Pointers and Dynamic Memory

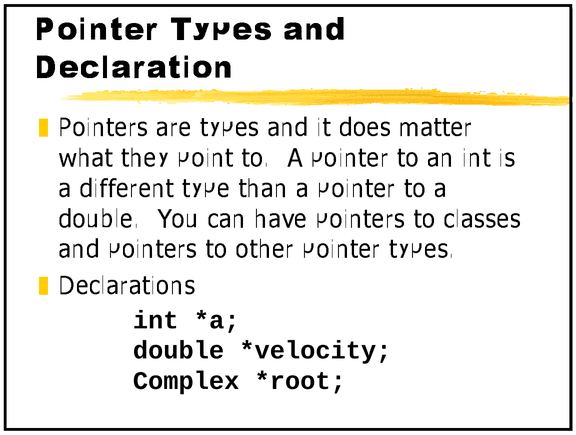
11 5 2001

Opening Discussion

- What did we talk about last class?
- Programming contest.
- Assignment #4 was fairly challenging. Completing it should make you feel good about your programming abilities. Problems 5 and 6 will be a fair bit easier (in ways).
- From reading chapter 9, tell me what a pointer is? What can you do with them?

What is a Pointer?

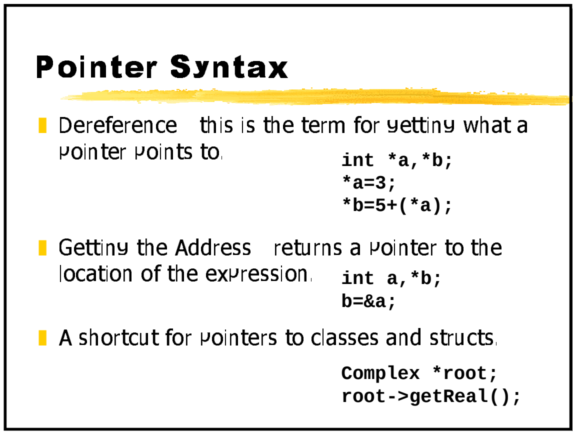
- Memory in a computer is laid out in a linear fashion and locations in it are specified by numbers. If you think about the arrows I ve drawn on the board that is what a pointer is.
- Pointers allow you to dynamically get memory as a program executes. They also allow you to create recursive data structures.



Pointer Types and Declaration

- Pointers are types and it does matter what they point to. A pointer to an int is a different type than a pointer to a double. You can have pointers to classes and pointers to other pointer types.
- Declarations

```
int *a;
double *velocity;
Complex *root;
```



Pointer Syntax

- Dereference this is the term for getting what a pointer points to.

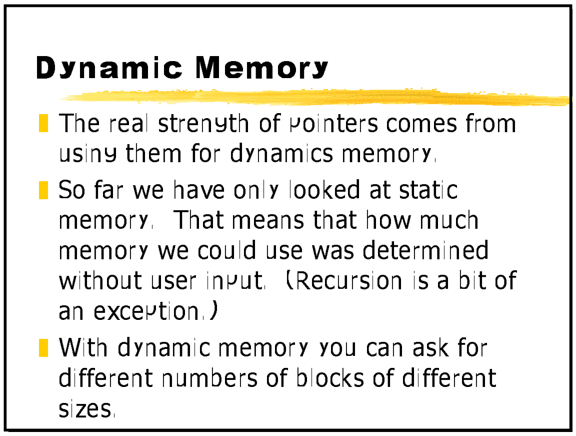
```
int *a,*b;
*a=3;
*b=5+(*a);
```

- Getting the Address returns a pointer to the location of the expression.

```
int a,*b;
b=&a;
```

- A shortcut for pointers to classes and structs.

```
Complex *root;
root->getReal();
```



Dynamic Memory

- The real strength of pointers comes from using them for dynamics memory.
- So far we have only looked at static memory. That means that how much memory we could use was determined without user input. (Recursion is a bit of an exception.)
- With dynamic memory you can ask for different numbers of blocks of different sizes.

Stack vs. Heap

- I have mentioned previously that function calls put variable and argument memory on a stack. It is much like the stack we discussed last class. A function call is a push and the return does a pop.
- Allocated memory comes from a different part of memory called the heap. Typically heap memory is opposite the stack memory in the section the program gets.

Memory Allocation

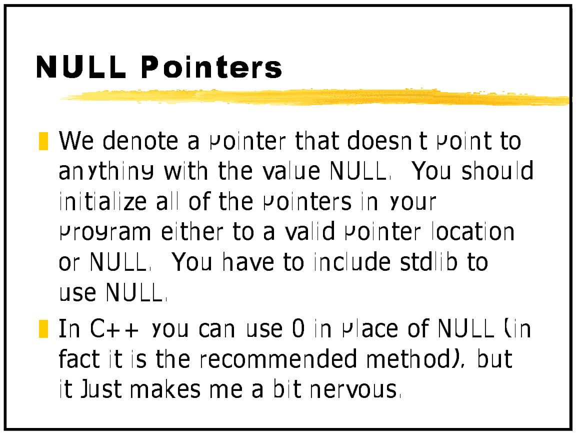
- To get these chunks of memory is what we call allocating memory. In C++ the syntax for allocating memory is to use the new operator.

```
Complex *c1=new Complex;       // default
Complex *c2=new Complex(3,5);  // with args
int n;
cin >> n;
int *a=new int[n];             // array
```

Memory Deallocation

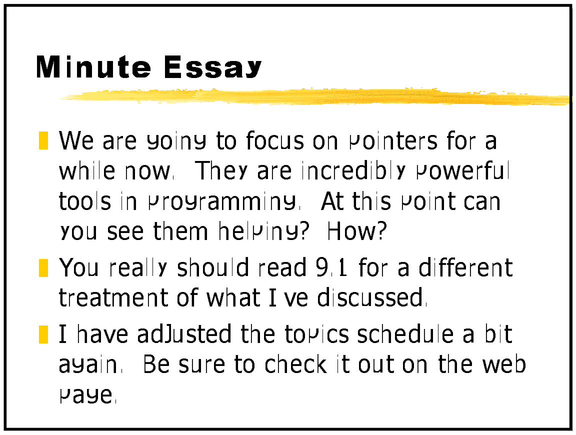
- Any memory that is allocated must be deallocated. This is done with the delete operator. Once you have deleted a pointer that memory should not be used again.

```
delete c1;      // delete single variable
delete[] a;     // delete arrays
```



NULL Pointers

- We denote a pointer that doesn t point to anything with the value NULL. You should initialize all of the pointers in your program either to a valid pointer location or NULL. You have to include stdlib to use NULL.
- In C++ you can use 0 in place of NULL (in fact it is the recommended method), but it just makes me a bit nervous.



Minute Essay

- We are going to focus on pointers for a while now. They are incredibly powerful tools in programming. At this point can you see them helping? How?
- You really should read 9.1 for a different treatment of what I ve discussed.
- I have adjusted the topics schedule a bit again. Be sure to check it out on the web page.