Lab 2 - Pointers and Arrays

Please take a minute to fill out this form:
https://goo.gl/forms/WR0vxKUJLbuu0DLd2

Introduction: About Melissa

- Senior studying Computer Science and Math
- Favorite things:
  - Programming
  - Teaching
  - Math
  - Westworld
- Email: mmgeorg@umich.edu
- Office hours (usually):
  - Monday 2 - 3pm
  - Tuesday 6 - 8pm
  - Thursday 6 - 7pm

About EECS 280 Lab

- When/ where is lab?
  - Section 24, Friday 2:30pm - 4:30pm 4153 USB
- Why come to lab?
  - Walk through worksheet solutions, time to work on the coding portion of the lab collaboratively
  - Interactive tutorials on debugging, etc.
  - Weekly lecture review
  - Written practice problems that will serve as exam prep.
  - You'll have an opportunity to ask all of your questions regarding labs, lectures, and projects
- How should I prepare for lab?
  - There are no computers in here - please bring your laptop, or contact eecs280staff@umich.edu so we can help you borrow one!
  - In the past, my students have found it useful to follow along in my slides. They will be posted every week before class in the "LabReference->Melissa George" folder.
- A few more notes:
  - I am here for YOU -- do what's best for your own learning
  - 2 halves of lab -- 1st ½ = material, 2nd ½ = 'office hours'

About EECS 280 In General

- My tips for success:
  - Keep up with lecture content
  - Start projects early
  - It’s okay to struggle
  - Google is your friend
  - Engage on Piazza - ask and answer questions
  - Practice and struggling through tough problems will make you great

- You CAN succeed in this course - I’m here to help you do that! Take advantage of lab and ask me all of your questions.

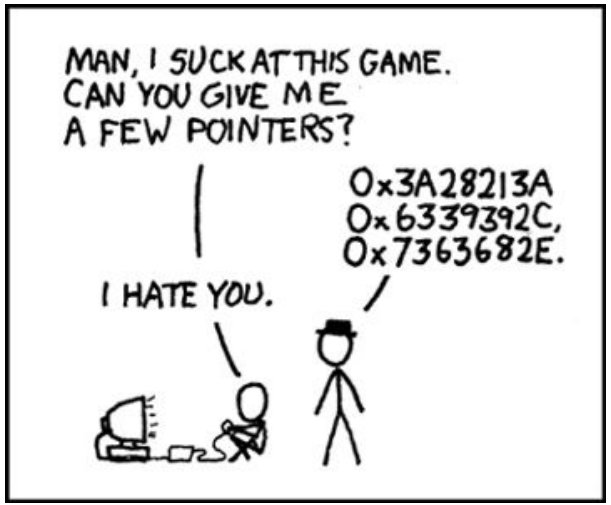
ReadMe

Reminders

- Lab 2 due Sunday, September 23rd at 8pm
- Project 2 due Thursday, October 5th at 8pm

Agenda

- Review pointers and arrays
- Lab 02 Worksheet
- Unit Test tutorial
- Main part of lab



Recall:

Where do objects live?

Objects live at specific addresses in your computer's memory. It can be useful to draw diagrams of this using stack frames-- one for each function call-- to represent what is happening in memory during your program.

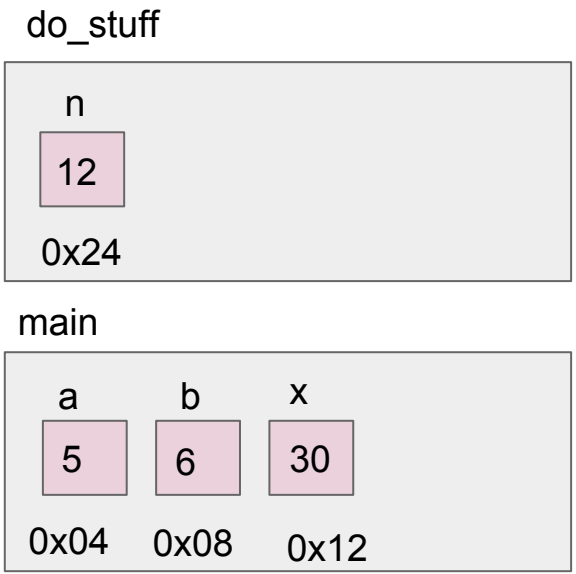
Practice: draw a diagram representing what happens in memory during the course of this program. Use a stack frame for each function call, draw boxes to represent locations where objects live in memory, and give those locations some addresses.

```
1  int do_stuff(int n) {
2      return n * 5 / 2;
3  }
4
5  int main() {
6      int a = 5;
7      int b = 6;
8
9      cout << "What is your address?\n";
10
11     int x = do_stuff(12);
12
13     cout << &x << endl;
14
15     return 0;
16 }
17
```

Recall:

Answer:

- The stack frames involved in this program would look something like this. At what point in your program should the stack look like this?

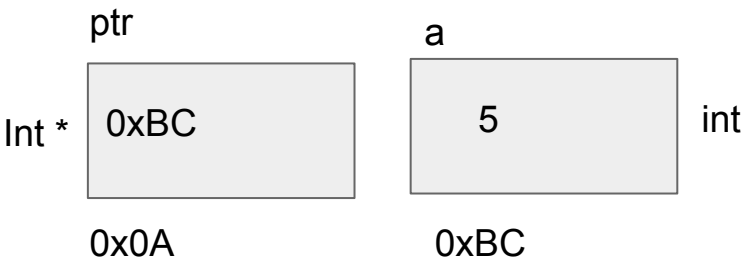



```
1  int do_stuff(int n) {
2      return n * 5 / 2;
3  }
4
5  int main() {
6      int a = 5;
7      int b = 6;
8
9      cout << "What is your address?\n";
10
11     int x = do_stuff(12);
12
13     cout << &x << endl;
14
15     return 0;
16 }
17
```

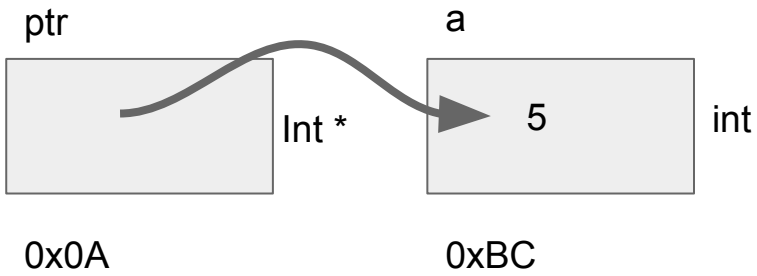
Pointers

- A pointer is a type of object that holds an address.

We can visualize pointers like this:



Or like this:



- We can declare pointers like this:

```
int a = 5;
int * ptr = &a;
```

use a * to say it's a pointer

Take the address of the int a to store in "ptr"

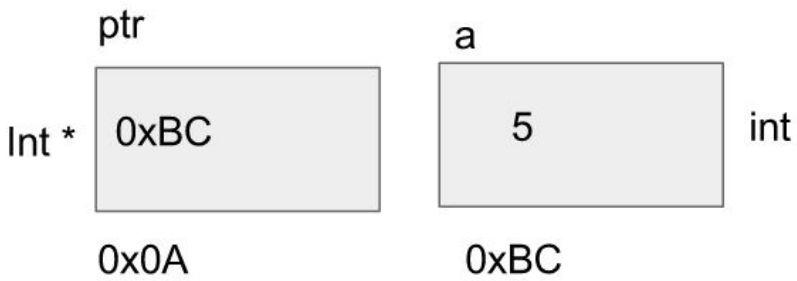
Declare the type of the thing at the end of the pointer

Pointers Continued

- When we dereference a pointer, this means:
  - “follow the pointer and return the value of the object at the end”
  - “go to the address and return the value of the object stored at that address”
- Dereferencing looks like this:

```
7   int a = 5;
8   int * ptr = &a;
9
10  cout << *ptr << endl; // dereferencing
11
12  cout << ptr << endl;
13
14
```

```
15  output:
16  5
17  0xBC
```



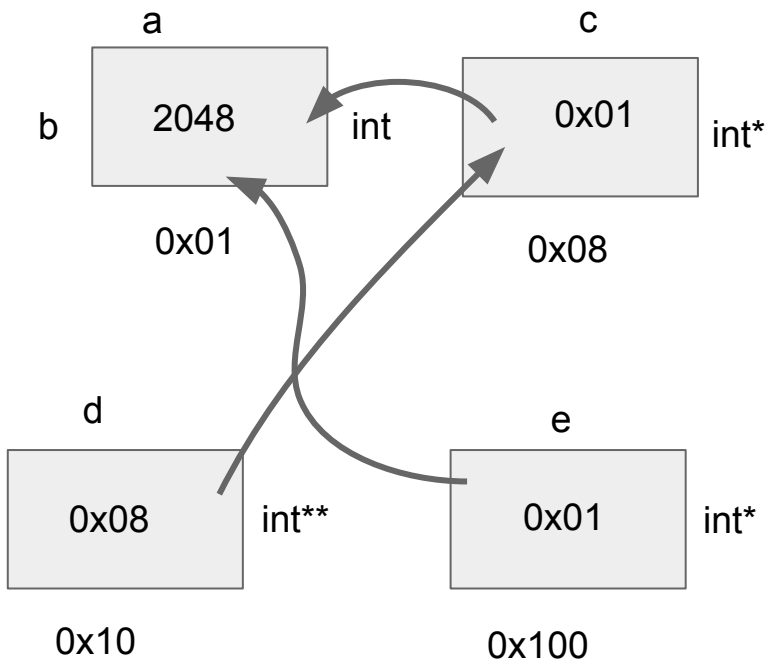
What would this look like in memory?

```
1  int a = 2048;    // a is an integer
2  int &b = a;      // b is a reference to an int
3  int * c = &a;    // c is a pointer to an int
4  int ** d = &c;   // d is a pointer to a pointer to an int
5  int * e = c;     // e is a pointer to an int
6
```

Answer:

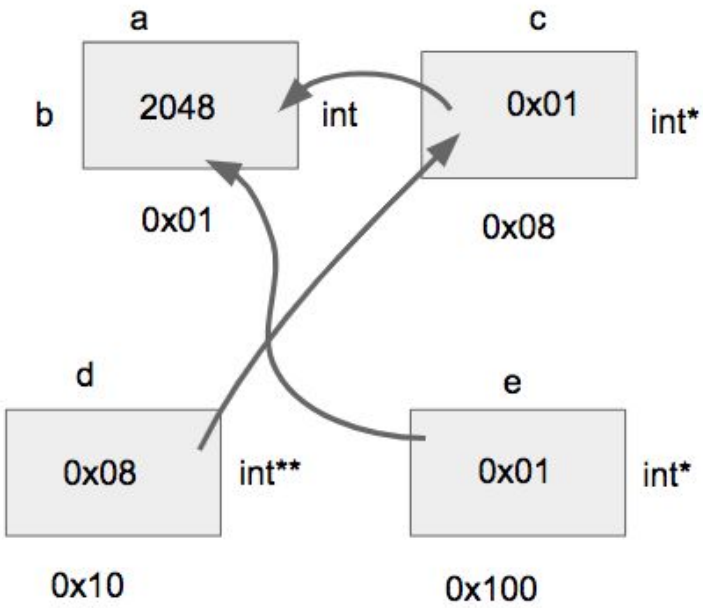
```
1  int a = 2048;   // a is an integer
2  int &b = a;     // b is a reference to an int
3  int * c = &a;   // c is a pointer to an int
4  int ** d = &c;  // d is a pointer to a pointer to an int
5  int * e = c;    // e is a pointer to an int
6
```

- What do these lines do?
  - cout << *c << endl;
  - cout << *d << endl;
  - cout << d << endl;
  - cout << b << endl;
  - cout << &a << endl;
  - cout << c << endl;
  - cout << &e << endl;
  - cout << e << endl;
  - cout << *e << endl;
  - cout << **d << endl;



Answer:

```
1  int a = 2048;   // a is an integer
2  int &b = a;     // b is a reference to an int
3  int * c = &a;   // c is a pointer to an int
4  int ** d = &c;  // d is a pointer to a pointer to an int
5  int * e = c;    // e is a pointer to an int
6
```

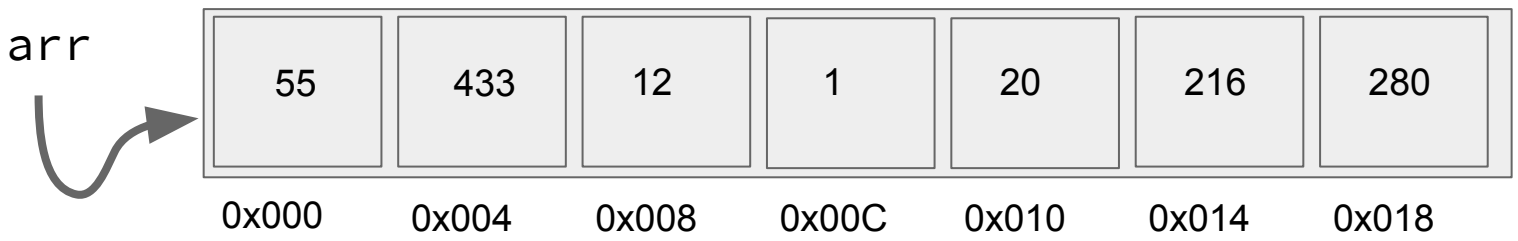


- What do these lines do?
  - cout << *c << endl; → 2048
  - cout << *d << endl; → 0x01
  - cout << d << endl; → 0x08
  - cout << b << endl; → 2048
  - cout << &a << endl; → 0x01
  - cout << c << endl; → 0x01
  - cout << &e << endl; → 0x100
  - cout << e << endl; → 0x01
  - cout << *e << endl; → 2048
  - cout << **d << endl; → 2048

Arrays

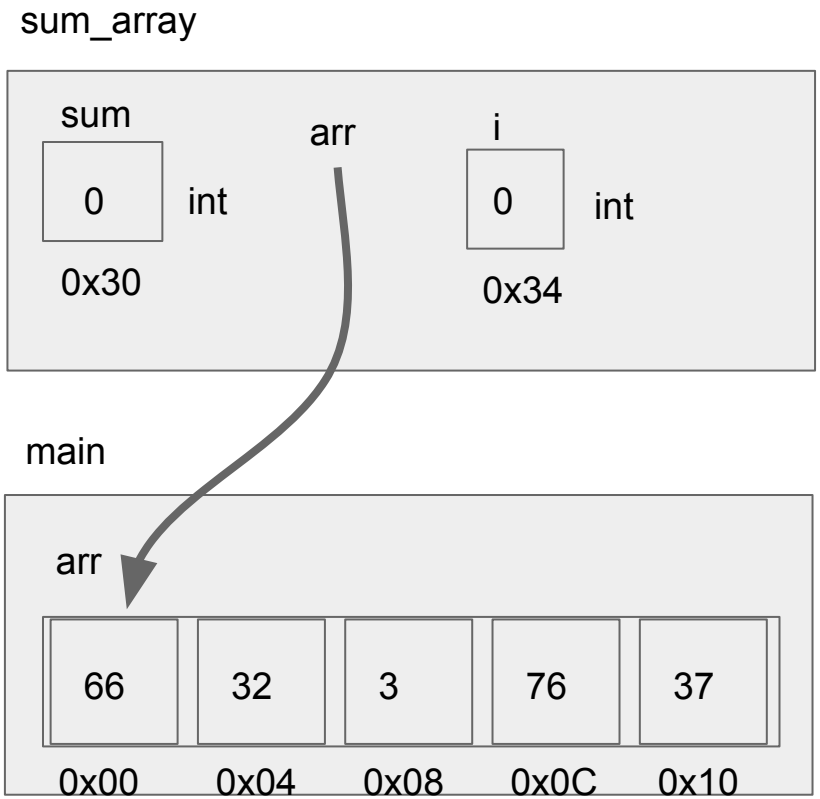
- What are arrays?
  - Arrays are contiguous blocks of memory that contain objects of the same type
  - “Contiguous” means “immediately next to one another”
  - Question -- what’s so special about contiguous memory?
  - Arrays look like this in memory:

```
int arr[7] = { 55, 433, 12, 1, 20, 216, 280 }
```



*Note: all of these addresses are “4 apart” - that’s because an integer takes up 4 bytes of memory (usually)

Passing arrays into functions




```
1  int sum_array(int size, int arr[]) {
2
3      int sum = 0;
4
5      for(int i = 0; i < size; ++i) {
6          sum += arr[i];
7      }
8
9      return sum;
10 }
11
12 int main() {
13
14     int arr[5] = {66, 32, 3, 76 37};
15
16     cout << sum_array(5, arr) << endl;
17 }
```

Next:

- Worksheet in small groups & as a class
- Unit Test Framework
  - Notes:
    - Find tutorial here: https://eecs280staff.github.io/unit_test_framework/
- Work on lab in small groups
  - Lab:
    - https://eecs280staff.github.io/lab/lab02/

Reminder : Get started on Project 2!

deflates slowly