CS2100 - Classes in C++

Note Title 9/8/2011

Announcements

- HW1 due Friday
- Look for HW2 on website soon.

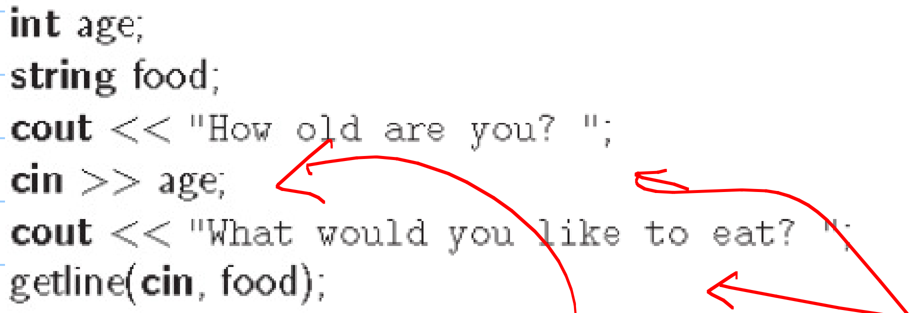
Getline

- getline is a function which saves the string up to (but not including) the next newline

Ex:

```
string person;
cout << "What is your name?";
getline(cin, person);
```

Another tricky example



```
int age;
string food;
cout << "How old are you? ";
cin >> age;
cout << "What would you like to eat? ";
getline(cin, food);
```

I type : 15
hot dogs

Problem: input steam : ~~15~~ \n hot ␣ dogs /n

Using File Streams - fstream

at top

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

if file is known:

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

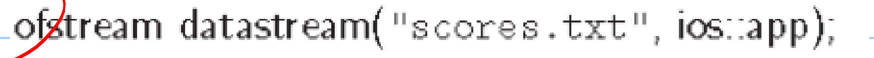
if not:

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

ofstream

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

To append:



'a'

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

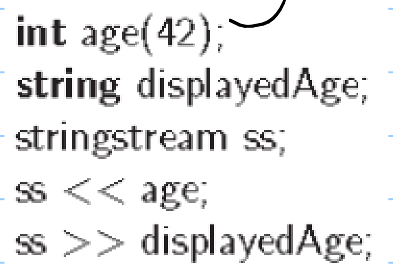
Reading and writing

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

Much more complex.

String Streams

Ex: Casting between numbers & strings.



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

age = 42

displayedAge = "42".

A note on variable scopes :

where is it valid & usable — lifetime

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

compile error

Similarly, for loops: Correct way:

```
{
  float x;
  int counter = 0;
  for (~~float~~ x = value; x > 2; x = x/2)
    {
    counter += 1;
    } // x is destroyed
                                                          OK
  cout << "The log of" << value << "is" << counter
             << endl;
  cout << "The amount left ~~was~~ "<< x << endl;
                                      error
} // counter is destroyed
   (+value)
```

Arrays as inputs to functions

Example: Write a function to specify if sum of values in an array is even.

```
bool evenSum(int anArray[], int size){
    int sum = 0;
    for (int i = 0; i<size; i++)
        sum += anArray[i];
    return ((sum % 2) == 0);
}
```

Note:

- int a[] actually makes a (the array) a pointer!
  (More on these later...)

Doesn't copy whole array but can pretend that it does - just use it like an array.

To call:

```
if (evenSum(myArray, length))
    cout << "The sum is even" <<endl;
```

Classes

What is a class?

"Container" for data &
predefined operations/methods

Creating an instance of a class

Example:

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

calling a constructor

optional input to initialize

Never:

```
string s( );
```

Why? Declaring a really dumb function

Never:

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

Why?

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;
  }
```

always capital

no self!

must explictly declare data

Constructor

no self

} (

};

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? Scope would only be in function

② Constructor Function

- name: same as object (only capital fcn you'll ever write)
- no return type ← only time
- can initialize variables via a list

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

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

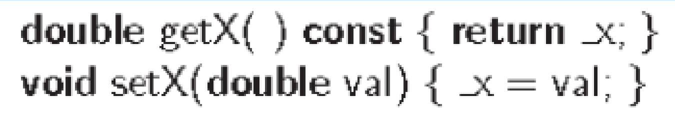
Other differences

(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!

⑤ Accessor versus mutator



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

Robust Point class : add functionality

only inside class -
otherwise : other.getX()

```
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)
```

__add__

Important things

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

2) using operator+ : in main:

Point z = x + y; ← other

actual _x + _y

3) two versions of *:

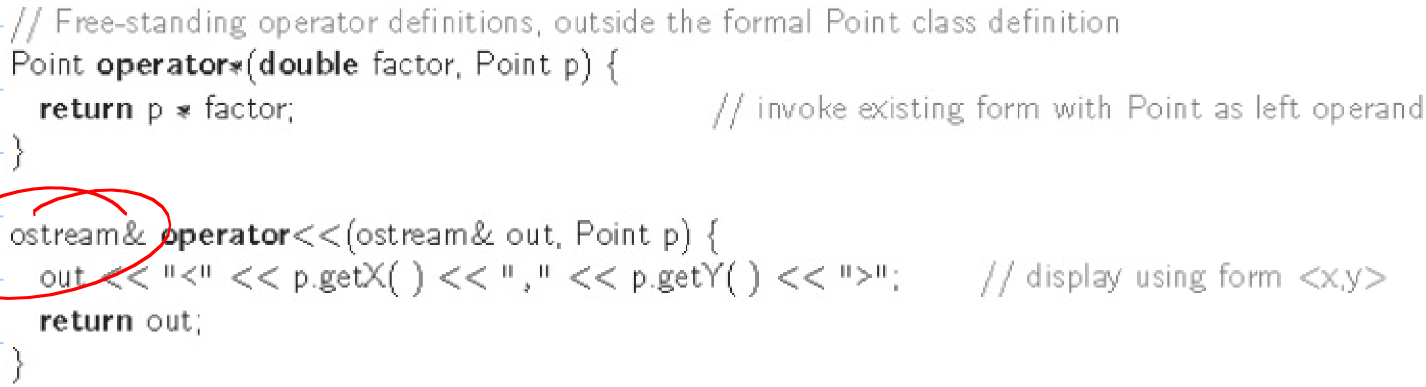
no way to return 2 different types

in Python: isInstence(x)

cout << x << y << endl;

Additional functions (Not in the class)

}; //end of Point class



```
// 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;
}
```

Why? First parameter would have to be a Point for fcn to go in class.

Separate class file : Point.h

```
class Point {
  private:
      :

  public:

};
//other functions
```

Sometimes, also have Point.cpp — hold functions