Announcements

- HW due Saturday by midnight
- Lab due Sunday by midnight
Problem 3 on HW
asks for an array as input to function

Modify problem slightly:

Don't write a function

Have main ask for size of array, then input array values
for loop
    cin << myarray[i][j]

Then just print if all values are distinct.
Using File Streams - `fstream`

```cpp
#include <iostream>
#include <fstream>
using namespace std;

if file is known:
    ifstream mydata("scores.txt");
    mydata >> variable;  // my data >> variable;

if not:
    ifstream mydata;
    string filename;
    cout << "What file? ";
    cin >> filename;
    mydata.open(filename.c_str());  // parameter to open must be a C-style string
```

---

Converting to C-style string
ofstream

By default, writing to a file overwrites the file. (Think ‘w’ in Python.)

To append:

```cpp
ofstream datastream("scores.txt", ios::app);
```

'>>' in Python

to use:
```cpp
datastream << "My output is " << variable << endl;
```
There is also an `ifstream` object which allows reading and writing to a single file.

Much more complex.
String Streams

Ex: Cashing between numbers & strings.

```cpp
#include <sstream>

int age(42);
string displayedAge;
stringstream ss;
ss << age;
ss >> displayedAge;
```

- Writing out an int
- Reading a string
A note on variable scopes:

```c
int main() {
    int a;
    cin >> a;
    if (a > 0)
        int b = 12;
    else
        int b = 16;
    cout << "a is " << a << " b is " << b << endl;
}
```

A variable is destroyed as end of the control structure in which it is created.

```c
if (a > 0)
    int b = 12;
else
    int b = 16;
```

will not compile

```c
cout << "a is " << a << " b is " << b << endl;
```

will compile

```c
if (a > 0)
    int b = 12;
else
    int b = 16;
```

§ a is destroyed
int i, j

for (i = 0; j < n; i++)

if (i is gone)

must create again

for (i = 0; j < n; i++)

int a, j;

function (s ... )
Classes

What is a class?
- Has its own behaviors (functions)
- Stores collection of data

Examples: Bullseye, language Helper, Animal, Lists
Creating an instance of a class

Example:

```cpp
string s;
string greeting("Hello");
```

Never:

```
string s();
```

Why? Created a function called `s`, which (should) return a string

Never:

```
string("Hello") greeting;
```

Why? Get compiler error
class Point {
private:
    double x;
    double y;
public:
    Point( ) : x(0), y(0) { }
    // constructor

    double getX( ) const {
        return x;
    }
    // accessor

    void setX(double val) {
        x = val;
    }
    // mutator

    double getY( ) const {
        return y;
    }
    // accessor

    void setY(double val) {
        y = val;
    }
    // mutator

    // explicit declaration of data members
    // use anywhere in class - not in main
Classes:

1) Data – public or private – is explicitly declared, not just used in constructor.

   This is done inside the class, but not inside a function.

Why? If created in function destroyed at end of function.

Declare all data & then initialize it in constructor.
2. Constructor Function

- name: always same as class

- no return type - only function in C++

- can initialize variables in a list

```
Point( ) : .x(0), .y(0) { }
```

```
\begin{align*}
\text{Point( ) } & \Leftrightarrow \text{ Point( ) } \\& \quad \text{no return type} \\
& \Leftrightarrow \begin{cases} 
- x = 0 \\
- y = 0 \\
\end{cases} \\
\text{Point(double initialX=0.0, double initialY=0.0) : .x(initialX), .y(initialY) \{} \\
- x \quad - y
\end{align*}
```
Other differences

3. No self! Can just use -x or -y, it immediately scopes to the class attributes.

(There is a “this”, but its usage is a bit more complex.)


   In main
   
   mypoint. -x = 0; ← error
   must use get_x or set_x
Accessor versus mutator:

```cpp
double getX() const { return x; }
void setX(double val) { x = val; }
```

- `double getX() const` means accessor
- `void setX(double val)` means mutator

- `no return type`
- `const is enforced by compiler, nothing in the function can change data`
- `val`
Robust point class: add functionality

double distance(Point other) const {
    double dx = x - other.x;
    double dy = y - other.y;
    return sqrt(dx * dx + dy * dy);  // sqrt imported from cmath library
}

void normalize() {
    double mag = distance(Point());  // measure distance to the origin
    if (mag > 0)
        scale(1/mag);
}

Point operator+(Point other) const {
    return Point(x + other.x, y + other.y);
}

Point operator*(double factor) const {
    return Point(x * factor, y * factor);
}

double operator^(Point other) const {
    return x * other.x + y * other.y;
};  // end of Point class (semicolon is required)
Important things

1) \texttt{x + other} \texttt{x} <== \text{allowed only inside the class}

2) using \texttt{operator+}
   \texttt{not mypoint, operator + (other pt)}
   \texttt{mypoInt + other pt}

3) two versions of \texttt{x}
   \texttt{myPoint * 2} \rightarrow \texttt{((1,1) * 2)} = \texttt{(2,2)}
   \texttt{(1,1) * (2,2)} = 1 * 2 + 1 * 2 = y
Additional functions

(Not in the class)

// Free-standing operator definitions, outside the normal Point class definition
Point operator*(double factor, Point p) {
    return p * factor;
    // invoke existing form with Point as left operand
}

cout << "<x, y>";

Why?

// display using form <x, y>

< -x, -y >

cout << myPoint;

not a point