Chapter Interlude 2: Pointers, Polymorphism, and Memory Allocation

See also

These CS 200 lectures might help you out…

- Pointers
  http://edu.moosader.com/lectures/cs200/16-Pointer.mp4
- Memory management
  http://edu.moosader.com/lectures/cs200/17-Memory-Management.mp4
- Dynamic variables & arrays
  http://edu.moosader.com/lectures/cs200/18-Dynamic-Arrays.mp4
- Polymorphism
  http://edu.moosader.com/lectures/cs200/22-Polymorphism.mp4

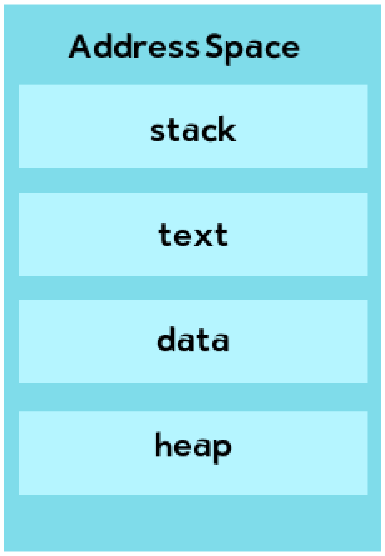
Vocabulary

Write out a description of each vocabulary term for reference later.

Term	**Description**
Early binding	
Late binding	
Pointer	
Polymorphism	
Reference	
Virtual method	

Concepts

Process address space



In general, every time you run a program, it begins a new process. If you run the program multiple times simultaneously, there will be multiple processes of the same program. A process has address space, which includes the **stack** – a place where local variables' memory is allocated, the **heap** – a place where dynamic variables' memory is allocated, the **text** – where the program is stored, and the **data** – where global and static variables' memory is allocated.

This is the sort of thing you'd learn from a Computer Architecture or Operating Systems class, but it can be handy to know when talking about dynamic memory.

(Note: The book calls the heap the **free store**, the stack the **run-time stack**, the data **static storage**, and the text **code storage**)

Types of memory errors

There are several types of memory errors that can occur once you're working with pointers. Using the slides from the Memory Management lecture, describe the following:

Error type	**Description**
Invalid memory access	
Memory leak	
Missing allocation	
Uninitialized memory access	
Dangling pointer	

Dynamic memory allocation

You can allocate memory for **dynamic arrays** and for **dynamic variables** by using pointers. The syntax is a little different for each of these.

Allocate memory for a dynamic variable:

Free memory for a dynamic variable:

Allocate memory for a dynamic array:

Free memory for a dynamic array:

Polymorphism

You can use polymorphism by taking advantage of class inheritance, pointers, and late-binding via virtual methods. The idea behind polymorphism is that the base class will define some *interface* that all subclasses will follow. Because they'll share a common interface, we can treat any of the subclasses as the base class (via a pointer). The program will figure out which subclass' functions to actually call during run-time, but from the code side, it's essentially working with pointers to the base class.

Confused? It can take some extra studying to get used to. Make sure to watch the video lecture.