Solving problems: coping with C instead of C++

- Only pass-by-value, no reference parameters

```cpp
void changeIt(int * x)
{
    *x = 3;
}
int val = 13;
changeIt(&val);
cout << val << endl;

void changeIt(int & x)
{
    x = 3;
}
int val = 13;
changeIt(val);
cout << val << endl;
```

- To make an array (allocate and pass back)

```cpp
void fillUp(int **x)
{
    *x = new int[100];
}
int * x;
fillUp(&x);
x[0] = 13;
```

what about int *x[] as param?

java: address-of operator is evil
**C++ and C-isms**

- In C++ a struct is a class in which public is default
  - supports everything a class does, including inheritance
  - in C, a struct requires the word `struct`, typedef often used

```c
typedef struct node_s
{
    int value;
    struct node_s * next;
} Node;
```

- pointers to functions: use right-left-right rule (watch parens)

```c
void foo(int (*df) (double,double));
typedef int (*iddfunc) (double,double);
void foo(iddfunc df);
```
Function objects instead of function pointers

- Problem: a function cannot maintain state that is accessible outside the function over a sequence of function calls
  - static inside the function maintains state: not accessible
  - global variables can be used, pollute the global namespace
  - solution: combine state and behavior, use a class

- Function object: a class that can be used as a function
  - overload `operator()` so that object can be used syntactically as a function [like a function pointer]
  - STL supplies these as templated functions, e.g., `less<int>`

```cpp
template <class T>
struct less : public binary_function<T,T,bool>{
    bool operator() (const T& x, const T& y){
        return x < y;
    }
};
```
When to filter entries?

- **when scandir object is constructed**
  ```
  StrFilter ccmatch(".cc"); Scandir scanner(ccmatch);
  ```

- **when DirEntry objects are loaded**
  ```
  Scandir scanner;
  scanner.load("afs/acpub/users/o/l/ola", StrFilter(".html"));
  ```

- **after DirEntry objects are loaded**
  ```
  scanner.load("/afs"); scanner.filter(ccmatch);
  ```

- **when iterating**
  ```
  Iterator<DirEntry> * iter = scanner.makeIterator(ccmatch);
  ```

- **pros/cons of different methods?**
- **What does StrFilter look like?**