2D ARRAYS

CS111 Lab

Queens College, CUNY

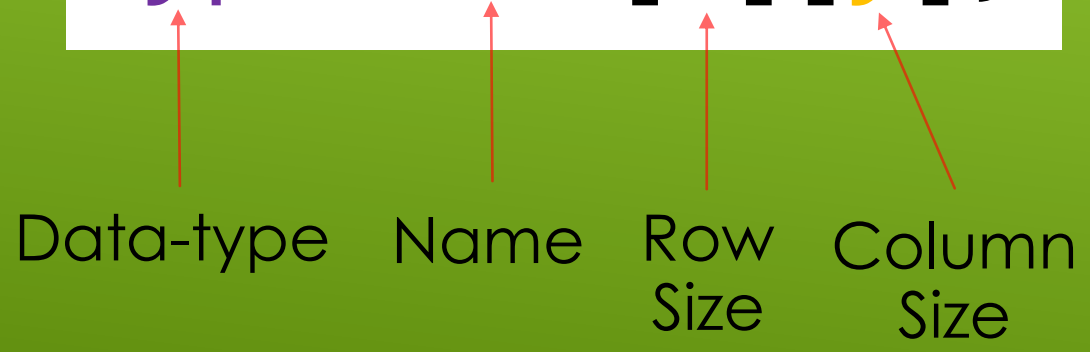
Instructor: Kent Chin

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?

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



2D ARRAYS IN MEMORY

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

Data-type Name Rows Columns

2D Array Access

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

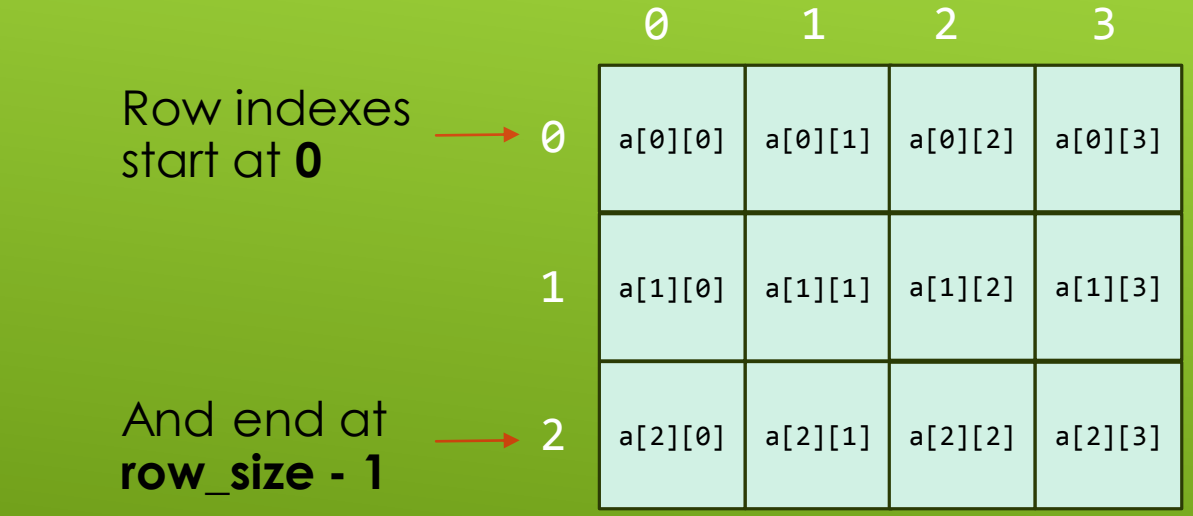
Row Index

Column Index

a:

Column indexes begin at **0**

And end at **column_size - 1**



2D ARRAY ACCESS WITH FOR-LOOPS

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

```
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

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

a:

	0	1	2
0	4	7	1
1	-1	0	8

```
cout << a[0][1]; //7
cout << a[1][2]; //8
cout << a[1][0]; //-1
```

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

```
a[1][1] += a[1][2];
cout << a[1][1]; //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
```

PROGRAM EXAMPLE

```
//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

```
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()*:

```
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”.

```
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()*:

```
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
```