CS 162
Intro to CS II

More Classes and File I/O

Oregon State University

Odds and Ends…

- Exercise #2 due tonight
- Assignment #1 due Sunday night

Revisit Static members…

- **Static variables:**

```
class math{
public:
   static double pi;
};
double math::pi = 3.14;  //init once outside class
int main() {
   math m, m1;
   m1.pi=2.0;  //since it isn't constant, it can change
   cout << m.pi;  //changes for all members
   cout << m1.pi;
   cout << math::pi;
   return 0;
}
```

class variable

where to find

2.0

2.0

2.0

Static members…

- **Static functions**:

```
class math{
private:
  static const double p=3.14;
public:
  static const double pi() {
    return p; //can only access static members
  }
};
int main() {
  math m, m1;
  cout << m.pi();
  cout << m1.pi();
  cout << math::pi();
  return 0;
}
```

class function

no one can get to p

The Big Three…

1 constructors
2 const in appropriate places
3 preprocessor guards on header, #ifndef

responsible for hewins

- If dynamic memory allocation in class, then…
  - 4 Destructor
  - 5 Copy Constructor → shallow copy / deep copy
  - Assignment operator overload: We'll revisit this…

What is a Destructor?

- Deallocate any member variable dynamically allocated…
- What would this destructor look like then?

delete/deconstruct things on heap, when I go out of scope

has same name as class

no return type

```
string::~string() {
    delete [] s;  //delete ignores NULL
    s=NULL;
}
```

string s1("hello") string s2(s1); argument

What is a copy constructor?

- Used in pass by value
- Returning an object from a function
- Pass the class type to a constructor

very common

used in all these cases

same name

no return type

```
string:: string(const string &other) {
  len=other.len;
  if(len == 0) s=NULL;
  else {
     s=new char[len];
     for(int i=0; i<len; i++)
       s[i] = other.s[i];
  }
}
```

Main.cpp…

access.engr.orst.edu - PuTTY

```
#include "./mystring.h"
#include <iostream>
using std::cout;
using std::endl;
//We need a copy constructor for the pass by value
void fun_str(string t){
    string s("hi");

    cout << (void *)s.get_s() << endl;
    t.set_s("hi");
    //Destructor for t and s are called at the end of function
}
int main() {
    string s, s2("hello");

    cout << s.length() << endl;
    cout << s2.length() << endl;
    cout << s.at(0) << endl;
    cout << s2.at(1) << endl;

    fun_str(s2);
    cout << s2.at(1) << endl;

    //fun_str();
    //fun_str();

    return 0;
}
```

^[3,1 All

this will change s2's string without a copy constructor because it does a shallow copy.

without a copy constructor and with a destructor will core dump because s was deleted

Mystring.h

access.engr.orst.edu - PuTTY

```
1  #ifndef MYSTRING_H
2  #define MYSTRING_H
3  class string{
4     private:
5        int len;
6        char *s;
7     public:
8        string(); //default should set s to NULL and len is zero
9        string(const char *); //set to specific string and change len
10       ~string(); //destructor
11       void set_s(const char *);
12       int length() const;
13       char at(int) const;
14       char * get_s() const;
15 };
16
17 #endif
                                                    16,0-1        All
```

Mystring.cpp

access.engr.orst.edu - PuTTY

```
1 #include "./mystring.h"
2 #include <string.h> //or cstring for c-style strings
3 #include <iostream>
4 using std::cout;
5 using std::endl;
6
7 string::string(){ //default should set s to NULL and len is zero
8     s=NULL;
9     len=0;
10 }
11 string::string(const char *str){ //set to str and change len
12     len=strlen(str);
13     s=new char[len];
14     for(int i=0; i<len; i++)
15         s[i]=str[i];
16 }
17 string::~string(){
18     delete [] s;
19     s=NULL;
20 }
-- INSERT --                                    1,1           Top
```

access.engr.orst.edu - PuTTY

```
21
22 void string::set_s(const char *str){
23    if(s!=NULL)
24       delete [] s;
25    len=strlen(str);
26    s=new char[len];
27    for(int i=0; i<len; i++)
28       s[i]=str[i];
29 }
30 int string::length() const {
31    return len;
32 }
33 char * string::get_s() const {
34    return s;
35 }
36 char string::at(int i) const {
37    if(i>=len || i<0) {
38       cout << "Error!" << endl;
39       return '\0';
40    }
41    else
42       return s[i];
43 }
-- INSERT --                                    43,2          Bot
```