CSE 220 - C Programming
Fall 2015
Multidimensional Arrays
Multidimensional Arrays
Would you define a separate array for each student?

```c
float student1[10];
float student2[10];
float student3[10];
...```
A two-dimensional array consists of both **rows** and **columns** of elements. It is essentially a **matrix**.

<table>
<thead>
<tr>
<th>Row 0</th>
<th>Row 1</th>
<th>Row 2</th>
<th>Row 3</th>
<th>Row 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>-4</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column 0</th>
<th>...</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>-4</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Declaration of 2D- Arrays in C

To declare a two-dimensional array, we merely use two sets of square brackets.
- The first contains the **number of rows**
- The second contains the **number of columns**

```c
array_Type arrayName [ numberOfRows ] [ numberOfColumns ];
```

- The **type of each entry** in the 2D-array
- The **name of the 2D-array**
- The **number of rows** in the 2D-array
- The **number of columns** in the 2D-array
Example of 2D- Arrays in C

array_Type arrayName [ numberOfRows ] [ numberOfColumns ];

int SudokuTable [9][9];

- The name of this array is “SudokuTable”.
- The type of individual elements is int
- This declaration sets aside a chunk of memory that is big enough to hold 81 integers.
- It does NOT initialize those memory locations to 0 or any other value.
How are 2D-arrays formatted in memory?

• In memory:

A static two-dimensional array looks like an array of arrays - it's just laid out contiguously in memory.
2D- Arrays

datatype array_name[row_size][column_size];

int matrix[3][4];

Row 0          | 4  | 1  | 0  | 2  |
Row 1          | -1 | 2  | 4  | 3  |
Row 2          | 0  | -1 | 3  | 1  |

Column 0       |     |     |     |     |
Column 1       |     |     |     |     |
Column 2       |     |     |     |     |
Column 3       |     |     |     |     |

in memory
Initialization 2-D Array
Initialization 2-D Array

• Initialize to constant values:

```c
int a[4][3] = {
    {1, 2, 3},  // Row 0
    {4, 5, 6},  // Row 1
    {7, 9, 10}, // Row 2
    {11, 12, 13} // Row 3
};
```

A 2D-array is actually an array of 1D-arrays
Initialization 2-D Array

```c
int x[4][4] = {{2, 3, 7, 2},
                {7, 4, 5, 9},
                {5, 1, 6, -3},
                {-5, 2, 6, -3},
            };
```

You can also **skip explicitly** setting the size of rows if you initialize the array.

```c
int x[][4] = {{2, 3, 7, 2},
              {7, 4, 5, 9},
              {5, 1, 6, -3},
              {2, 5, -1, 3}
          };
```
Initialization

• Short initializers:
  – Fill first few rows as specified, remaining with 0’s:
    
    ```
    int a[4][3] = {{1, 2, 3}, {4, 5, 6}};
    ```

  – Fill first few elements of each row as specified, remaining with 0’s:
    
    ```
    int a[4][3] = {{1}, {4, 5, 6}; {7, 9}, {11, 12, 13} }; 
    ```
Accessing Individual Entries of a 2-D Array
# 2D- Arrays

```c
int table[5][3]; // 5 Rows, 3 Columns
```

A 2D array is a collection of elements arranged in a two-dimensional grid. In this example, we have a 2D array named `table` which consists of 5 rows and 3 columns. Each element in the array can be accessed using its row and column indices.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><code>table[0][0]</code></td>
<td><code>table[0][1]</code></td>
<td><code>table[0][2]</code></td>
</tr>
<tr>
<td>1</td>
<td><code>table[1][0]</code></td>
<td><code>table[1][1]</code></td>
<td><code>table[1][2]</code></td>
</tr>
<tr>
<td>2</td>
<td><code>table[2][0]</code></td>
<td><code>table[2][1]</code></td>
<td><code>table[2][2]</code></td>
</tr>
<tr>
<td>3</td>
<td><code>table[3][0]</code></td>
<td><code>table[3][1]</code></td>
<td><code>table[3][2]</code></td>
</tr>
<tr>
<td>4</td>
<td><code>table[4][0]</code></td>
<td><code>table[4][1]</code></td>
<td><code>table[4][2]</code></td>
</tr>
</tbody>
</table>
2D- Arrays

arrayName [ row_number ] [ column_number ];

- Just like 1D arrays, once you have specified the index, you are just working with a single variable of the given data type.

```c
scanf("%d", &table[2][1]);

printf("%d", table[2][1]);
```
2D- Arrays

arrayName [ row_number ] [column_number ];

- Just like 1D arrays, once you have specified the index, you are just working with a single variable of the given data type.

```c
printf(“%d”, table[2][1]);
scanf(“%d”, &table[2][1])
```

- 2D arrays work well with (for) loops like 1D arrays. However, to access all elements, typically you will need **nested loops** for 2D arrays.
Initialization

```c
int i, j;
int matrix[3][4];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        matrix[i][j] = i;
```
Initialization

```c
int i, j;
int matrix[3][4];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        matrix[i][j] = i;
```
Initialization

```c
int i, j, matrix[3][4];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        matrix[i][j] = j;
```

Table initialization:
Initialization

```c
int i, j, matrix[3][4];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        matrix[i][j] = j;

```

i

j

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
int i, j, matrix[3][4];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        matrix[i][j] = j;

Example(I)

- Write a program that prints the sum of all numbers in a two dimensional array with N rows and M columns:

The name of array is: values [N][M]
Example(I)

- Write a program that prints the sum of all numbers in a two dimensional array with N rows and M columns:

```c
//Declare array
int values[N][M], sum = 0;
//Read values
....
//Print content
for (i=0; i<=N-1; i++)
{
    for (j=0; j<=M-1; j++)
        sum = sum + values[i][j];
}
printf("%d ",sum);
```
Loop Structures and 2D-Array

• Write a program that prints the content of a two
dimensional array with N rows and M columns:

//Declare array
int values[N][M];
//Read values
....
//Print content
for (i=0; i<=N-1; i++) {
    for (i=0; i<=M-1; i++) {
        sum = sum + values[i][i];
    }
}

What's wrong here?
Loop Structures and 2D-Array

• Write a program that prints the content of a two dimensional array with N rows and M columns:

```c
//Declare array
int values[N][M];
//Read values
....
//Print content
for (i=0; i<=N-1; i++) {
    for (i=0; i<=M-1; i++) {
        sum = sum + values[i][i];
    }
}
```

What's wrong here?
- Must use different variable names
• Write a program that prints the content of a two dimensional array with \( N \) rows and \( M \) columns:

```c
int i, j;
int a[3][3] = { {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9} };

for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
        printf("%d ", a[i][j]);
    }
}
```
• Write a program that prints the content of a two dimensional array with N rows and M columns:

```c
int i, j;
int a[3][3] = { {1, 2, 3},
                 {4, 5, 6},
                 {7, 8, 9} };

for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
        printf("%d ", a[i][j]);
    }
}
```

Output: 1 5 9

<table>
<thead>
<tr>
<th>step</th>
<th>i (outer)</th>
<th>i (inner)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>3 (exit)</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td></td>
<td>3 (exit)</td>
</tr>
</tbody>
</table>
Write a program that prints the content of a two-dimensional array with N rows and M columns:

```c
//Declare array
int values[N][M];

//Read values
....

//Print content
for (i=0, j=0; i<=N-1, j<=M-1; i++,j++) {
    printf("%d ", values[i][j]);
}
```

What's wrong with this one?
• Write a program that prints the content of a two dimensional array with N rows and M columns:

```c
int i, j;
int a[3][3] = { {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9} };
for (i=0, j=0; i < 3, j < 3; i++, j++) {
    printf("%d ", a[i][j]);
}
```

Output:

```
1 5 9
```

<table>
<thead>
<tr>
<th>step</th>
<th>i</th>
<th>j</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 (a[0][0])</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5 (a[1][1])</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>9 (a[2][2])</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3 (exit)</td>
<td>3 (exit)</td>
</tr>
</tbody>
</table>
Loop Structures and 2D-Array

• Write a program that prints the content of a two dimensional array with $N$ rows and $M$ columns:

correct implementation

```c
//Declare array
int values[N][M];
//Read values
    ....
//Print content
for (i=0; i<=N-1; i++)
{
    for (j=0; j<=M-1; j++)
    {
        printf("%d ", values[i][j]);
    }
}
```

<table>
<thead>
<tr>
<th>step</th>
<th>i</th>
<th>j</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$a[0][0]$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$a[0][1]$</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>$a[0][2]$</td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>$M$ (exit)</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>$a[1][0]$</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>$a[1][1]$</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>$a[1][2]$</td>
</tr>
<tr>
<td>9</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>$M$ (exit)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Find the maximum of int matrix[3][4]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Output: 4
Find the maximum of int matrix[3][4]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

int max = matrix[0][0];
for (i=0; i<3; i++)
    for (j=0; j<4; j++)
        if (matrix[i][j] > max)
            max = matrix[i][j];
Find the number of times x appears in int matrix[3][4]

```c
int count = 0;
    for (i=0; i<3; i++)
        for (j=0; j<4; j++)
            if (matrix[i][j] == x)
                count = count + 1;
```
Exercise: Compute the addition of two matrices

(write down your name, PID, and sec#, hand in your paper)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \text{matrix1} \]

\[ \text{matrix2} \]

\[ \text{sum} \]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 1 & 0 & 2 \\
-1 & 2 & 4 & 3 \\
0 & -1 & 3 & 1 \\
\end{array} + \begin{array}{cccc}
0 & 1 & 2 & 3 \\
3 & -1 & 3 & 1 \\
1 & 4 & 2 & 0 \\
2 & 1 & 1 & 3 \\
\end{array} = \begin{array}{cccc}
0 & 1 & 2 & 3 \\
3 & 0 & 3 & 3 \\
0 & 6 & 6 & 3 \\
2 & 0 & 4 & 4 \\
\end{array}
\]
Compute the addition of two matrices

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 1 & 0 & 2 \\
-1 & 2 & 4 & 3 \\
0 & -1 & 3 & 1 \\
\end{array} + \begin{array}{cccc}
0 & 1 & 2 & 3 \\
3 & -1 & 3 & 1 \\
1 & 4 & 2 & 0 \\
2 & 1 & 1 & 3 \\
\end{array} = \begin{array}{cccc}
0 & 1 & 2 & 3 \\
3 & 0 & 3 & 3 \\
0 & 6 & 6 & 3 \\
2 & 0 & 4 & 4 \\
\end{array}
\]

```c
int matrix1[3][4]={ {0,1,0,2}, {-1,2,4,3}, {0,-1,3,1} },
    matrix2[3][4] = { {3,-1,3,1}, {1,4,2,0}, {2,1,1,3} },
    sum[3][4];

for (i=0; i<3; i++)
    for (j=0; j<4; j++)
    {
        sum[i][j] = matrix1[i][j]+matrix2[i][j];
        printf(“%d”,sum[i][j]);
    }
```
## Transpose

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
Transpose

\[
\begin{array}{ccc}
1 & 5 & 3 \\
4 & 2 & 6 \\
\end{array}
\]

\[
\begin{array}{cc}
1 & 4 \\
5 & 2 \\
3 & 6 \\
\end{array}
\]

```c
int a [2][3] ={{1,5,3},{4,2,6}}
int b[3][2];
    for(i=0; i<2; i++) {
        for(j=0; j<3; j++) {
            b[j][i] = a[i][j];
        }
    }
```
Summary

• Array types
  – One dimensional
  – Multi dimensional

• Initialization

• Access and Bounds