Pointers and Recursion - BSTs

Joel Willoughby

COP 3503

Outline

Agenda

- Discuss a small number of random topics
- Recursion review
- Pointer review
- Binary Search Trees

Outline

Constructors

```
1  class Test{
2    private:
3    int magic_no;
4
5    public:
6    Test(){
7      int magic_no = 0;
8    }
9
10   Test(int a): magic_no(a){}
11 };
```

- What is wrong with the above?
- What does line 10 do?

Outline

Recursion

- To walk 1000 steps, you can just walk 1 step 1000 times
- To understand recursion, you must first understand recursion
- The idea behind recursion is that some problems can be broken down into smaller and smaller sub-problems, each of which is solved the same way
- There is always a base case that ends the recursion, otherwise it loops forever

Recursion vs Iteration

- In CS, recursion presents itself as a function that calls itself (either directly or indirectly).
- Everything that can be done recursively can also be done using iteration (and a stack)
- Recursive algorithms are (almost always) slower than their iterative counterparts. This is due to the complexity of maintaining the run time stack.
- Why do we use recursion then? Many algorithms are much easier to understand when written recursively

Pointers

- Pointers are just things that point to other things
- They are made up of the address that that thing is stored at.
- Often they are used as paramters in functions to ensure changes happen to the original thing
- Often it is useful to have a pointer to nothing (indicates it isn't being used). You can do this using the NULL or nullptr keywords. nullptr is preferred (must compile with -std=c++0x)
- Their other main use is in conjunction with the new keyword

Dynamic Memory

```
1 Something * some_ptr = new Something(a, b, c);
```

- The new keyword tells the compiler to dynamically allocate some memory (as opposed to statically allocating it)
- This memory is in what is called the heap (whereas static memory is in the stack)
- After the memory is allocated, a pointer to that memory location is returned.
- It is VERY important to keep track of this pointer. If you lose where it is pointing, then that memory is said to have “leaked”
- To avoid this, we use the delete keyword:

```
1 delete some_ptr;
```

Binary Search Trees

Discussion done on the board in class...

Outline

Questions

???