CS 368 Announcements
Wednesday, February 13, 2013

Record your attendance on the sign-in sheet.

Programming Assignment 1
- due Friday, February 15 by 10:00 pm

Last Time
- arrays
- vectors
- start Ch. 3
- variables, references, pointers
- parameter passing
- pointer basics

Today
- continue Ch. 3
- pointers to structs
- dynamic allocation
- pointers and arrays
- pointers and `const`
- pointer caveats

Next Time
- wrap up Ch. 3
- **`typedef`**
- reference variables
- C++ memory model
- pointers and parameter passing
- pointers and return values

Recall Pointers

Declaring a pointer

Assigning a value to a pointer

Accessing the pointee (location pointed to)

Recall Structures

Defining (needs to appear in the file before it is used)

```
struct Address {
  string city;
  int zip;
};

struct Student {
  int id;
  Address addr;
};
```

Using

```
Student pupil;

pupil.id = 98765;
pupil.addr.city = "Madison";
pupil.addr.zip = 53713;
```

Pointers to `struct`s (and `class`es)

Declare a pointer

Dynamically allocate space

Assign values to fields of the struct pointed to

or (alternate notation):

Dynamically deallocate space

Arrays

In C++ (and C), arrays are really pointers!

```
int a[10];

int *p = new int[10];

for (int i = 0; i < 10; i++) {
    p[i] = i;
    a[i] = 2*i;
}

int *q = a;
```

2-D Arrays

Pointers and `const`

Recall `const`

Examples:

```
const int *p
```

```
int * const p
```

```
const int * const p
```

Pointer caveats

Be careful with testing for equality

Don't dereference uninitialized pointers

Don't dereference NULL pointers

Don't dereference deleted pointers

Watch out for memory leaks