Week 2 Lab: Arrays

Lab Lesson

There are files that contain stubs for the programs you will be writing. You can download them from the web using the commands **`wget ds.nathanielgmartin.com/wk02_lab.zip`** and **`wget ds.nathanielgmartin.com/complexity.zip`**. The first file contains the stubs for the arrays backlog. The second contains the files for the complexity lab.

For the lab this week, we will be working on complexity. To start work on the lab, download the files from the web using wget, then edit complexity.c to write functions that will show various complexity classes. In detail:

1. Get the files using **`wget ds.nathanielgmartin.com/complexity.zip`** from a Unix terminal.
2. Where the comment **`/* put code here */`** appears, add c code that will increment the variable **`calculations`** so that it will demonstrate the complexity class of that function. You should only increment the variable calculations. That is, only use the expression **`calculations++`** to increase its size.
    1. For **`constant_time`**, add code the will return a constant number as the calculations.
    2. For **`linear_time`**, add code that will return a linear function of the input as the number of calculations.
    3. For **`quadratic_time`**, add code that will return a quadratic function of the input as the number of calculations.
    4. For **`cubic_time`**, add code that will return a cubic function of the input as the number of calculations.
    5. For **`logarithmic_time`**, add code that will return a logarithmic function of the input as the number of calculations.

The expected output appears on the next page.

```
complexity> gcc -std=c11 -o complexity complexity_completed.c
complexity> ./complexity
Constant: Input length = 0; time = 10
Constant: Input length = 100; time = 10
Constant: Input length = 200; time = 10
Constant: Input length = 300; time = 10
Constant: Input length = 400; time = 10
Constant: Input length = 500; time = 10
Constant: Input length = 600; time = 10
Constant: Input length = 700; time = 10
Constant: Input length = 800; time = 10
Constant: Input length = 900; time = 10
O(n): Input length = 0; time = 0
O(n): Input length = 100; time = 100
O(n): Input length = 200; time = 200
O(n): Input length = 300; time = 300
O(n): Input length = 400; time = 400
O(n): Input length = 500; time = 500
O(n): Input length = 600; time = 600
O(n): Input length = 700; time = 700
O(n): Input length = 800; time = 800
O(n): Input length = 900; time = 900
O(n^2): Input length = 0; time = 0
O(n^2): Input length = 100; time = 10000
O(n^2): Input length = 200; time = 40000
O(n^2): Input length = 300; time = 90000
O(n^2): Input length = 400; time = 160000
O(n^2): Input length = 500; time = 250000
O(n^2): Input length = 600; time = 360000
O(n^2): Input length = 700; time = 490000
O(n^2): Input length = 800; time = 640000
O(n^2): Input length = 900; time = 810000
O(n^3): Input length = 0; time = 0
O(n^3): Input length = 100; time = 10000
O(n^3): Input length = 200; time = 40000
O(n^3): Input length = 300; time = 90000
O(n^3): Input length = 400; time = 160000
O(n^3): Input length = 500; time = 250000
O(n^3): Input length = 600; time = 360000
O(n^3): Input length = 700; time = 490000
O(n^3): Input length = 800; time = 640000
O(n^3): Input length = 900; time = 810000
O(log(n)): Input length = 0; time = 0
O(log(n)): Input length = 100; time = 7
O(log(n)): Input length = 200; time = 8
O(log(n)): Input length = 300; time = 9
O(log(n)): Input length = 400; time = 9
O(log(n)): Input length = 500; time = 9
O(log(n)): Input length = 600; time = 10
O(log(n)): Input length = 700; time = 10
O(log(n)): Input length = 800; time = 10
O(log(n)): Input length = 900; time = 10
```

Demo

Each week, you will demonstrate the work you have been doing since the last demo. The lab teachers will use the rubric below to determine the marks you will get for the assignment. At the end of the semester, the lab class instructors will calculate the practical marks for the class by calculating the percentage of the total marks for all the demos and multiplied by the maximum practical marks for the class.

Tips

array

1. **Code must be formatted correctly**. You get marks for correctly formatting your code. The code you download is formatted correctly, so you can get 5 marks for each of the stories below by simply downloading and turning in the files.
2. **Code must compile without warnings**. You will receive marks for code that compiles. The code you download compiles without warning, so you can get 5 marks for each of the stories below simply by downloading an turing in the files.
3. **So, start early and build good habits**. You can receive 66% simply by turning in the file you download from the web, but 66% is not a good average and the percentage you gain from formatting and compiling will decrease over the semester. Easy marks are available now; later they will be harder to earn.
4. **Add 1 or two lines and make sure the formatting and compilation are correct before moving adding more lines**. In this way, you can always be sure that you can get maximum marks for formatting and compilation. It will also make it easier to program.
5. **Do the stories in order and test as you go**. The stories are arranged so you can use the earlier stories to test them. If you make sure that the early stories work; it is easier to complete the later stories.

Story 1 (15 Marks)

As a user, I want to be able to print out a 1 dimensional array of integers so I can see the results of my array algorithms.

1. Formatting is correct (5 Marks)
2. Compiles (5 Marks)
3. Runs (5 Marks)

1. Array on one line. (1 Mark)
2. Commas after each element except last (2 marks)
3. Brackets in correct place (2 marks)

Story 2 (15 Marks)

As a user, I want to enter a 1-dimensional array of integers so I can sort on them.

4. Formatting is correct (5 Marks)
5. Compiles (5 Marks)
6. Runs (5 Marks)
    1. Reads arrays (2 Marks)
    2. Good prompts (3 Marks)

Story 3 (15 Marks)

As a user, I want to be able to read and write a 1 dimensional array in a file so I can save and retrieve the array.

7. Formatting is correct (5 Marks)
8. Compiles (5 Marks)
9. Runs (5 Marks)
    1. Writes to file (2 Marks)
    2. Reads from file (3 Marks)