Lecture 04
Pointers & Arrays
February 10th, 2016

Announcements
• ACM club tonight @6pm, CSB 130
  – Total Benchmark Solutions (Healthcare Data)
  – Pizza!
• Quiz #3: 1 version of 1 question was wrong
  – Wait until tomorrow, check your grade
  – If the grade in Canvas hasn't changed, email me
• PA2 is due on today
  – Questions?
• PA3...

Memory Management
• Java (unlike C++) keeps objects on the heap
  – Objects are created by ‘new’
  – Objects live indefinitely, until
    • No more references to them exist
    • And the garbage collector gets rid of them
  – Therefore, no destructors
• C++ has no garbage collector
  – More efficient
  – Permits real-time processing
  – The #1 source of errors in C++
    • Memory leaks
    • Invalid references/pointers

Memory Management Strategy #0
• Use local variables, avoid pointers
  – Conctructed when they enter scope
  – Destructed when they leave scope
• No memory leaks are possible
  – Memory reclaimed when function returns
• Invalid references almost impossible
  – Unless you return an invalid reference

Limits to Strategy #0
• Limited scope
  – Local variables only live as long as the
    method/function that declares them
• Fixed size
  – The size of the stack frame must be known at compile time
  – Therefore, the size of local variables must be
    known at compile time

When data is persistent or dynamic, we need another strategy

Dynamic Data
• Sometimes, the size of data is unknown at compile time
• Example from PA1/2/3
  – We know the size of Point2D at compile time
  – We know the size of a Pose
    • 25 Point2Ds
  – We don’t know how many poses per file
• Dynamic data can’t be stored on the stack
  But wait, what about a vector of Point2Ds?
Virtual Memory
- Every process gets its own virtual memory
- Half belongs to the OS
- In your process' half:
  - The stack starts at one end
  - The heap starts at the other
  - Hopefully, they never meet

The Heap
- The heap is memory under programmer control
  - Starts at the other end of VM
  - The ‘new’ operator:
    - Allocates memory on the heap
    - For objects, calls the class constructor to initialize it
    - Returns a pointer to the allocated object
  - The ‘delete’ operator:
    - Takes a pointer to an object as a parameter
    - Calls the destructor (if an object)
    - Deallocates the memory
      - Available for use by ‘new’ again
  - Memory Management: every ‘new’ needs a matching ‘delete’

Pointers
- A pointer is a typed VM address
  - As such, it is its own entity (datum)
  - It is not the same thing as the object it points to
  - It can point into the heap or the stack
  - It can point to uninitialized or unallocated memory
    - But this is almost always a bug (or malware)
  - It can point into the OS’s half of memory
    - But most attempts to use such an address will flag an error (core dump)

Pointer Data Types
- Assume I define a class called Quagga
  - Quagga is a data type
  - Quagga* is a data type
    - A pointer to a Quagga
  - Quagga** is a data type
    - A pointer to a pointer to a Quagga
  - Quagga*** is a data type
    - ... and so on
  - Quagga& is also a data type
    - But is not the same as Quagga*

Operations on pointers
- ‘new’ allocates a pointer (from the heap)
- ‘delete’ de-allocates a pointer
  - If the pointer does not point to the heap, your code will crash
  - Delete does not change the value of the pointer
  - Setting it to NULL after delete is good style
  - Deleting a NULL pointer is a no-op
- If quik is an instance of Quagga:
  - &quik returns a pointer to quik
- If qptr is a pointer to a Quagga:
  - *qptr returns the quagga pointed to by qptr
  - qptr->foo() call the foo method of *qptr
  - qptr->bar returns the bar field of *qptr
  - qptr->bar is same as (*qptr).bar

Why doesn’t Java have Pointers?
- Hint: its related to why it doesn’t have destructors...
Objects vs Variables

- Imagine the following line of code:
  Quagga* qptr = new Quagga(10);

- Two simple questions:
  - What is the data type of qptr?
    - Quagga* (pointer to Quagga)
  - What is the variable name of the Quagga that was allocated?
    - It doesn’t have one!
    - The Quagga is an object, but not a variable
    - qptr is a variable of type Quagga*