Passing Arrays to Functions

- Syntax:

```
void printArray(int *array, int
length);


main()
{
  int array[5] = {5, 2, 6, 3, 1};
  printfArray(array, 5);
}
```

Therefore, a pointer should receive the array parameter.

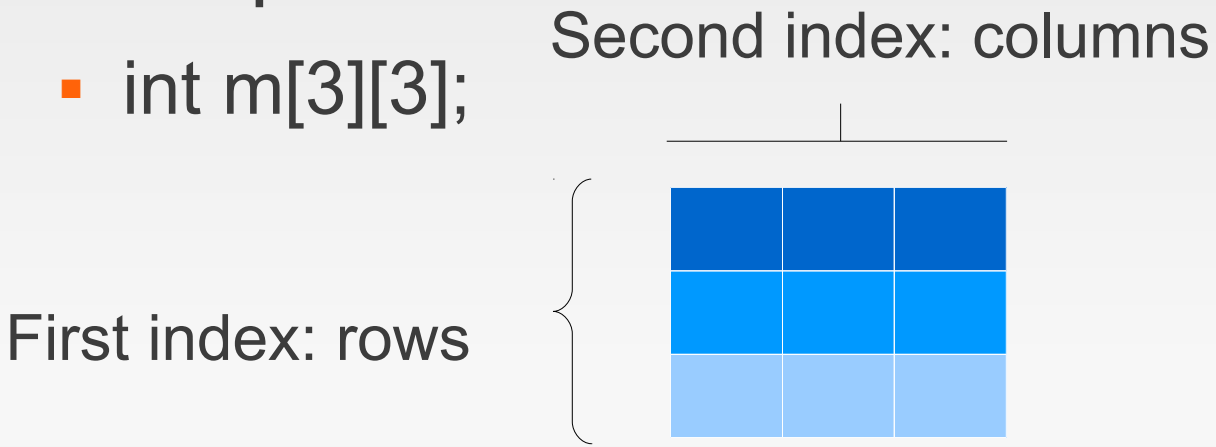
As discussed earlier, is a pointer to the first element of the array.

Passing Arrays to Functions

- Always pass-by-reference
- Passes the address of first element

Multi-dimensional Arrays (1)

- Arrays can have more than one dimension/subscript
- 2-Dimensional (2D) arrays are also common
- Can be visualized a grid with rows and columns
- Example:
  - int m[3][3];

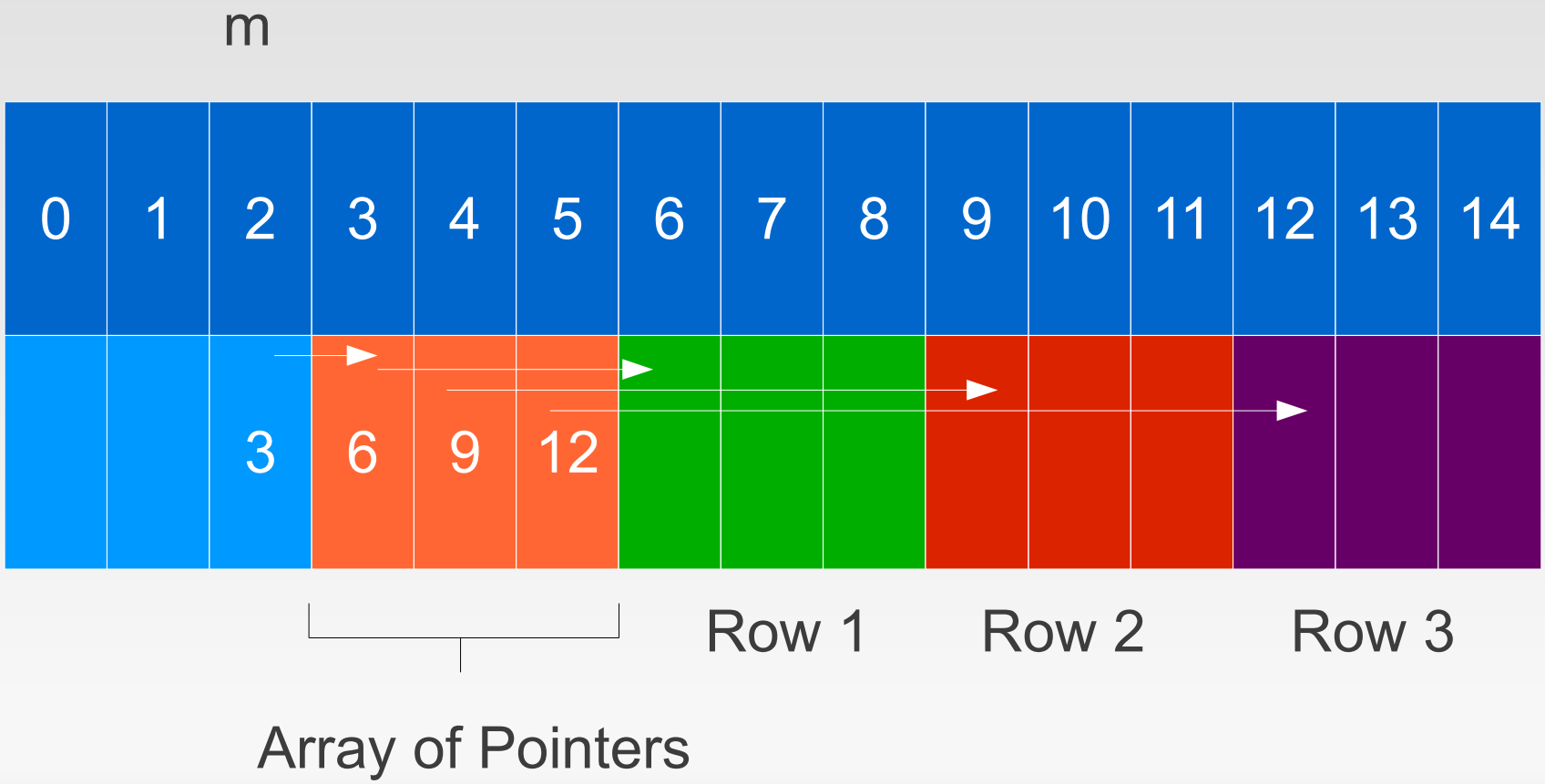


Multi-dimensional Arrays (2)

- Array elements are accessed using two subscripts
- Example:
  - m[1][0] = element on the second row, first column

Memory Allocation of 2D Arrays

- 2D arrays are allocated like this...



Static vs Dynamic (1)

- The variables we have been using so far have been ***static***
- They are allocated memory upon declaration, and deallocted upon program termination
- Size and memory address remain constant throughout program execution
- Pointers can be used for ***dynamic memory allocation***

Dynamic Memory Allocation

- Pointers can be used together with our knowledge of the memory allocation of arrays in order to create dynamic arrays
- Two important functions:
  - `malloc`
  - `free`

The `malloc` Function

- Used to ***dynamically allocate*** space anywhere in the program
- Usage:

```
int *p;
p = (int *) malloc(<size>);
```

- Returns a void pointer
  - Needs to be typecast to the correct pointer type
- Allocates <size> memory cells

The `free` Function

- Used to ***deallocate*** memory allocated using `malloc`
- Usage:
  - `free(<pointer>);`
- Example:
  - `free(p);`
- Every `malloc` needs to have a `free` partner
- If not used to deallocate dynamically allocated memory, will cause ***memory leaks***

What about the size?

- C has a `sizeof` operator
- Takes a data type for a parameter
- Returns the size (in bytes) that is needed to store one variable of that data type
- Example:
  - `sizeof(int)` = returns the number of bytes needed to store one variable of data type `int`

So then...

- ...how do we create a dynamic array?
- ...how do we create a dynamic 2D array?