CS 161
Introduction to CS I
Lecture 16

- Review references, pointers
- Review static and dynamic memory
- Structured data: 1-dimensional arrays

Oregon State University
College of Engineering

Week 6 tips

- Early reports indicate that lab 6 is lengthy
  - Provides additional practice with pass-by-reference, passing pointers, using pointers, and dynamic memory
  - You will get to check off more than 3 points next lab (if needed)
  - You can do it!
  - For more good practice, come to Thursday's **study session**
  - Reminder: submit your lab files on TEACH (required)
- PythonTutor – useful visualization tool
  - You'll need to #include <cstdlib> or <iostream> to use NULL (otherwise just use 0)

Oregon State University
College of Engineering

Week 6 tips (2)

- Assignment 4 – demo slots are 15 mins long (weeks 8 & 9)
- Reminder – no late submissions without prior approval
  - Any extension requests must come at least **24 hours before deadline** (emergencies excepted) and **with a good reason**
- Strategy
  - Submit early versions (we will use your latest submission)
  - Do your work on the ENGR servers, not locally on your laptop
  - If your program isn't 100% complete, submit anyway:
    - (1) partially complete (but compiling) program for partial credit (rather than 0)
    - (2) answers to written questions
  - If you delete your file, use the .snapshot directory to find and recover the hourly backup (practice this in advance)

Casey Patterson's study

Review: references and pointers

- Reference: an alias to some variable (permanent)
  - `int& r = s;`
  - Can assign new values to `r` (which is `s`), but cannot make `r` be an alias to another variable later
  - Must be initialized when declared
- Pointer: stores the address of some variable
  - `int* p = &s;`
  - Can change what address `r` contains (where it points to) anytime
  - Can be declared, then initialized later

Your turn: implement div_string()

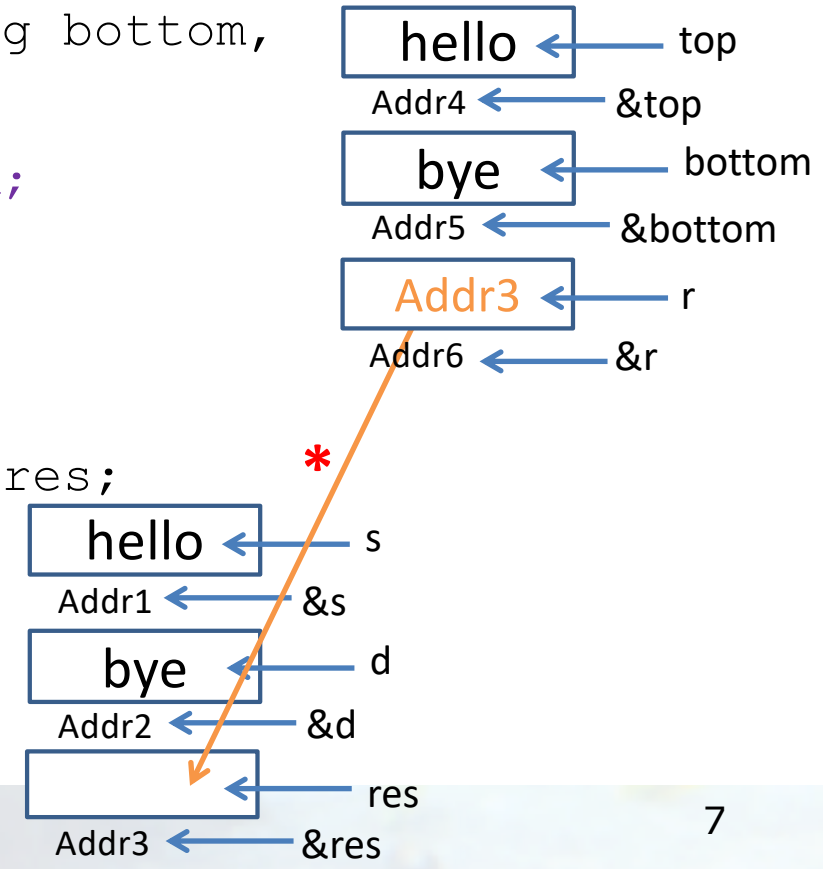
```
1. /* implement div_string() here */
2. /*  what return type? */
3. /*  what arguments?   */
4. /*  hint: what does \n do inside a string? */

5. int main() {
6.   string s = "hello", d = "bye", res;
7.   div_string(s, d, &res);
8.   cout << res << endl;
9.   return 0;
10.}
```

Pass arguments as pointers

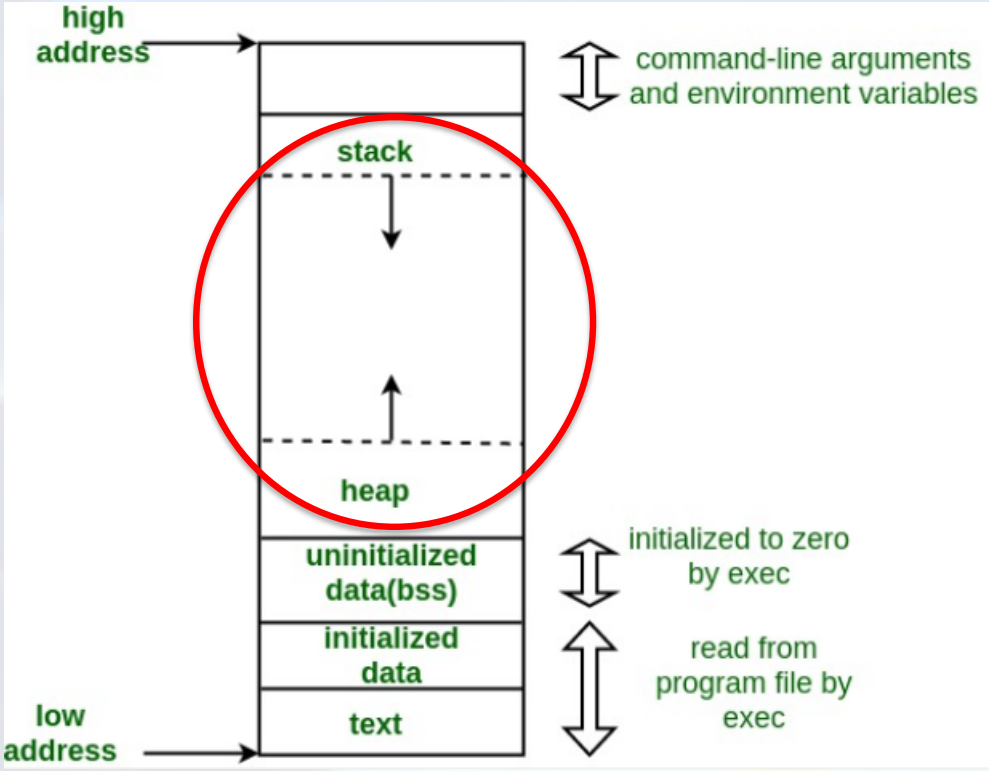
```
1. void div_string(string top, string bottom,
2.                 string* r) {
3.   *r = top + "\n-----\n" + bottom;
4. }

5. int main() {
6.   string s = "hello", d = "bye", res;
7.   div_string(s, d, &res);
8.   cout << res << endl;
9.   return 0;
10.}
```



Review: memory model

- Stack: static memory
- Heap: dynamic memory
- Why do we care about the difference?
- Heap management:
  - new (create)
  - delete (free/release)
    - doesn't delete the pointer, but instead **the memory it points to**



Your turn: On the stack or the heap?

```
1. int mercury = 5;
2. char* venus = NULL;
3. long* earth = new long;
4. int& mars = mercury;
5. short jupiter = mars + 27;
6. venus = new char;
7. int* saturn = &mercury;
8. long* uranus = earth;
```

Your turn: On the stack or the heap?

```
1. int mercury = 5;
2. char* venus = NULL;
3. long* earth = new long;
4. int& mars = mercury;
5. short jupiter = mars + 27;
6. venus = new char;
7. int* saturn = &mercury;
8. long* uranus = earth;
```

Good memory hygiene: clean up the heap

```
1. int mercury = 5;
2. char* venus = NULL;
3. long* earth = new long;
4. int& mars = mercury;
5. short jupiter = mars + 27;
6. venus = new char;
7. int* saturn = &mercury;
8. long* uranus = earth;
```

```
1. delete venus; venus = NULL;
2. delete earth; earth = NULL;
3. delete saturn?
4. delete uranus?
```

Course map

Basics
Storing data, calculations, interacting with users

Decision making (adaptation) and **repetition** (write once, repeat forever!)

Divide and conquer part 2 (recursion)

Structured data
(arrays and objects)

Divide and conquer
(modularization and code re-use in functions)

Dynamic growth
(memory allocation and management)

Oregon State University
College of Engineering

How can we compute with a lot of data?

- Imagine storing the contents of every page in a book
  - `string page_1 = "Once upon a time, ..."`
  - `string page_2 = "Further down the road, she found"`
  - `string page_3 = "They rode quickly all night, and"`
  - …
  - Very tedious!
- I want to print out each page.
  - `cout << page_1 << endl;`
  - `cout << page_2 << endl;`
  - …!

Oregon State University
College of Engineering

Array: ordered arrangement of similar items

Arrays enable easy iteration

Oregon State University
College of Engineering

```
1. string page[1024]; /* book with 1024 pages */
2. cout << page[0] << endl;  /* print page 0  */
3. cout << page[10] << endl; /* print page 10 */

4. /* Loop over all pages */
5. for (int p = 0; p < 1024; p++)
6.   cout << page[p] << endl; /* print page p */
```

Oregon State University
College of Engineering

Week 6 continues

- ❑ Attend lab (laptop required)
- ❑ Read **Rao Lesson 4** (pp. 63-71)
  C-style strings:
  https://www.cprogramming.com/tutorial/lesson9.html
  and functions: http://www.cplusplus.com/reference/cstring/
- ❑ Attend study session **Thursday, 6-7 p.m., LINC 268**
- ❑ **Assignment 4 Design** (due **Sunday, Feb. 16**)

See you Friday!

- ❑ Bring: an example of an array in real life