CS 180 - More C++

Note Title 9/8/2011

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

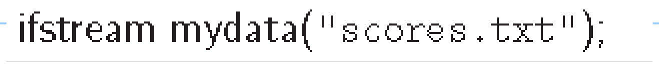
Have main ask for size of array,
then input array values
for loop
cin << myarray[i];

Then just print if all values are distinct.

Using File Streams - fstream

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

if file is known:

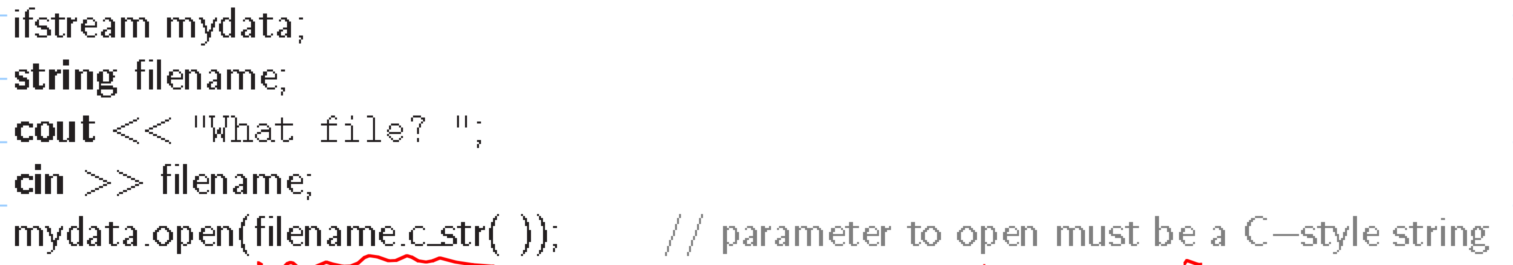


```
ifstream mydata("scores.txt");
```

→ mydata >> variable;

mydata >> variable;

if not:



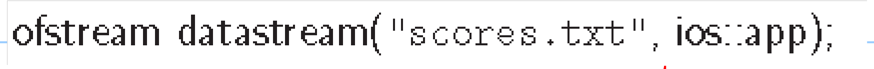
```
ifstream mydata;
string filename;
cout << "What file? ";
cin >> filename;
mydata.open(filename.c_str( ));        // parameter to open must be a C−style string
```

converts to c-style string

ofstream

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

To append:



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

'a' in Python

to use:
datastream << "My output is " << varicbb << endl;

Reading and writing

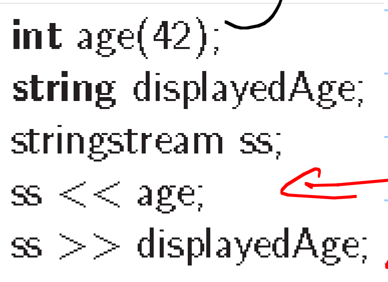
There is also an fstream object, which allows reading & writing to a single file.

Much more complex.

String Streams

#include <sstream>

Ex: Casting between numbers & strings.



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

writing out an int

reading a string

A note on variable scopes:

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

will not compile

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

int i;

for (~~int~~ i=0; .) {

} //i is gone

must create again

for (~~int~~ i = 10; . . .)

int a;

function (a . .)

Classes

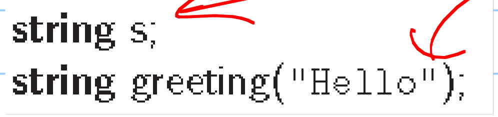
What is a class?

- Has its own behaviors (functions)
- Stores collection of data

Examples: Bullseye, language Helper, Animal, Lists

Creating an instance of a class

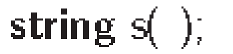
Example:



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

(" ") (calling the constructor)

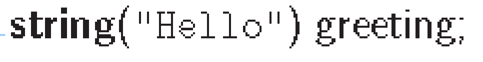
Never:



```
string s( );
```

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

Never:



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

Why? Get compiler error

Example :

```
class Point {
private:
   double _x;                       // explicit declaration of data members
   double _y;

public:
   Point( ) : _x(0), _y(0) { }      // constructor

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

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

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

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

use anywhere in class – not in main

Constructor

mypoint.getX();

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.

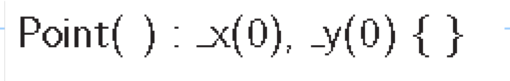
Point mypoint(a,b);

② Constructor Function

- name: always same as class

- no return type - only function in C++ w/ no return type

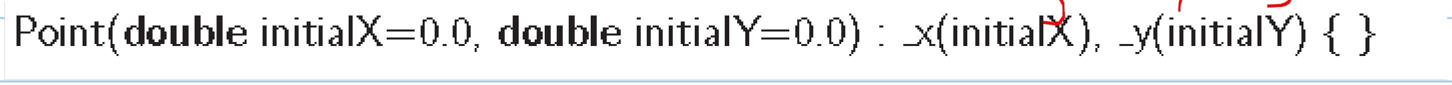
- can initialize variables in a list



```
Point( ) : _x(0), _y(0) { }
```

⟺

```
Point() {
  _x=0;
  _y=0;
}
```



```
Point(double initialX=0.0, double initialY=0.0) : _x(initialX), _y(initialY) { }
```

_x & _y

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.)

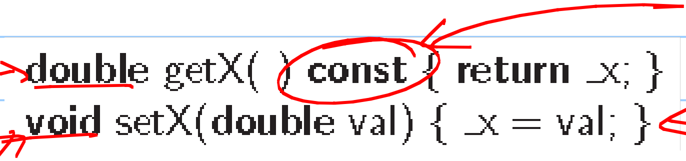
4 Access control - public versus private. enforced by compiler.

in main

mypoint._x = 0; ← error

must use getX or setX

⑤ Accessor versus mutator:



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

means accessor

mutator

no return type

const is enforeed by compiler
nothing in the function can change data.

~~val =~~

Robust Point class : add functionality

distance between 2 points:

myPoint.distance(otherPt);

```
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);
}
```

myPoint + otherPt →

```
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) _x + other._x ← allowed only inside the class

local

same

2) using operator+

not

mypoint.operator+(otherpt)

mypoint + otherpt

3) two versions of *

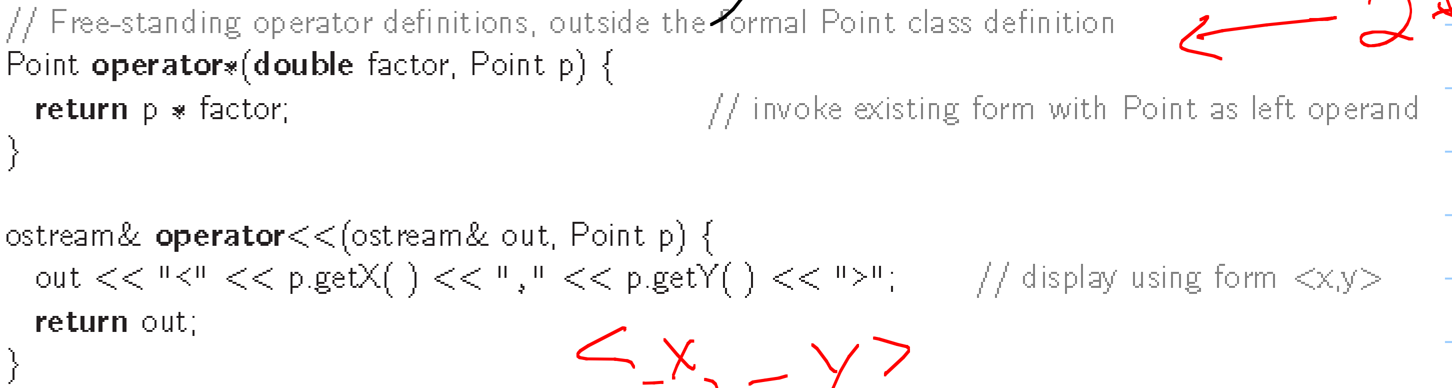
myPoint*2 → (1,1)*2 = (2,2)

(1,1) * (2,2) = 1*2 + 1*2 = 4

Additional functions

(Not in the class)

2 * mypoint



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

ostream& operator<<(ostream& out, Point p) {
  out << "<" << p.getX( ) << "," << p.getY( ) << ">";      // display using form <x,y>
  return out;
}
```

<_x, _y>

Why?

cout << mypoint;

not a point