ROW-MAJOR ORDERING AND INITIALIZER LISTS

Mapping array elements to memory locations

and compile time array initialization

Delroy A. Brinkerhoff

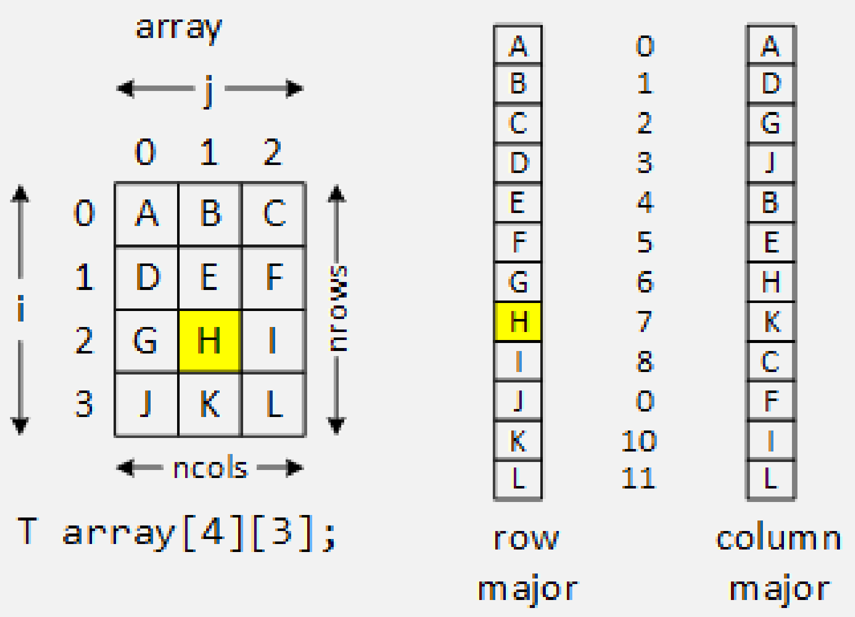
PASSING TWO-DIMENSIONAL ARRAYS

```
char  a1[4][3] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L' };
char  a2[3][2] = { 'u', 'v', 'w', 'x', 'y', 'z' };
```

```
void print(char array[][3], int i, int j)
{
    cout << array[i][j] << endl;
}

void print(char array[][2], int i, int j)
{
    cout << array[i][j] << endl;
}
```

STORING 2D ARRAYS IN MEMORY



- Row-major mapping
  - i * ncols + j
  - `void print(char array[][3], int i, int j)`

- Example:
  - array[2][1]
  - 2 * 3 + 1 = 7

PROGRAMMER-IMPLEMENTED INDEXING

```
void print(char* array, int i, int j, int ncols)
{
    cout << array[i * ncols + j] << endl;
}
```

```
char a1[4][3] = { 'A', 'B', 'C', 'D', 'E', 'F',
                  'G', 'H', 'I', 'J', 'K', 'L' };
char a2[3][2] = { 'u', 'v', 'w', 'x', 'y', 'z' };

print((char *)a1, 2, 1, 3);
print((char *)a2, 1, 0, 2);
```

SYNTHESIZING 2D ARRAYS

```
inline int index(int row, int col, int ncols)
{
    return row * ncols + col;
}

char* array = new char[nrows * ncols];
cout << array[index(2, 1, 3)] << endl;

char  array[4][3];
cout << *((char*)(array) + index(2, 1, 3)) << endl;
```

GENERALIZING ARRAY FUNCTIONS

```
void print(char array[][3], int i, int j)
{
    cout << array[i][j] << endl;
}

void print(char array[][2], int i, int j)
{
    cout << array[i][j] << endl;
}


void print(char* array, int i, int j, int ncols)

{

    cout << array[i * ncols + j] << endl;

}
```

```
char a1[4][3];
char a2[3][2];

print(a1, 2, 1);
print(a2, 1, 0);
print((char *)a1, 2, 1, 3);
print((char *)a2, 1, 0, 2);
```