Scott’s Intro to C++
struct Time {
    int hour;
    int minute;
};

• Time data structure
• Two member variables: hour and minute
Struct Time

void init(struct Time *t, int h, int m) {
    t->hour = h;  t->minute = m;
}

void delete(struct Time *t){; /* Nothing to do */}

void show(struct Time t) {
    printf("The time is %d:%d", t.hour, t.minute);
}

void change(struct Time *t, int h, int m) {
    t->hour = h;  t->minute = m;
}

• Four functions: init(), delete(), show(), change()
• All take “struct Time t” or “struct Time *t” as a parameter
• Declare objects of type “Time” and use them
• Explicitly pass “t1” and “t2” to the functions
Class Time

class Time {
public:
    Time(void);
    Time(int h, int m); // init
    ~Time(void);       // delete
    void show(void);
    void change(int h, int m);

private:
    int hour;
    int minute;
};

• Time data structure - with functions
• Four public member functions: Time(), ~Time(), show(), change()
• Two private member variables: hour and minute
Class Time

```cpp
Time::Time(int h, int m) {
    hour = h; minute = m;
}

Time::~Time(void) { ; //nothing to do }

void Time::show(void) {
    printf("The time is %d:%d\n", hour, minute);
}

void Time::change(int h, int m) {
    hour = h; minute = m;
}

• All member functions have *implicit* access to all member variables
```
main() {  
    Time t1(1,30);  
    Time t2(3,21);  
    Time *t3;  
    t3 = new Time(0,0);  
    
    t1.show();  
    t1.change(7,15);  
    t1.show();  
    t2.show();  
    t3->show();  
    delete t3;  
}

• Declare objects of type “Time” and use them
• Don’t have to pass “t1” or “t2” to the functions