ARRAYS

Lab Instructor: Jean Lai
ARRAYS

- List of items of the same type
- Think of it a list of boxes.
- First box starts at index 0.
- Each box (element) afterwards is the previous index + 1.

- allow you to store more than one item in only one variable.
ARR

Y

AC

DECLARATION

- **SYNTAX**
  
  type array_name [size];

- **EXAMPLES**

  int numArray[100];
  // an array with 100 elements
  double decimalArray[10];
  char grade[10];
ARRAY INITIALIZATION

- **SYNTAX**
  
  type array[size] = { item1, item2, …, item 3};
  type array[index] = item;

- **EXAMPLES**
  
  int children[3] = {2, 12, 1};
  //declaring and initializing at the same time.
  scores[0] = 2.7;
  //array size is already declared.
- Array of size N elements.
- Indexes run from 0 to N-1.

```
scores[0] = 0;
scores[1] = 95;
Scores[2] = 60;
scores[3] = 75;
scores[4] = 45;
scores[5] = 57;
```
Loops allow access to all elements of the array.

**EXAMPLES**

```cpp
for (int i = 0; i < 5; ++i)
    cout << scores[i] << " ";
for (int i = 0; i < 5; ++i)
    cin >> scores[i];
```
ARRAY AND FUNCTIONS

- Indexed variables as Function Arguments
  
  ```
  myFunction(a[3]);
  ```

- Entire Arrays as Function arguments
  
  ```
  void fillup(int scores[], int size);
  ```

- Function MAY NOT return an array in the same way it returns type int or double!

- Only pointers may returned.
SORTING AN ARRAY

- Pass by reference!
Multidimensional Arrays
2D ARRAYS

- To store even more large amount of information of the same type.
- Same as 1D array, but two indexes for the dimension of the array.
- Each index MUST BE enclosed in its own set of square brackets.
2D ARRAY DECLARATION

- **SYNTAX**
  
  ```
  type array_name [size1][size2];
  ```

- **EXAMPLES**
  
  ```
  int numArray[10][10];
  // an array with 10x10 elements (total 100)
  ```
<table>
<thead>
<tr>
<th>Student 1</th>
<th>Student 2</th>
<th>Student 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>Test 2</td>
<td>Test 3</td>
</tr>
<tr>
<td>grade[0][0]</td>
<td>grade[0][1]</td>
<td>grade[0][2]</td>
</tr>
<tr>
<td>grade[1][0]</td>
<td>grade[1][1]</td>
<td>grade[1][2]</td>
</tr>
<tr>
<td>grade[2][0]</td>
<td>grade[2][1]</td>
<td>grade[2][2]</td>
</tr>
</tbody>
</table>
int rows = 3, cols = 3;
int grade = new int[rows][cols];

for (int r = 0; r < rows; ++r)
    for (int c = 0; c < cols; ++c)
        cin >> grade[r][c];
Size of the first dimension is not given, but the remaining dimension size MUST be given in square brackets.

Example

```c
int getMin (int p[][100], int sizedimension1);
```