

# CS 162

## Intro to CS II

More Classes

# Odds and Ends...

- Sign up for Assignment #1 demo
- Assignment #2 released
- Sign up for Exercise #3 study session
- Email late Assignment #1 to me

get two weeks, but  
do this sooner  
rather than  
later

Get started early!

"Poor planning on your part  
doesn't constitute an  
emergency on my end!!!"

I will acknowledge when  
I've uploaded and  
it's ready to  
demo

# Class Type Member

```
class Point {  
public:  
    Point(); //Default Constructor  
...  
private:  
    int x;  
    int y;  
};  
class Points {  
public:  
    Point p;  
};  
int main() {  
    Points pts;  
    cout << pts.p.get_x();  
    return 0;  
}
```

notice two dots to access p's stuff.

"has a" or is "part of" relationship

- member of a class is an object

Points "has a" Point  
or  
Point is "part of" Points

# Class Type Member

```
class Point {  
public:  
    Point(); //Default Constructor  
...  
private:  
    int x;  
    int y;  
};  
class Points {  
public:  
    Points(); //Default Constructor  
private:  
    Point p;  
};  
Points::Points() : p() { }
```

need to  
call constructor  
for member object

# Revisit Copy Constructors/Destructors

```
access.engr.orst.edu - PuTTY
1 #ifndef MYSTRING_H
2 #define MYSTRING_H
3
4 class string {
5     public:
6         string(); //should set s to NULL, and set len to zero
7         string(const char *);
8         ~string(); ← destructor needed for dynamic memory
9         int length() const;
10        void modify_str(const char *str);
11        char at(int) const;
12        char *s_addr() const;
13    private:
14        char *s;
15        int len;
16 };
17
18
19
20 #endif
```

constructors }

so far we've made the destructor but no copy constructor

1,7 All

# mystring.cpp has destructor but no copy constructor

```
access.engr.orst.edu - PuTTY
1 #include "./mystring.h"
2 #include <string.h> //same as cstring
3 #include <iostream>
4 using std::cout;
5 using std::endl;
6
7 string::string() { //should set s to NULL, and set len to zero
8     s=NULL;
9     len=0;
10 }
11 string::string(const char *str){
12     len=strlen(str);
13     s = new char[len];
14     for(int i=0; i<len; i++)
15         s[i]=str[i];
16 }
17
18 string::~~string() {
19     delete [] s;
20     s=NULL;
21 }

```

default constructor, called when string str;

non-default constructor called when string str2("hello");

destructor to prevent memory leaks when object goes out of scope

"mystring.cpp" 59L, 1011C 21,1 Top

# Shallow Copy without Copy Constructor

access.engr.orst.edu - PuTTY

```

1 #include "../mystring.h"
2 #include <iostream>
3 using std::cout;
4 using std::endl;
5 void fun_str(string t) {
6     string s("hi");
7     t.modify_str("hi");
8     cout << (void *) s.s_addr() << endl;
9 }
10 int main() {
11     string str, str2("hello");
12
13     cout << str.length() << endl;
14     cout << str2.length() << endl;
15     cout << str.at(1) << endl;
16     cout << str2.at(1) << endl;
17
18     fun_str(str2);
19     cout << str2.at(1) << endl;
20     //fun_str(str2);
21 }

```

*Handwritten notes and diagrams:*

- Copy constructor called to copy str2 to t** (pointing to line 6)
- we don't have one, so shallow copy** (pointing to line 6)
- freed and sets mem** (pointing to line 8)
- s and t destructors called, which frees mem** (pointing to line 9)
- pass by value** (pointing to line 19)
- notice str2 len is 5 but s's string is 10, 1** (pointing to line 20)
- this has been freed by modify-str** (pointing to line 20)

# mystring.cpp has destructor and copy constructor

```
access.engr.orst.edu - PuTTY
11 string::string(const char *str){
12     len=strlen(str);
13
14     s = new char[len];
15     for(int i=0; i<len; i++)
16         s[i]=str[i];
17 }
18 string::~~string() {
19     delete [] s;
20     s=NULL;
21 }
22 string::string(const string &str){
23     len=str.len;
24     if(len == 0)
25         s=NULL;
26     else {
27         s = new char[len];
28         for(int i=0; i<len; i++)
29             s[i]=str.at(i);
30     }
31 }
```

it knows it is a copy constructor, when parameter is same type as class.

copy constructor called when an object is passed by value, returned from a function or called directly, ex. string str3(str)

deep copy

"mystring.cpp" 60L, 997C written 11,6 25%



# Deep Copy with Copy Constructor

access.engr.orst.edu - PuTTY

```
1 #include "../mystring.h"
2 #include <iostream>
3 using std::cout;
4 using std::endl;
5 void fun_str(string t) {
6     string s("hi");
7
8     t.modify_str("hi");
9     cout << (void *) s.s_addr() << endl;
10 }
11 int main() {
12     string str, str2("hello");
13
14     cout << str.length() << endl;
15     cout << str2.length() << endl;
16     cout << str.at(1) << endl;
17     cout << str2.at(1) << endl;
18
19     fun_str(str2);
20     cout << str2.at(1) << endl;
21     //fun_str(str2);
```

**Handwritten Diagrams:**

- Top Left (t):** A box representing a string object 't'. It has a 'len' field with value 2 (crossed out) and a 's' field with address 0x5b8. An arrow points from 's' to a memory block containing 'hello' (address 0x512).
- Top Right (s):** A box representing a string object 's'. It has a 'len' field with value 2 and a 's' field with address 0x21. An arrow points from 's' to a memory block containing 'hi' (address 0x21).
- Bottom Right (str2):** A box representing a string object 'str2'. It has a 'len' field with value 5 and a 's' field with address 0x5b4. An arrow points from 's' to a memory block containing 'hello' (address 0x10).

**Annotations:**

- A red arrow points from the 's' field of the top-left box to the 'hi' memory block, with the text "freed by modify-str" written below it.
- A red arrow points from the 's' field of the bottom-right box to the 'hello' memory block, with the text "str2 has not been changed" written next to it.
- Red text at the top left of the code area says: "t + s destructor is called to free mem".

10,1 Top