Pointers and memory

Ch 9 & 13.1

```
#include <stdio.h>
static char *ptr = "okay".
int main(int argc, char *argv) {
    while (argv[1] != NULL) {
        printf("\%c\n", *ptr++);
    }
}
```
Highlights

- new & delete

```cpp
int *xp;
xp = new int;
*xp = 5;
delete xp;
```
Pointers

A pointer is used to store a memory address and denoted by a * (star!)

```
int x=6;
int *xp;
xp = &x;
```

Here variable xp is a integer pointer

```
cout << *(&x);  // *(&x) same as x
```

The * goes from address to variable (much like when you hit ENTER on a url)
(See last time: pointerBasics.cpp)
Boxes

What is comes next in this pattern?

Basic programming: \texttt{int \hspace{1em} x;}
Ask for one box with a name

Intermediate programming: \texttt{int \hspace{1em} x[20];}
Ask for multiple boxes with one name

Advanced programming: ???
???
Boxes

What is comes next in this pattern?

Basic programming: \(\text{int } x;\)
Ask for one box with a name

Intermediate programming: \(\text{int } x[20];\)
Ask for multiple boxes with one name

Advanced programming: \(\text{new int;}\)
Ask for a box without giving it a name
Pointers are also especially useful to use with the `new` command.

The `new` command will create a variable (box) of the type you want. The new integer has no separate name, just part of `xp` (as array boxes part of array name) (See: `newMemory.cpp`)

```c++
int *xp;
xp = new int;
*xp = 4;
```
What does this do?

```c
int main()
{
    while(true)
    {
        int *x = new int;
    }
    return 0; //totally going to get here!
}
```
What does this do?

```cpp
int main()
{
    while(true)
    {
        int *x = new int;
    }
    return 0; //totally going to get here!
}
```

Asking for a lot of boxes there...

(See: memoryLeak.cpp)
When your program exits, the operating system will clean up your memory.

If you want to clean up your memory while the program is running, use `delete` command.

```cpp
int *imaPointer; // pointer box (holds address)
imaPointer = new int; // point here!
// do some stuff...
delete imaPointer; // goodbye pointer
```

(See: deleteMemory.cpp)
As you can manage how you want to create new variables/boxes, using `new/delete` is called **dynamic memory**.

Before, the computer took care of memory by creating variables/boxes when you use a type then deleting when the function ends.
This is also a memory leak:

```cpp
int *ptr; // make a pointer
ptr = new int; // point here
ptr = new int; // more the merrier
delete ptr; // ERASE
```

By the 3\textsuperscript{rd} line, there is no link back to the box on the 2\textsuperscript{nd} line (dangling pointer)

There should be a “delete” for every “new”
Memory management is a hard part of C++

You need to ensure you delete all your boxes after you are done with them, but before the pointer falls out of scope (see: lostPointer.cpp)

Some other languages manage memory for you
Person class

The ability to have non-named boxes allows you to more easily initialize pointers

```cpp
class person{
    string name;
    person* mother;
    person* father;
};
```

(See: personV3.cpp)
You can have multiple stars next to types:

```c
int*** x;
```

Each star indicates **how many arrows** you need to follow before you find the variable `x`.

(See: pointerPointers.cpp)