

# Pre-Lab #5: Programming with Arrays

— *The last lab of Part 1!*

L2A

March 1, 2021

Prepared by: Amanda

Code samples here (uploaded after  
Tuesday L2D):

[https://github.com/amandalazar/  
apsc160-lab-lessons](https://github.com/amandalazar/apsc160-lab-lessons)

# Learning Goals

---

1. Write programs in C that use arrays
2. Write functions that take arrays as parameters

## Example 1: *What is the output of this code?*

---

```
int data[5] = {1, 2, 3, 4, 5};  
printf("%d", data[1]);
```

***Remember - C arrays use zero-based indexing!***

## Example 2: *What is the purpose of this code?*

---

```
int data[5] = {1, 2, 3, 4, 5};  
for (int i = 0; i < 5; i++) {  
    data[i] += 10;  
}
```

## Example 3: *Is there anything wrong with this code?*

---

```
double testScores[NUM_STUDENTS] = {66.0, 90.0, 85.5, 68.5, 55.0,  
                                     82.5, 45.0, 87.0, 75.0, 75.0};  
  
double sum = 0.0;  
double average;  
  
for (int i = 0; i <= NUM_STUDENTS; i++) {  
    sum += testScores[i];  
}  
  
average = sum / NUM_STUDENTS;  
printf("Computed class average: %.2lf%%", average);
```

# Example 3 Corrected

---

```
double testScores[NUM_STUDENTS] = {66.0, 90.0, 85.5, 68.5, 55.0,  
                                     82.5, 45.0, 87.0, 75.0, 75.0};
```

```
double sum = 0.0;
```

```
double average;
```

```
for (int i = 0; i < NUM_STUDENTS; i++) {  
    sum += testScores[i];  
}
```

```
average = sum / NUM_STUDENTS;
```

```
printf("Computed class average: %.2lf%%", average);
```

***Be careful to not overstep  
your array's boundaries,  
to avoid any unexpected  
behavior!***

# *Which of the following are legal C array operations?*

---

- a. Having an array as a function's return type
- b. Comparing two arrays using "=="
- c. Both a and b
- d. None of the above

# *Which of the following are legal C array operations?*

---

- a. Having an array as a function's return type
- b. Comparing two arrays using "=="
- c. Both a and b
- d. **None of the above**

*Follow-up Question #1: How could we return multiple values from a function?*

*Follow-up Question #2: How could we determine if two arrays are equivalent?*



# Great Resource on Common Mistakes with Arrays

<https://newton.ex.ac.uk/teaching/resources/jmr/6-goodbad.html>

Try stepping through the code

Hide value arrows

Next step

Value and type of last evaluated expression: **(double) 5.2**

Code

```
int main(){  
    double x=0, sides[ 3 ], y=1;  
    sides[ 1 ] = 5.2 ;  
    sides[ 2 ] = 3.1 ;  
    sides[ 3 ] = 4.6 ;// Oops!  
  
    // ... Do something  
    return 0 ;  
}
```

Memory

main()

128	129	130	131	132	133	134	135
y =	1						
136	137	138	139	140	141	142	143
sides[0] =	-3.02292746e+267						
144	145	146	147	148	149	150	151
<b>sides[1] =</b>	<b>5.2</b>						
152	153	154	155	156	157	158	159
sides[2] =	4.93359634e-270						
160	161	162	163	164	165	166	167
x =	0						

NB: the actual memory address of each variable is the address shown plus 4287852616 (0xFF937048).

# Preview of 2D Arrays

---

```
#define NUM_ROWS 3
```

```
#define NUM_COLS 4
```

```
// ...
```

```
int a[NUM_ROWS][NUM_COLS]  
    = { {0,0,0,0}, {1,2,3,4}, {2,4,6,8}};
```

# Preview of 2D Arrays

---

```
#define NUM_ROWS 3
```

```
#define NUM_COLS 4
```

```
// ...
```

```
int a[NUM_ROWS][NUM_COLS]  
    = { {0,0,0,0}, {1,2,3,4}, {2,4,6,8}};
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0] <sup>0</sup>	a[0][1] <sup>0</sup>	a[0][2] <sup>0</sup>	a[0][3] <sup>0</sup>
Row 1	a[1][0] <sup>1</sup>	a[1][1] <sup>2</sup>	a[1][2] <sup>3</sup>	a[1][3] <sup>4</sup>
Row 2	a[2][0] <sup>2</sup>	a[2][1] <sup>4</sup>	a[2][2] <sup>6</sup>	a[2][3] <sup>8</sup>

# Steps for Writing Programs

---

1. Understand the problem
2. Think through your algorithm
3. Come up with a test suite (both valid inputs and edge cases)
4. Code your algorithm
5. Test your algorithm

# Thanks for listening! 😊

Feel free to ask any Pre-Lab #5 related questions,  
or really any other questions you might have!