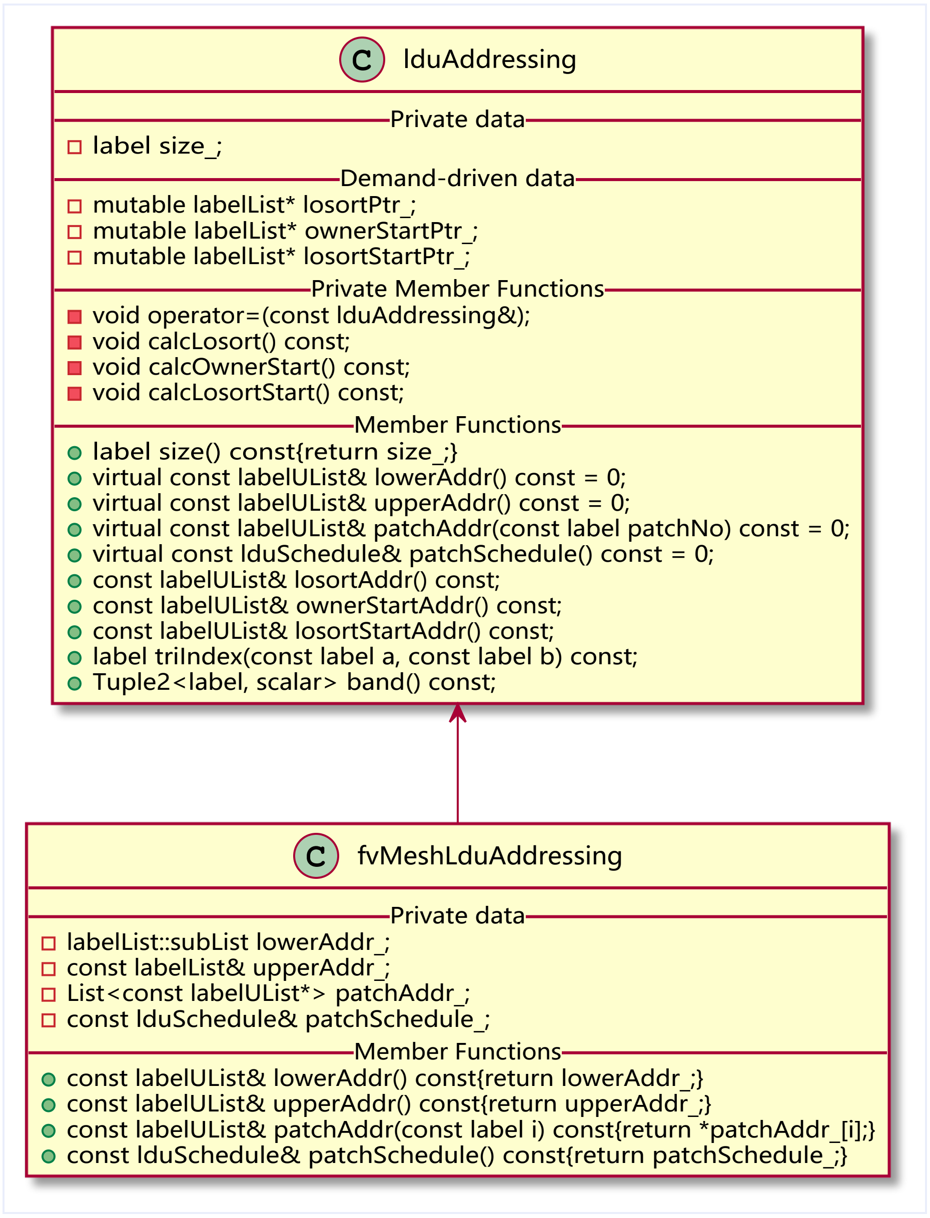
```c++
//- Internal face owner
const labelUList& owner() const
{
    return lduAddr().lowerAddr();
}
```

c++

```c++
fvMesh::lduAddr(){return *lduPtr_;}
```

c++

```c++
mutable fvMeshLduAddressing* lduPtr_;
```


C lduAddressing
Private data
□ label size_;
Demand-driven data
□ mutable labelList* losortPtr_;
□ mutable labelList* ownerStartPtr_;
□ mutable labelList* losortStartPtr_;
Private Member Functions
■ void operator=(const lduAddressing&);
■ void calcLosort() const;
■ void calcOwnerStart() const;
■ void calcLosortStart() const;
Member Functions
● label size() const{return size_;}
● virtual const labelUList& lowerAddr() const = 0;
● virtual const labelUList& upperAddr() const = 0;
● virtual const labelUList& patchAddr(const label patchNo) const = 0;
● virtual const lduSchedule& patchSchedule() const = 0;
● const labelUList& losortAddr() const;
● const labelUList& ownerStartAddr() const;
● const labelUList& losortStartAddr() const;
● label triIndex(const label a, const label b) const;
● Tuple2<label, scalar> band() const;
C fvMeshLduAddressing
Private data
□ labelList::subList lowerAddr_;
□ const labelList& upperAddr_;
□ List<const labelUList*> patchAddr_;
□ const lduSchedule& patchSchedule_;
Member Functions
● const labelUList& lowerAddr() const{return lowerAddr_;}
● const labelUList& upperAddr() const{return upperAddr_;}
● const labelUList& patchAddr(const label i) const{return *patchAddr_[i];}
● const lduSchedule& patchSchedule() const{return patchSchedule_;}