Milestone 2

Berechnung eines passiven Skalars mit Hilfe der Finiten Volumen Methode (FVM)

(Instationär, Konvektion, Diffusion, Strömungsfeld bekannt)

Bilanzgleichungen

- Nicht konservativ

$$\rho\frac{\partial\phi}{\partial t}+\rho\vec{v}\cdot\vec{\nabla}\phi=\vec{\nabla}\cdot(a\vec{\nabla}\phi)+s$$

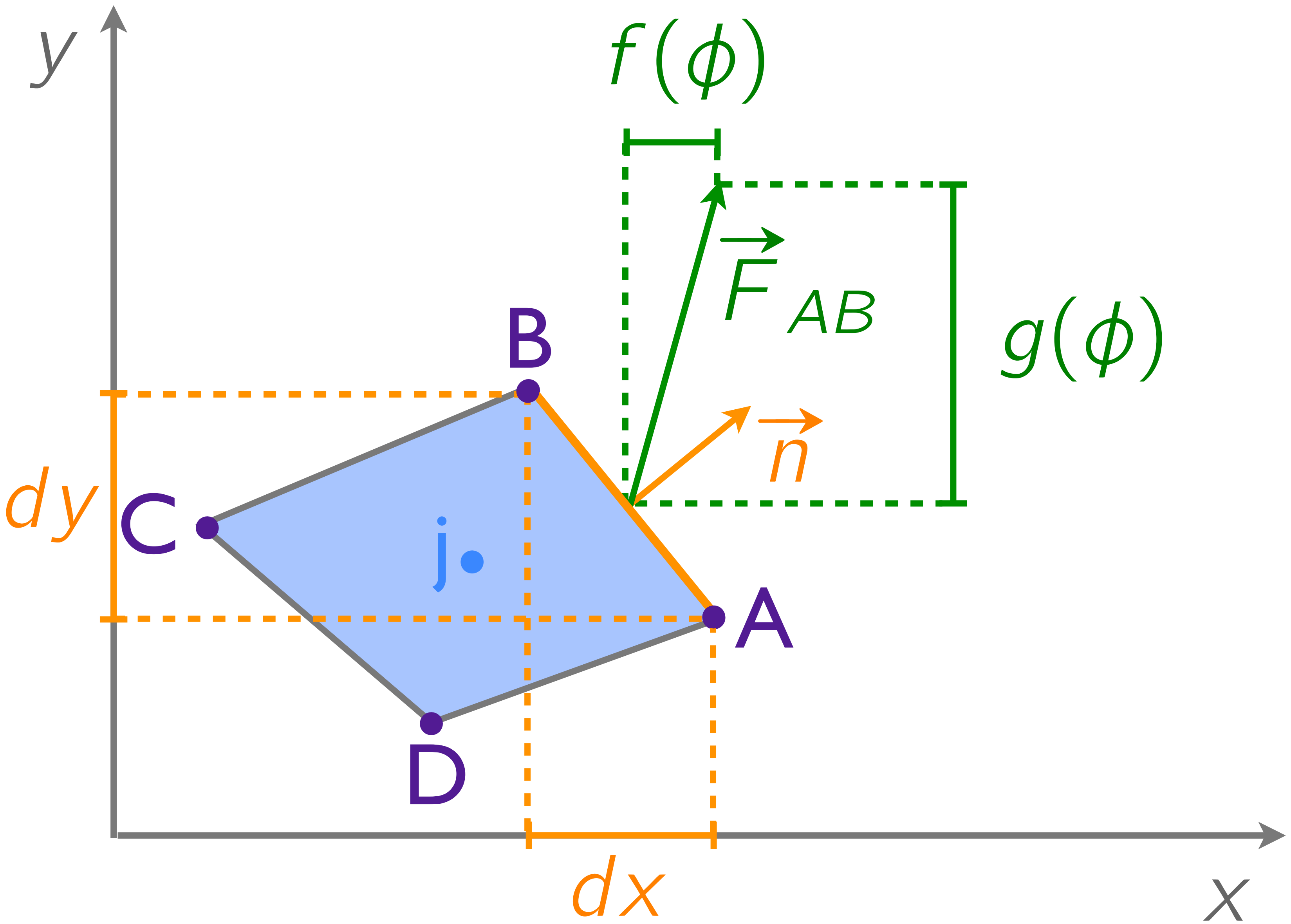
- Konservativ

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\phi dV+\oint_{\partial\Omega}\vec{F}(\phi)\cdot\overrightarrow{dA}=\int_{\Omega}s dV$$

- Für finites Volumen *j*:

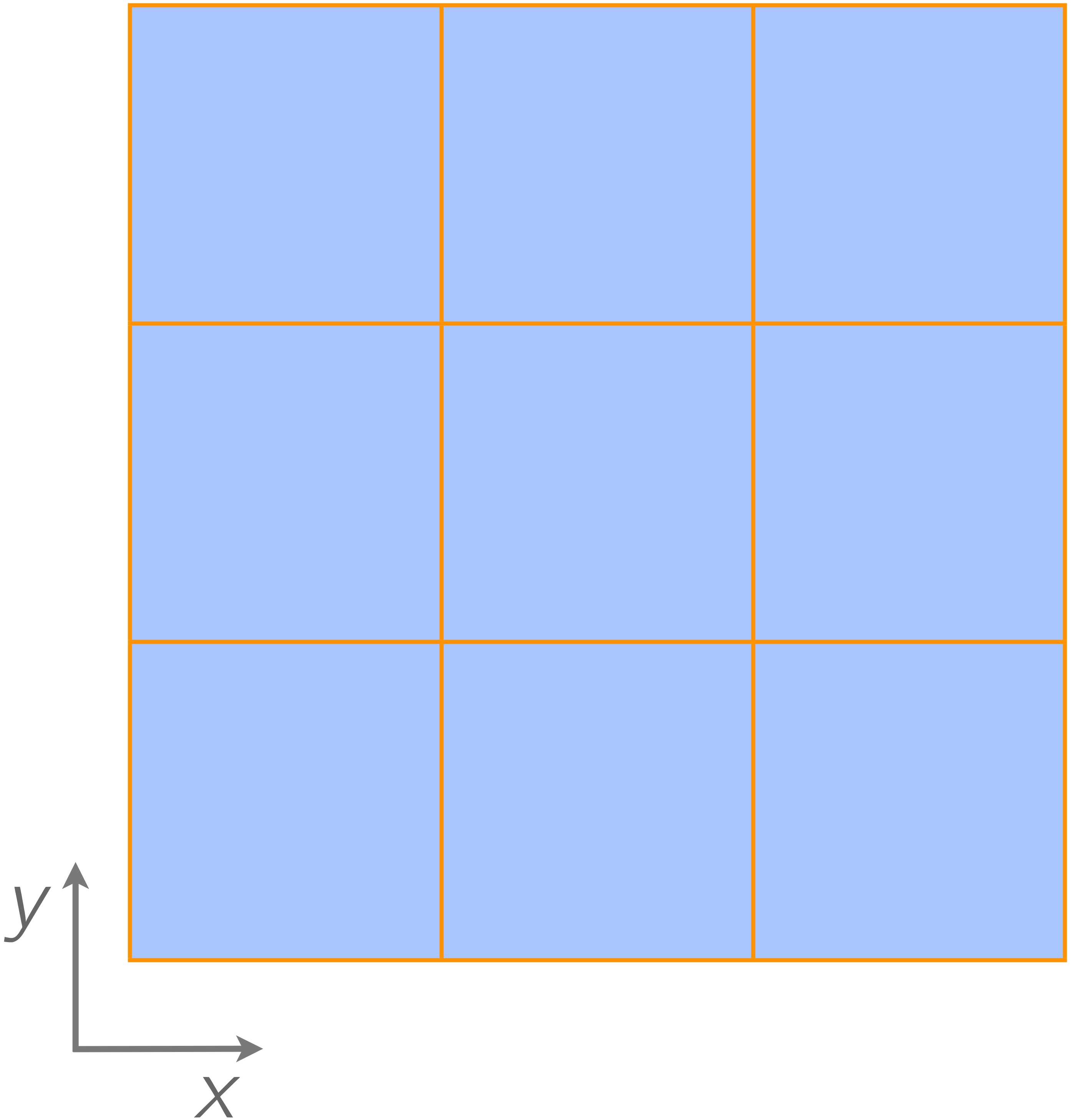
$$\frac{\partial}{\partial t}(\rho_j\phi_j\Delta V_j)+\sum_{f=1}^{n_{\text{faces}}}\vec{F}_f(\phi)\cdot\overrightarrow{dA_f}=s_j\Delta V_j$$

Der Flussvektor $\vec{F}(\phi)$



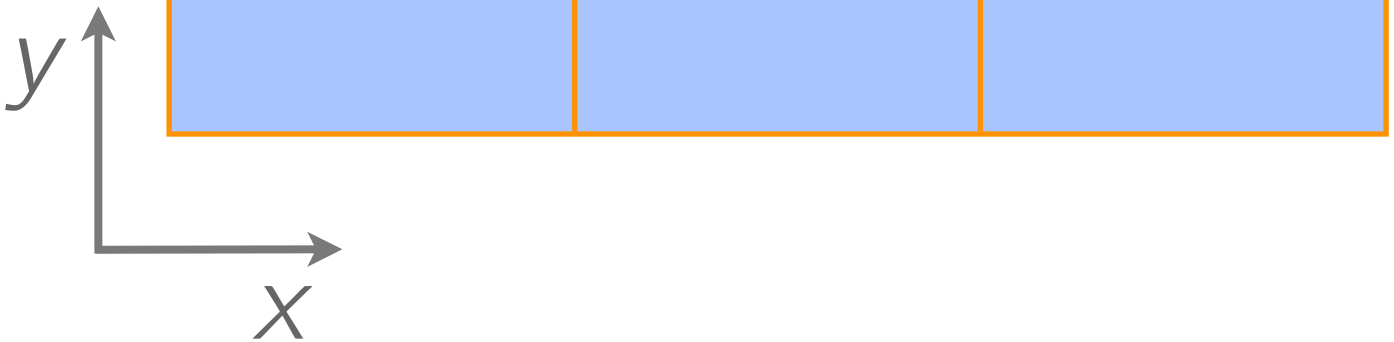
$$\vec{F}_{AB}(\phi) \cdot \vec{dA} = f(\phi)\Delta y - g(\phi)\Delta x$$

Datenstruktur



```
struct sData{


};
```

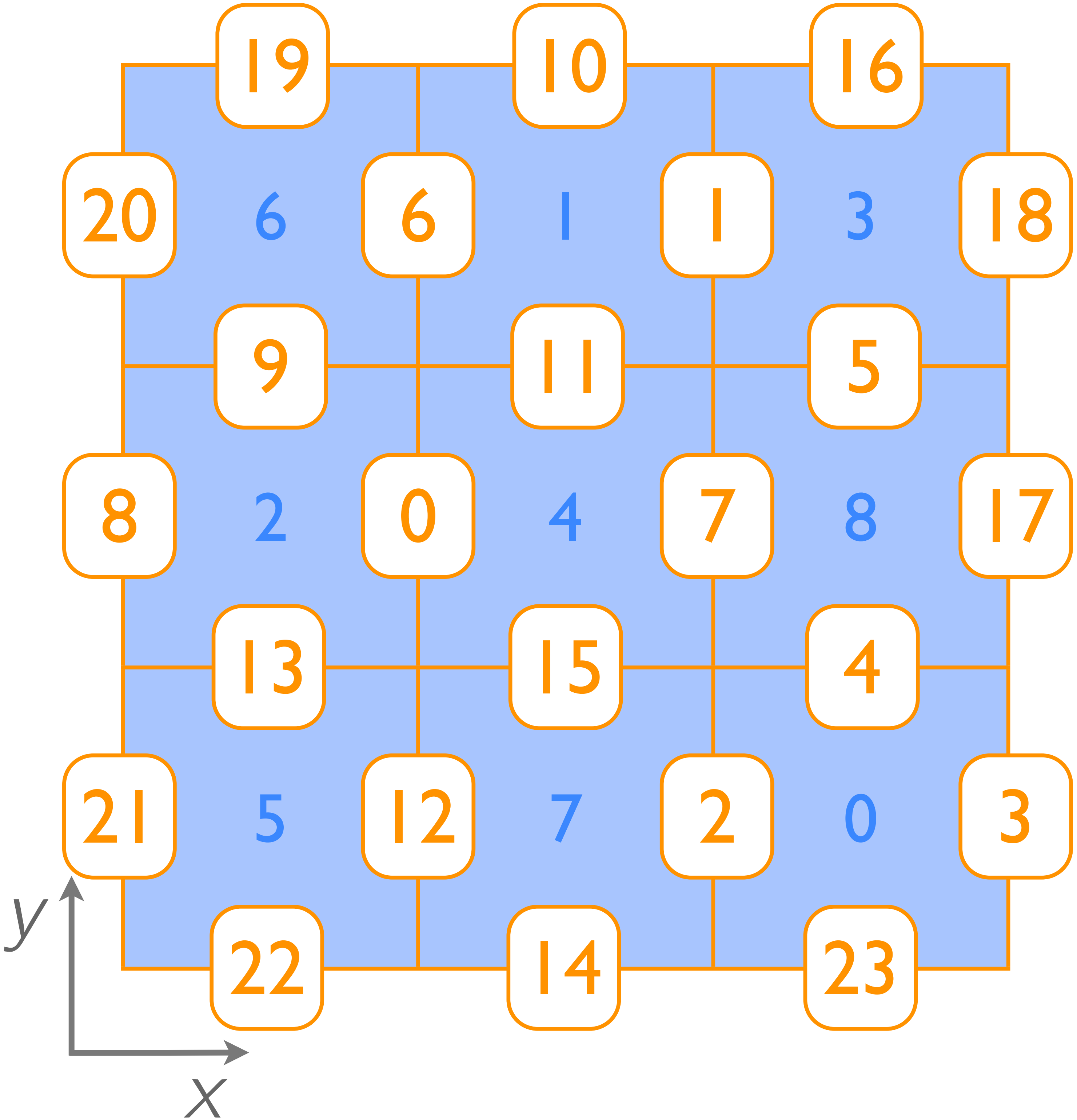
Datenstruktur



6	1	3
2	4	8
5	7	0

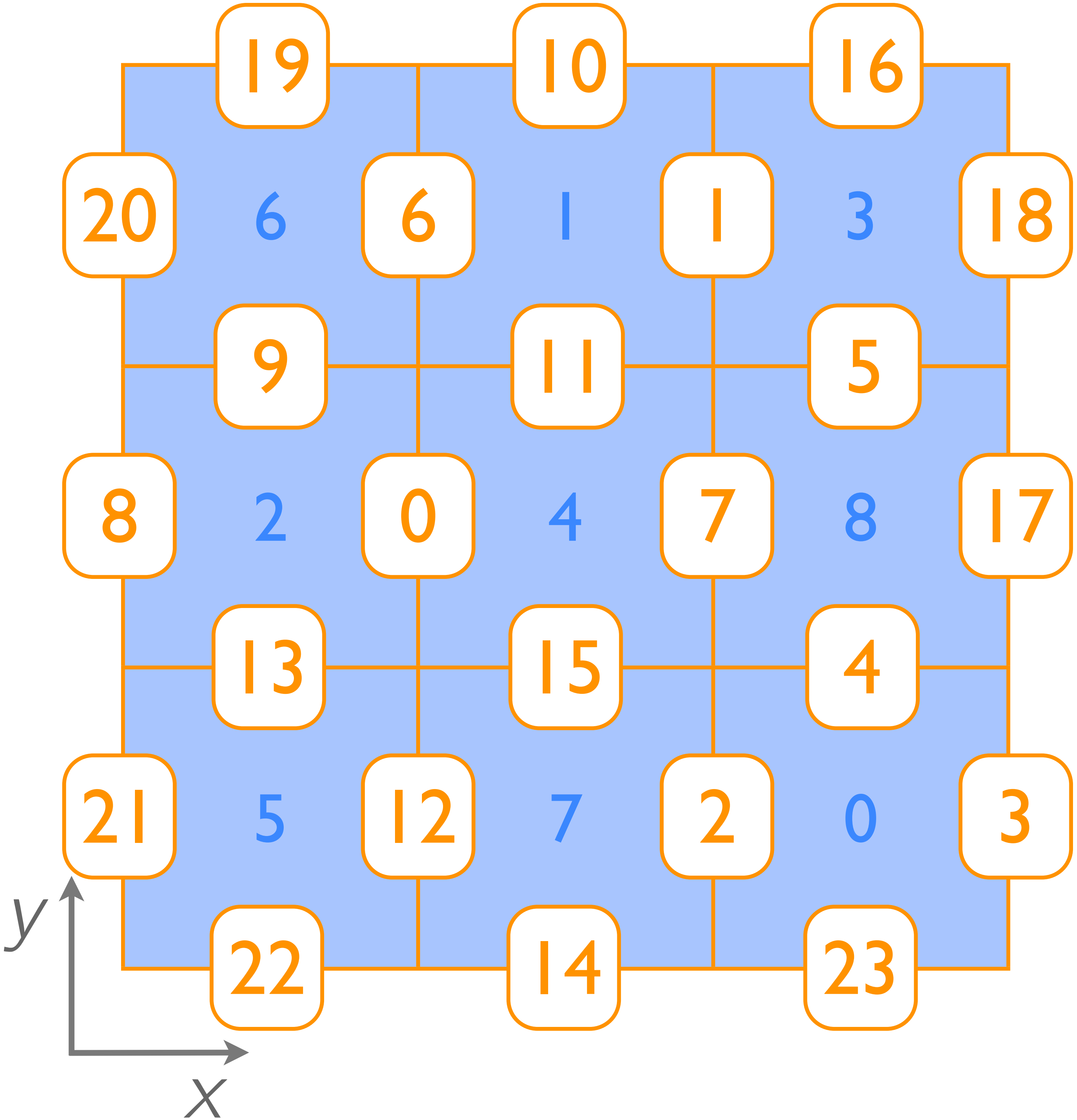
```
struct sData{
            sCell* cells;

};
struct sCell{
            int id;
            double xy[2];


};
```

Datenstruktur



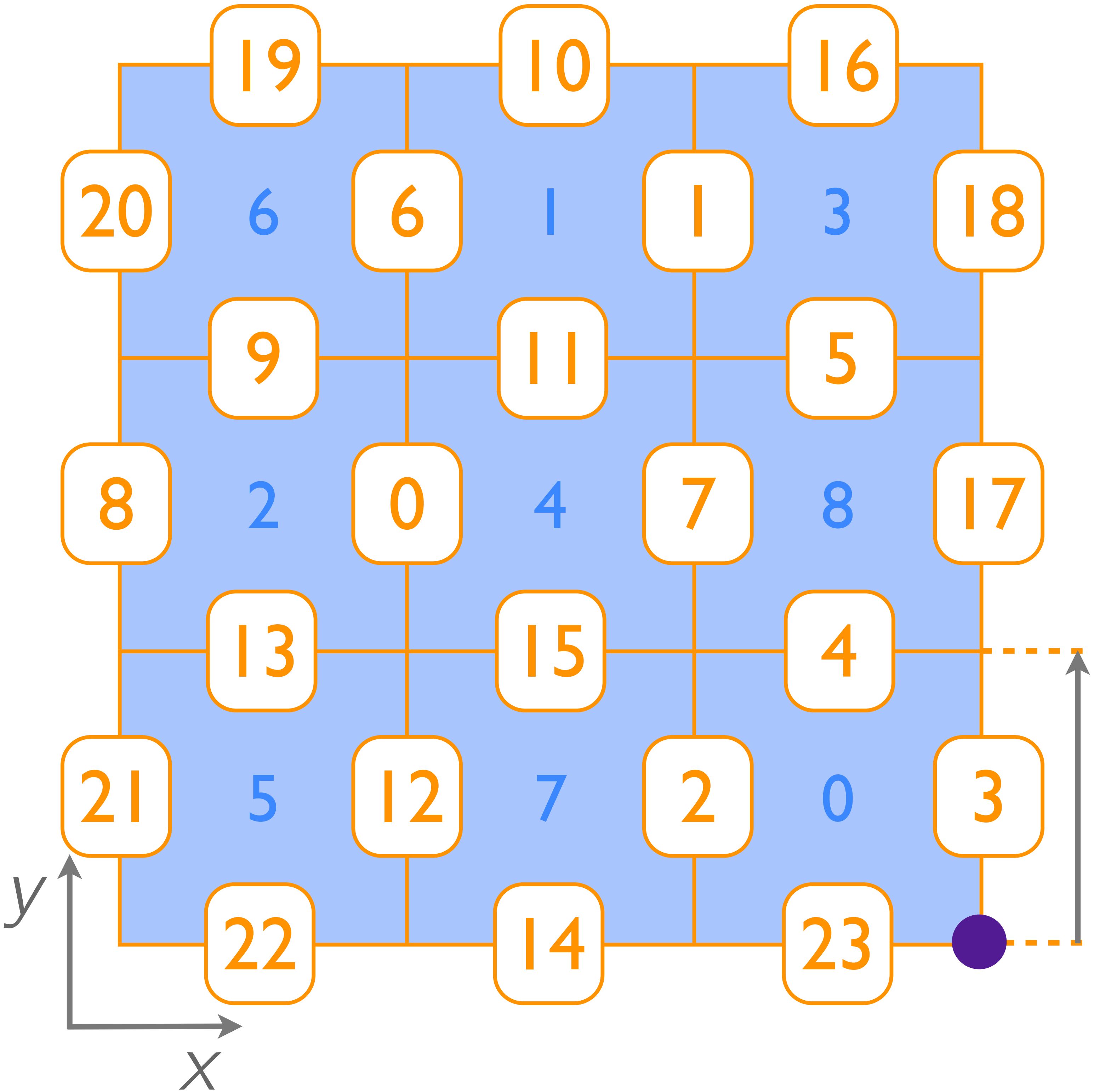
```
struct sData{
                sCell* cells;
                sFace* faces;
};
struct sCell{
                int id;
                double xy[2];



};
struct sFace{
                int id;




};
```

Datenstruktur



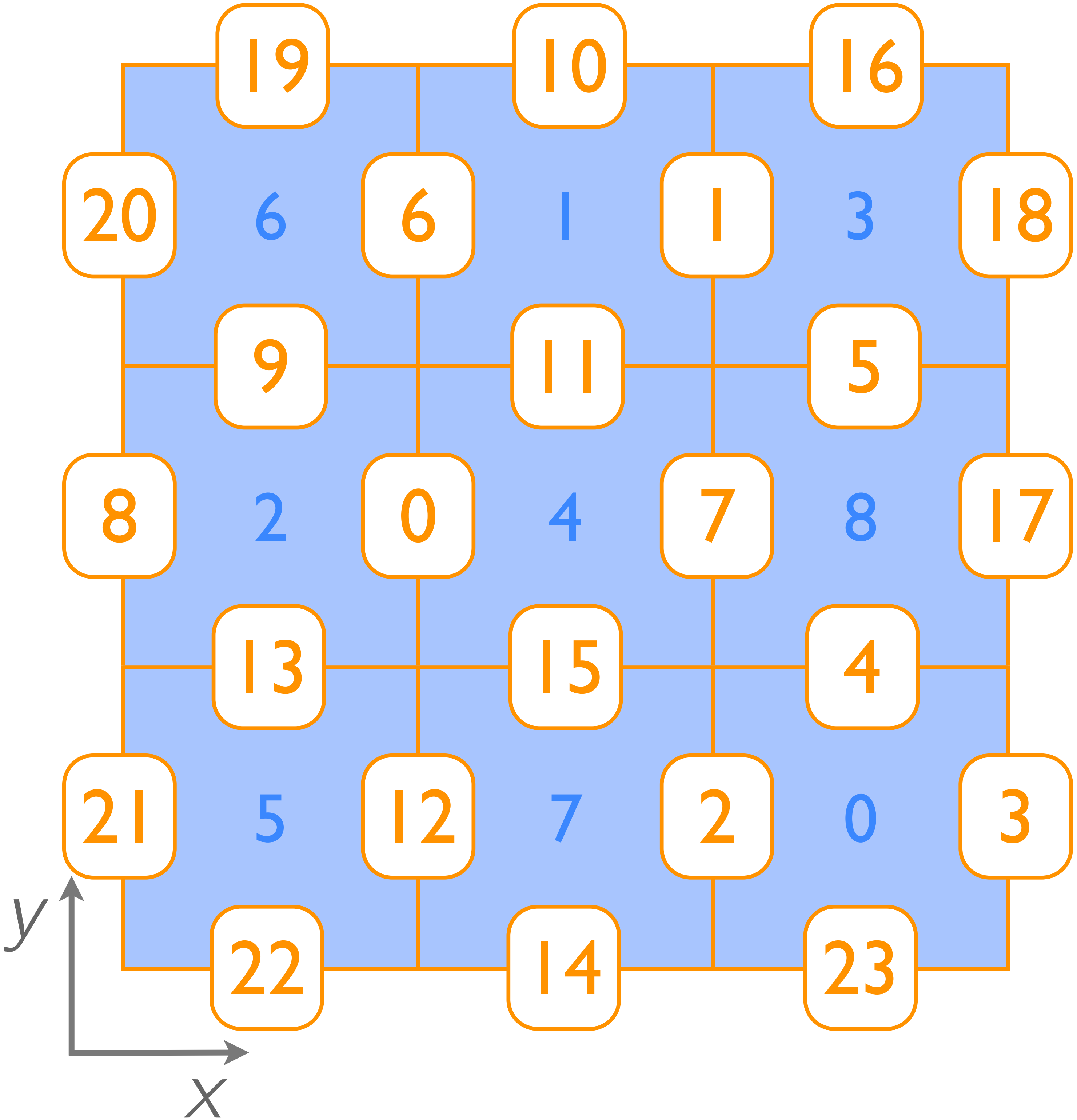
```
struct sData{
                sCell* cells;
                sFace* faces;
};
struct sCell{
                int id;
                double xy[2];
                sFace* cFaces[4];

};
struct sFace{
                int id;



};
```

Datenstruktur



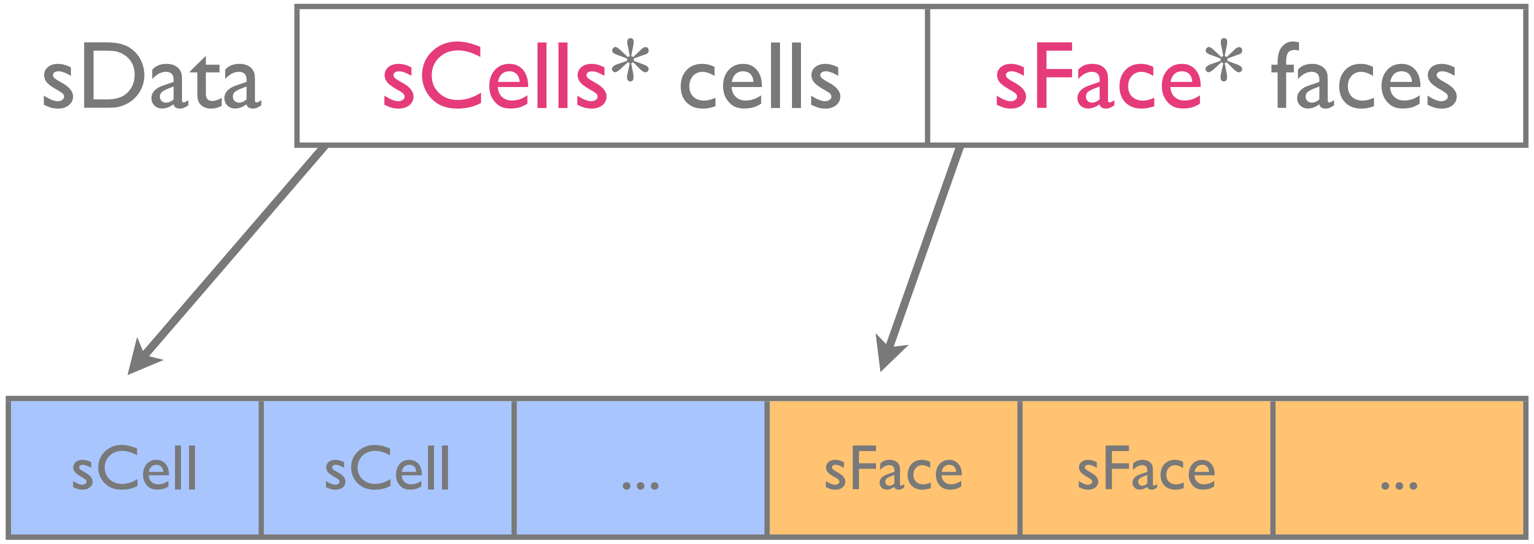
```
struct sData{
        sCell* cells;
        sFace* faces;
};
struct sCell{
        int id;
        double xy[2];
        sFace* cFaces[4];

};
struct sFace{
        int id;
        double xy[2];
        double deltaxy[2];

};
```

Datenstruktur



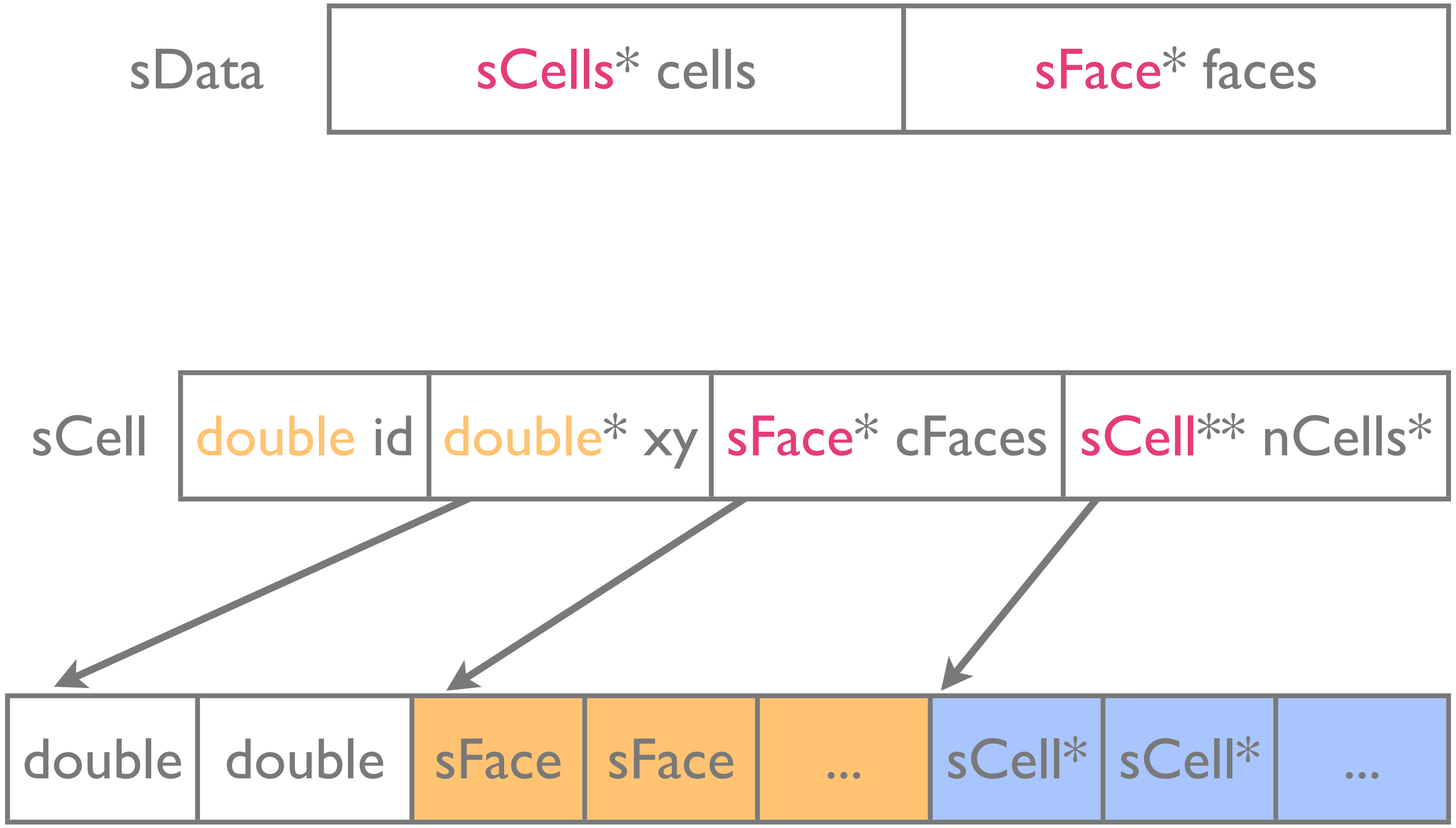
```
struct sData{
        sCell* cells;
        sFace* faces;
};
struct sCell{
        int id;
        double xy[2];
        sFace* cFaces[4];
        sCell** nCells;
};
struct sFace{
        int id;
        double xy[2];
        double deltaxy[2];
        sCell* nCells[2];
};
```

Datenstruktur



```
struct sData{
        sCell* cells;
        sFace* faces;
};
struct sCell{
        int id;
        double xy[2];
        sFace* cFaces[4];
        sCell** nCells;
};
struct sFace{
        int id;
        double xy[2];
        double deltaxy[2];
        sCell* nCells[2];
};
```

Datenstruktur



sData

sCells* cells

sFace* faces

sCell

double id

double* xy

sFace* cFaces

sCell** nCells*

double

double

sFace

sFace

...

sCell*

sCell*

...

```
struct sData{
          sCell* cells;
          sFace* faces;
};
struct sCell{
          int id;
          double xy[2];
          sFace* cFaces[4];
          sCell** nCells;
};
struct sFace{
          int id;
          double xy[2];
          double deltaxy[2];
          sCell* nCells[2];
};
```

Datenstruktur

cells.cfg

```
cells 9
#       cType   = 4     (Pixel)
#       id      cType   x       y       face1   face2   face3   face4
        0       4       2.5     0.5     23      3       4       2
        1       4       1.5     2.5     11      1       10      6
        2       4       0.5     1.5     13      0       9       8
        3       4       2.5     2.5     5       18      16      1
        4       4       1.5     1.5     15      7       11      0
        5       4       0.5     0.5     22      12      13      21
        6       4       0.5     2.5     9       6       19      20
        7       4       1.5     0.5     14      2       15      12
        8       4       2.5     1.5     4       17      5       7
```

```
boundaries
#               id      bType   value0
                0       1       3.0
                1       1       1.755
                2       1       1.0
                3       1       3.0
                5       1       1.0
                6       1       1.0
                7       1       1.755
                8       1       3.0
```

```
initvalues
#               id      phi
                -1      2.0     # default
```

Datenstruktur

faces.cfg

```
faces 24
#       id   x   y   dx   dy
        0    1   1   0    1
        1    2   2   0    1
        2    2   0   0    1
        3    3   0   0    1
        4    2   1   1    0
        5    3   2   -1   0
        6    1   3   0    -1
        7    2   2   0    -1
        8    0   1   0    1
...
```

```
boundaries 12
#       type:  1=Skip,  2=Const.
#       id     bType    value0    value1
        3      1        9999      9999
        8      1        9999      9999
        10     2        9999      0.0
        14     2        9999      0.0
        16     2        9999      0.0
        17     1        9999      9999
        18     1        9999      9999
        19     2        9999      0.0
...
```