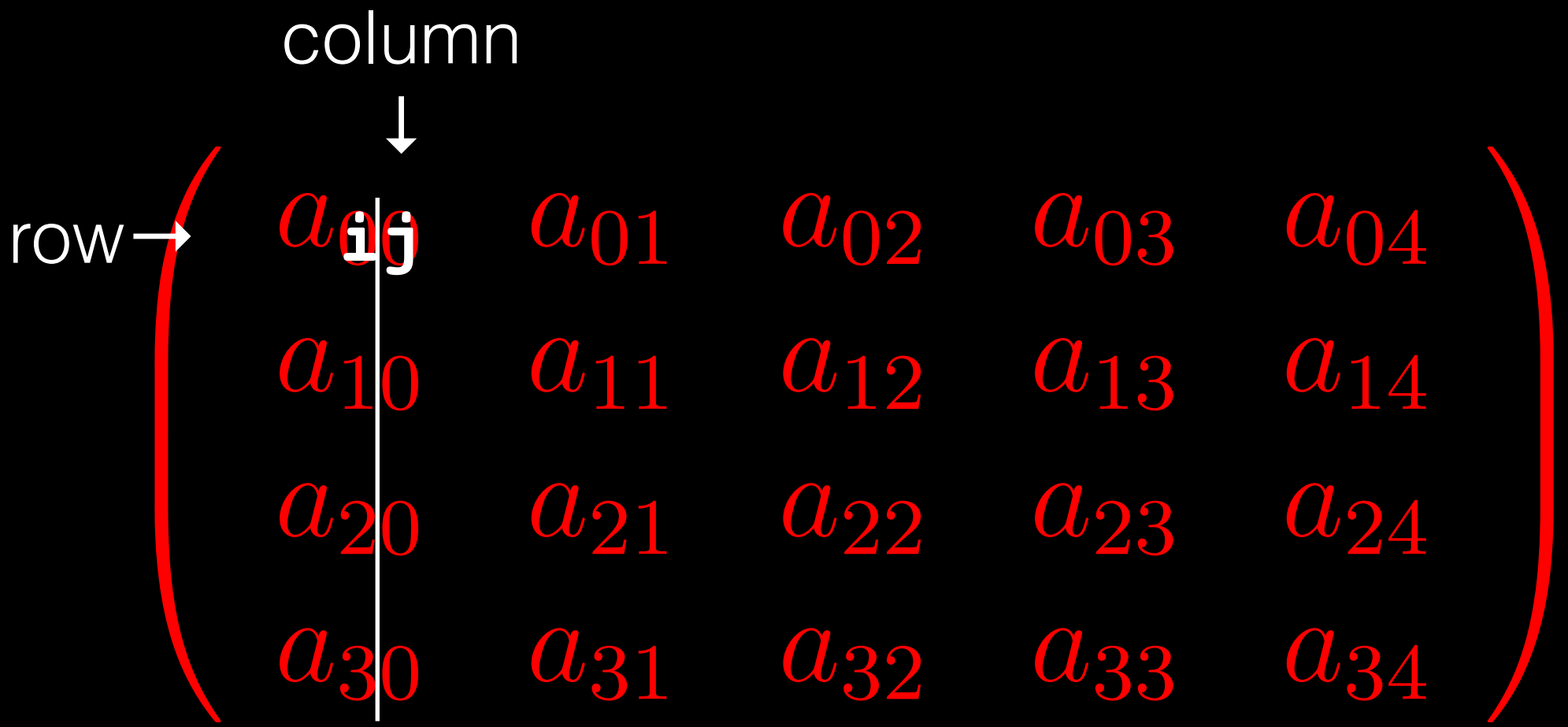
4 Matrices

1. Structs. The data type struct. Structs as parameters.
2. Arrays. array<>. **Multidimensional arrays.** array as parameter
3. String of chars. string. string as parameter.
4. Examples

arrays of arrays

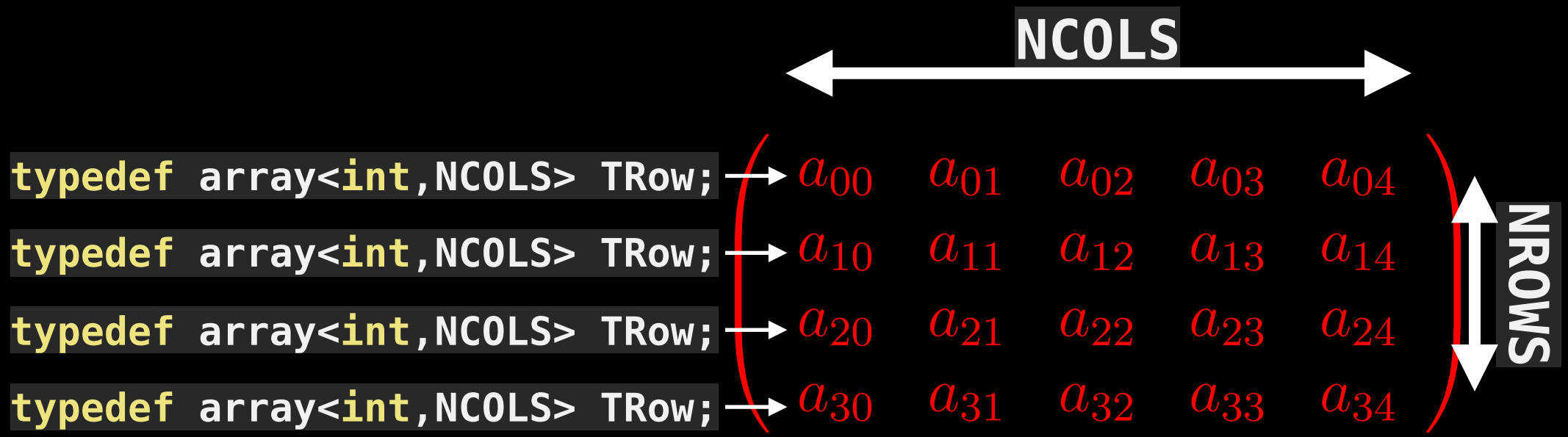
`a[row][column]`

`a[i][j]`



`typedef`: is especially convenient here

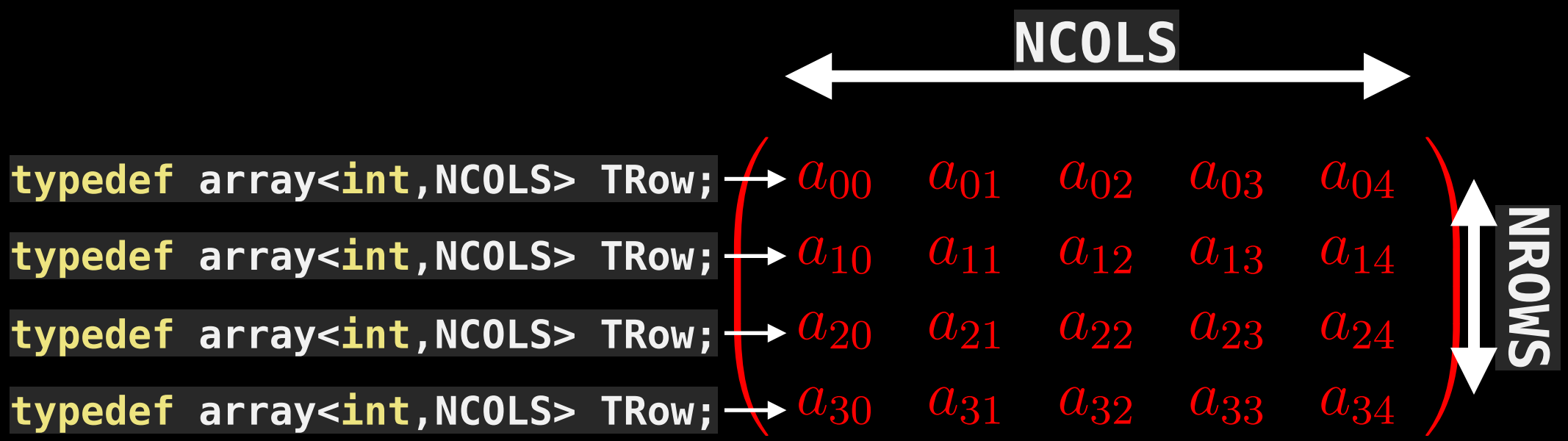
```
const int NROWS = 4;
const int NCOLS = 5;
typedef array<int,NCOLS> TRow;
typedef array<TRow,NROWS> TMatrix;
```



`typedef`: all of it at once

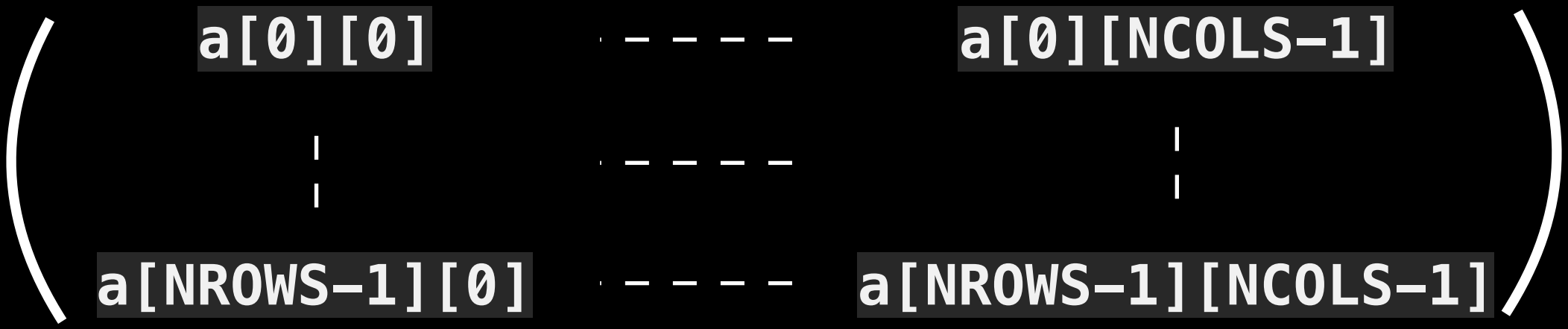
```
typedef array<array<int,NCOLS>,NROWS> TMatrix;
```

be careful with the order of the limits here



```
typedef array<array<int,NCOLS>,NROWS> TMatrix;
```

be careful with the order of the limits here



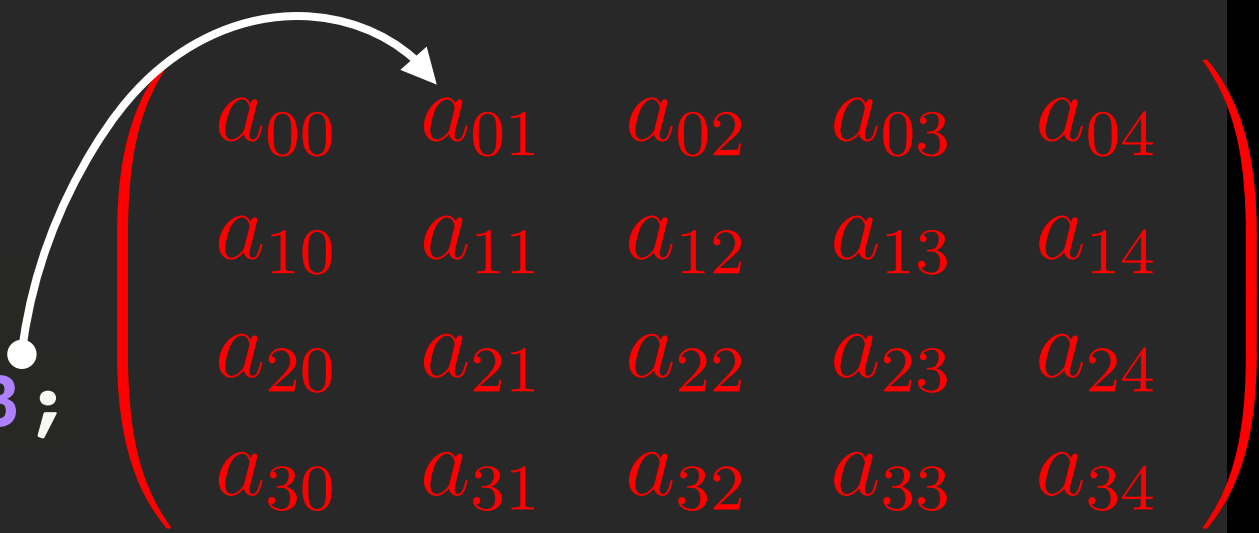
Declaring a matrix

```
const int NROWS = 4;
const int NCOLS = 5;
typedef array<array<float,NCOLS>,NROWS> TMatrix;
```

example

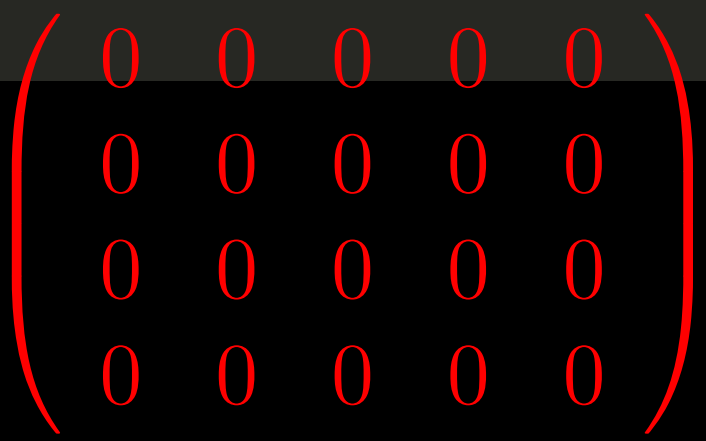
```
const int NROWS = 4;
const int NCOLS = 5;
typedef
 array<array<float,NCOLS>,NROWS> TMatrix2;

int main()
{
    TMatrix m, a;
    a[0][1] = 32.3;
```



traversing

```
TMatrix a;
// set it all to 0
for (int i = 0; i < NROWS; ++i) {
    for (int j = 0; j < NCOLS; ++j) {
        a[i][j] = 0;
    }
}
```



unitary matrix, how?

```
TSqMatrix a;
// set diagonal to 1, rest t 0
for (int i = 0; i < NELE; ++i)
 for (int j = 0; j < NELE; ++j)
  if (i==j) a[i][j]=1; else a[i][j]=0;
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
a[i][j] = (i==j)? 1 : 0;
```

quick initialisation

```
const int N = 3;
typedef array<array<int,N>,N> TSqMat;

TSqMat a = {{
             {{1,2,3}},
             {{4,5,6}},
             {{7,8,9}}
           }};
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Exercise

- Build a function that returns the multiplication of two square matrices:

```
const int N = 3;
typedef array<array<float,N>, N> TSqTMat;

TSqTMat mult(TSqTMat a, TSqTMat b);
```

- To see the result you will want a procedure to print it:

```
void print(TSqTMat a);
```