2D ARRAYS

CS111 Lab
Queens College, CUNY
Instructor: Kent Chin
TWO-DIMENSIONAL (2D) ARRAYS

Just like “regular” 1D arrays, 2D arrays enable us to work with any number of variables! 2D arrays have the following properties:

- **Data-type** – Can be `int`, `char`, `double`, `string`, `bool`, etc
- **Name** – Arrays have names just like variables do!
- **Row Size** – How many rows would you like?
- **Column Size** – How many columns would you like?

```plaintext
type name[x][y];
```

Data-type  Name  Row Size  Column Size
2D ARRAYS IN MEMORY

```plaintext
int a[3][4];
```

Data-type: int, Name: a, Rows: 3, Columns: 4

**2D Array Access**

```plaintext
   a[i][j]
```

Row index start at 0 and end at row_size - 1

Column indexes begin at 0 and end at column_size - 1

Row 0:
- `a[0][0]`
- `a[0][1]`
- `a[0][2]`
- `a[0][3]`

Row 1:
- `a[1][0]`
- `a[1][1]`
- `a[1][2]`
- `a[1][3]`

Row 2:
- `a[2][0]`
- `a[2][1]`
- `a[2][2]`
- `a[2][3]`
We usually use nested for-loops to traverse (travel across) 2D arrays! The outer loop usually takes care of the rows and the inner loop usually takes care of the columns (though the roles may be switched at times):

Our “usual” loops

```c
for (int i = 0; i < rows; ++i) {
    //loop for rows
    for (int j = 0; j < cols; ++j) {
        //loop for columns
        //do something with array entry at [i][j].
    }
}
```
2D ARRAYS INITIALIZATION AND ACCESS

```c++
int a[2][3] = {{4, 7, 1}, {-1, 0, 8}};

for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 3; ++j) {
        cout << a[i][j] << " ";
    }
    cout << endl;
} //print with loops

for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 3; ++j) {
        cin >> a[i][j];
    }
} //input with loops
```

```c++
a[1][0]++;
cout << a[1][0]; //0

a[1][1] += a[1][2];
cout << a[1][1]; //8
```
//10x10 multiplication table!
int main() {
    int table[10][10]; //2D int array with 10 rows and 10 columns

    for (int i = 0; i < 10; ++i) {
        for (int j = 0; j < 10; ++j) {
            table[i][j] = i * j;
        }
    } //assigning values to each array entry

    for (int i = 0; i < 10; ++i) {
        for (int j = 0; j < 10; ++j) {
            cout << table[i][j] << " ";
        }
        cout << endl;
    } //printing table

    return 0;
}
2D ARRAYS AND FUNCTIONS

When specifying a 2D array parameter...

- Row size is optional
- Column size is required

Column size may vary

```cpp
void print(int a[][2], int r, int c) {
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
}  //typical 2D array print function
```

In `main()`:

```cpp
int main() {
    int a[2][2] = {{1, 4}, {0, 8}};
    //pass the NAME of the array
    //to the function!
    print(a, 2, 2);
    return 0;
}
```

Output

```
1 4
0 8
```
2D ARRAYS AND FUNCTIONS

2D arrays are also “passed by reference”.

```c
void sub5(int a[][2], int r, int c) {
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j) {
            a[i][j] -= 5;
        }
    }
}

void print(int a[][2], int r, int c) {
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
} //typical 2D array print function
```

In `main()`:

```c
int main() {
    int a[2][2] = {{1, 4}, {0, 8}};
    cout << "Now: ";
    print(a, 2, 2);
    sub5(a, 2, 2);
    cout << "Later: ";
    print(a, 2, 2);
    return 0;
}
```

```
| Now: | 1 4
| 0 8  |
| Later: | -4 -1
|        | -5 3 |
```