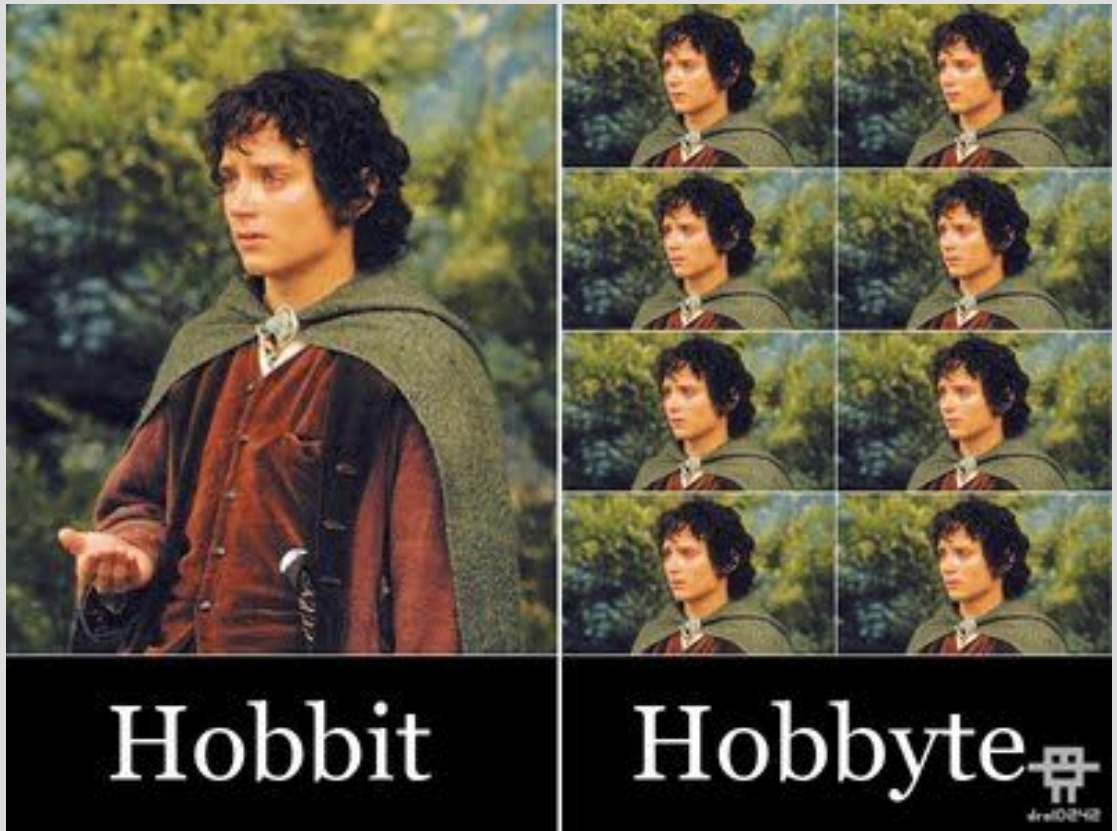
2D Arrays

Ch 7



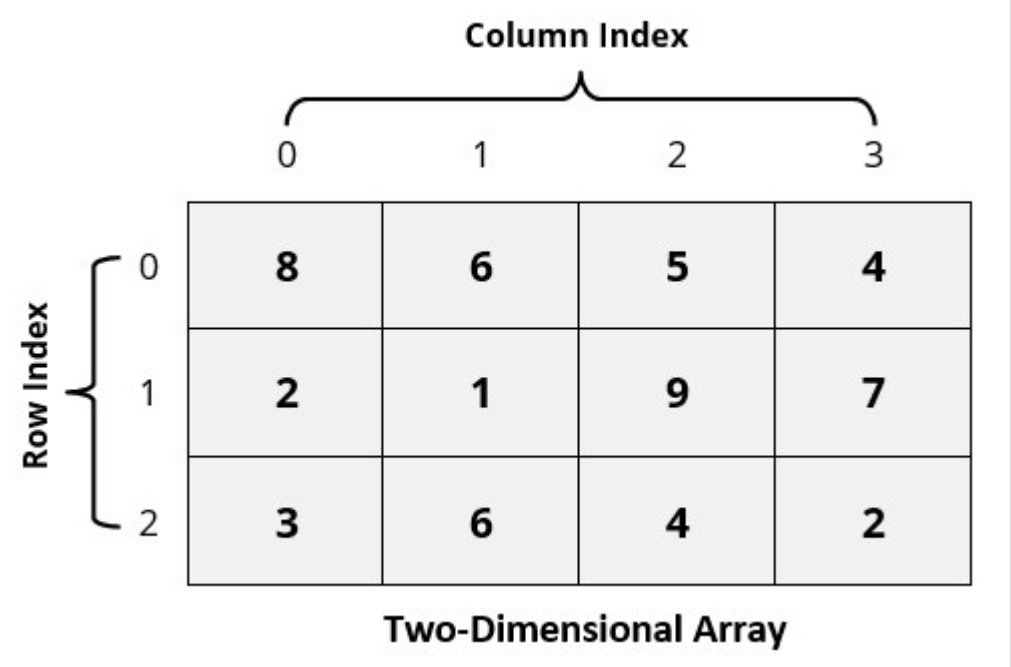
Highlights

- arrays in functions

```
double arr[5];
foo(arr);
```

- 2D arrays

```
int box[3][4];
// 3 rows, 4 columns
```



Arrays

Remember, arrays are memory addresses, much like call-by-reference: void f(int &x)

Thus we have to deal with two categories of variables:

Variables for values	Variables for addresses
int	call-by-reference
char	arrays
string	
...	

Array - array passing

But wait! This means the function can change the data since we share the memory address

(See: reverse.cpp)

Array - array passing

If we want to prevent a function from modifying an array, we can use const in the function header:

```
void reverse(const int word[]);
```

This also means any function called inside reverse must also use const on this array

(See: reverseFail.cpp)

Array - returning arrays

However, we do not know how to return arrays from functions (yet)

```
int[] foo(){
    int x[] = {1,2};
    return x;
} // x dies here, what are you returning?
```

syntax error

For now, you will have to pass in an array to be changed, much like call-by-reference

Sort

Let's practice arrays by sorting!

(See: sort.cpp)

Sort

Let's practice arrays by sorting!

Plan of attack:

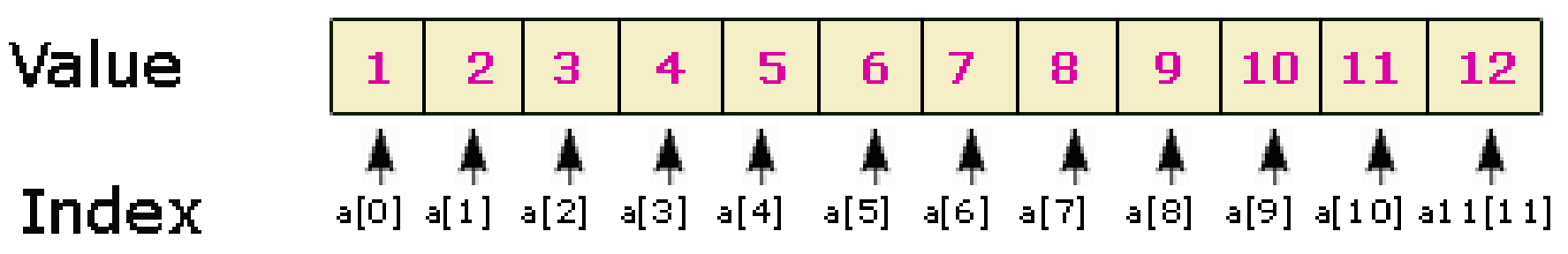
1. Make a new array
2. Find minimum element in original array and copy into new array
3. Replace minimum element in original array with the maximum element
4. Repeat 2 to 3 until done

(See: sort.cpp)

Multidimensional Arrays

So far we have dealt with simple (one dimensional) arrays

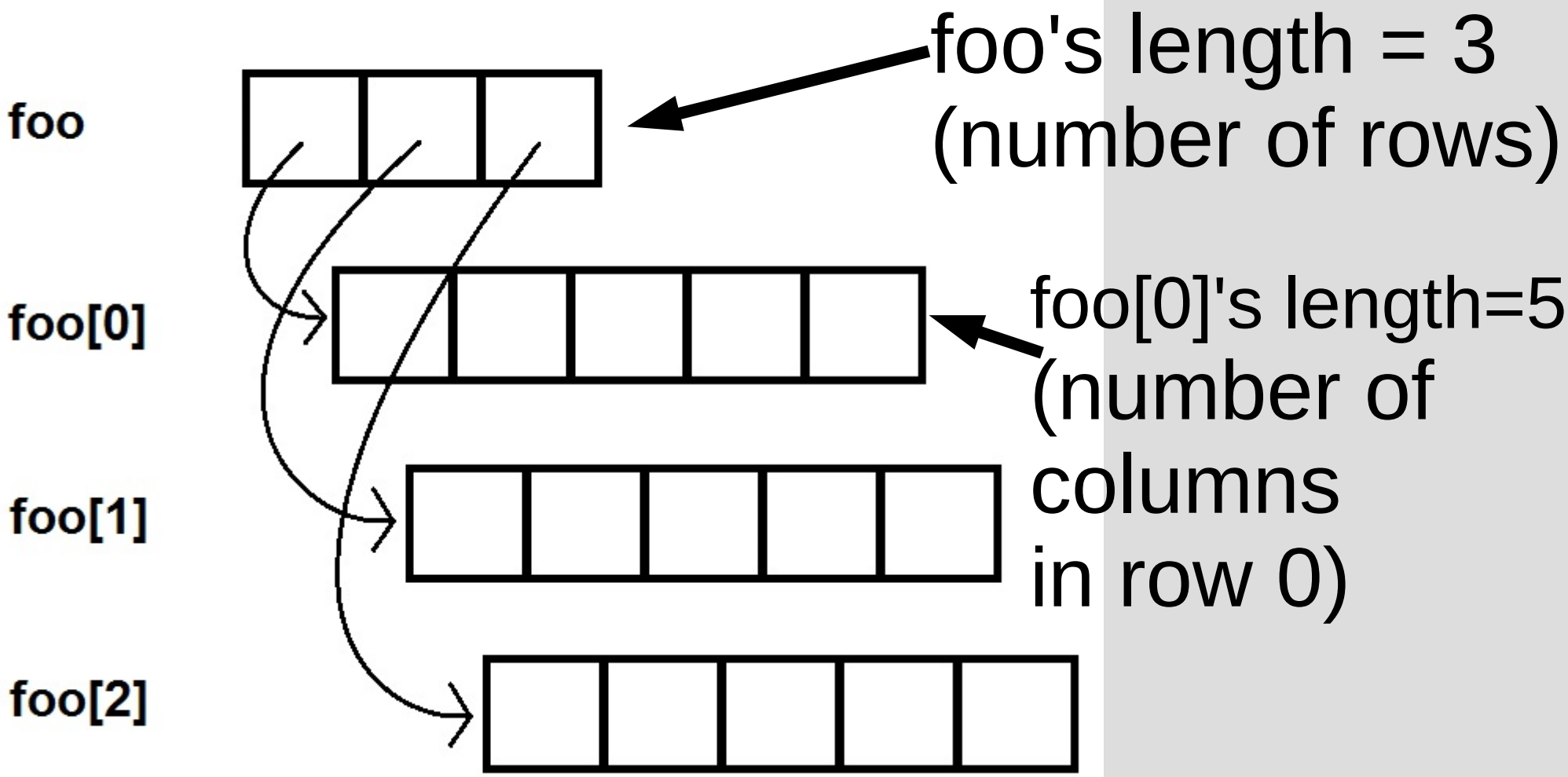
We have represented this as all the data being stored in a line



(See: lineWorld.cpp)

Multidimensional Arrays

```
int foo[][] = new int[3][5];
```



Multidimensional Arrays

If we think of a couple simple (one dimensional) arrays on top of each other...

Index	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50
5	51	52	53	54	55	56	57	58	59	60
6	61	62	63	64	65	66	67	68	69	70
7	71	72	73	74	75	76	77	78	79	80
8	81	82	83	84	85	86	87	88	89	90
9	91	92	93	94	95	96	97	98	99	100

One array for numbers 1-10

One array for numbers 71-80

(See: gridWorld.cpp)

Multidimensional Arrays

Recreate:

Index	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	11	12	13	14	15	16	17	18	19	20
2	21	22	23	24	25	26	27	28	29	30
3	31	32	33	34	35	36	37	38	39	40
4	41	42	43	44	45	46	47	48	49	50
5	51	52	53	54	55	56	57	58	59	60
6	61	62	63	64	65	66	67	68	69	70
7	71	72	73	74	75	76	77	78	79	80
8	81	82	83	84	85	86	87	88	89	90
9	91	92	93	94	95	96	97	98	99	100

(See: oneToAHundred.cpp)