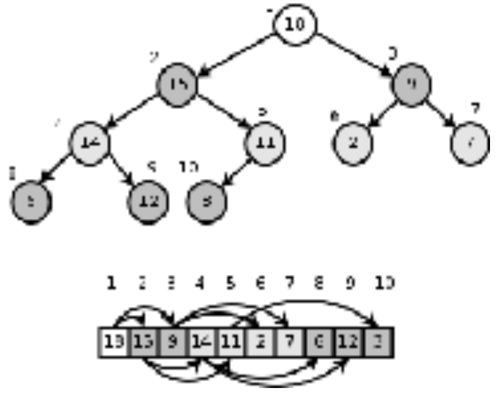
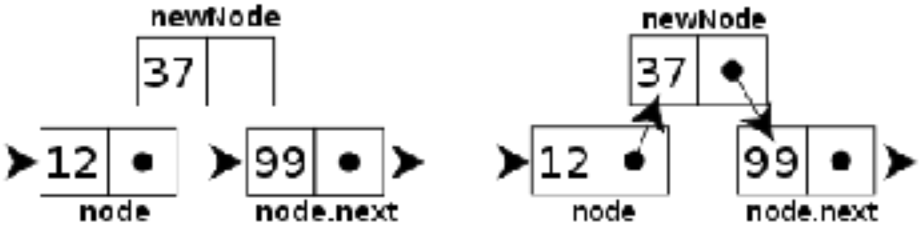
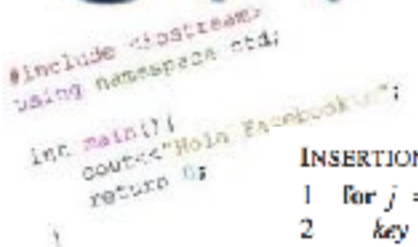




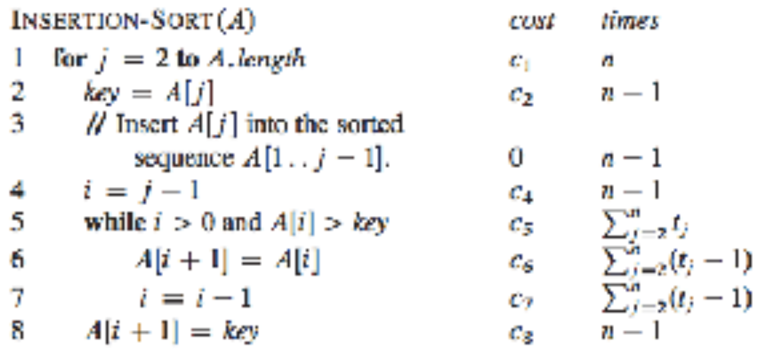
WELCOME TO CS 24!

Problem Solving with Computers-II

https://ucsb-cs24-w18.github.io/



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

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



	INSERTION-SORT(A)	*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$\quad key = A[j]$	c_2	$n-1$
3	$\quad$// Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4	$\quad i = j-1$	c_4	$n-1$
5	$\quad$**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$\qquad A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7	$\qquad i = i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8	$\quad A[i+1] = key$	c_8	$n-1$

Read the syllabus. Know what's required. Know how to get help.

Enrollment status: 110/105

About me

- Diba Mirza (diba@ucsb.edu)
  - PhD (Computer Engineering, UCSD)
  - First year as faculty at UCSB!
  - Before this: Teaching faculty at UCSD for three years
- Office hours (starting next week 1/22):
  - MW: 4p - 5p (right after lecture) at Caje Café IV
  - R: 11a-noon (HFH 1155)
  - Or by appointment
  - Check the Google calendar on course website
- You can reach me via
  - Piazza (highly recommended)
  - Email: Include [CS24] on the subject line

Ask me about:

- Course content!
- The how and why of what we are learning

Tell me about:

- Yourself!
- Experience in the class
- Interaction with the staff
- Climate of the labs

Our teaching staff - TAs !

Nidhi Hiremath

Jack Alexander

Ask the TAs about:

- Course logistics
- Regrades (labs and homeworks):
- Research and projects
- Course content
- Exceptions to late submissions because of seriour illness (with the exception of exams)

REMEMBER: You can send messages to individual staff on our class discussion forum

Structure of the staff

Instructor (me)

Course Mentors

You!

How to succeed in this course - first steps

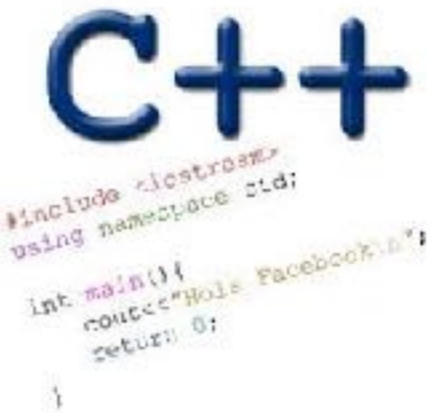
- Come to office hours and introduce yourself
- Setup a regular time to meet outside of section time with your
  - Mentor
  - Programming partner
- Communicate with the staff in person and remotely on:

PIAZZA

- Open lab hours – Phelps 3525 (starting this week):
  - Mondays: 5p - 8p (3 hrs)
  - Thursdays: noon - 3p (3 hrs)
  - Fridays: 1p -5pm (4 hrs)

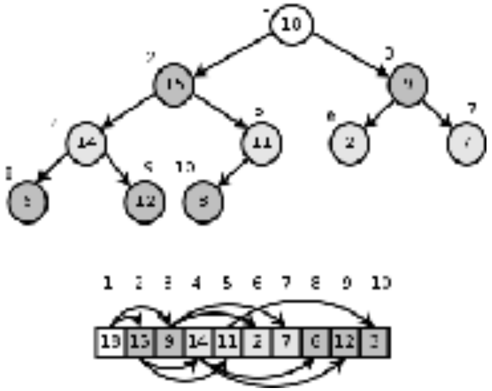
The open lab hours will be held by the TAs and tutors

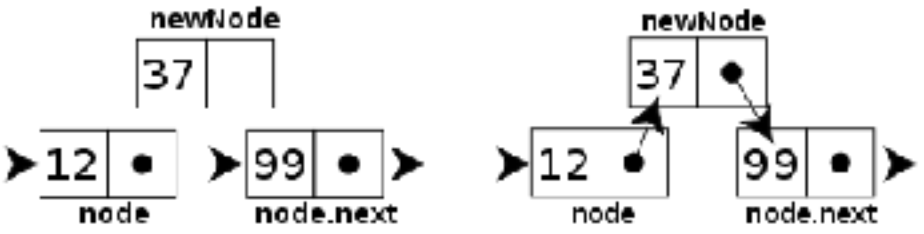
About this course, more on the course website: https://ucsb-cs24-w18.github.io/



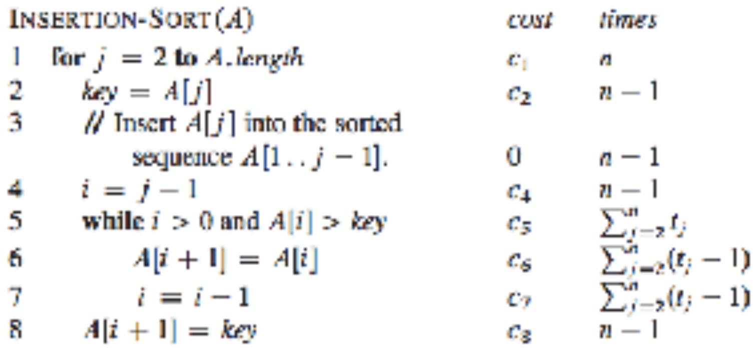
```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;

}
```

Data Structures





Complexity Analysis



```
INSERTION-SORT(A)                                   cost    times
1  for $j = 2$ to $A.length$                        $c_1$   $n$
2      $key = A[j]$                                 $c_2$   $n-1$
3      // Insert $A[j]$ into the sorted
              sequence $A[1..j-1]$.                 $0$     $n-1$
4      $i = j - 1$                                  $c_4$   $n-1$
5      while $i > 0$ and $A[i] > key$               $c_5$   $\sum_{j=2}^{n} t_j$
6          $A[i+1] = A[i]$                          $c_6$   $\sum_{j=2}^{n} (t_j - 1)$
7          $i = i - 1$                              $c_7$   $\sum_{j=2}^{n} (t_j - 1)$
8      $A[i+1] = key$                               $c_8$   $n-1$
```

Course Logistics

- Grading
  - Class and section participation (iclickers): : 2%
  - Homeworks/Quizzes (due every week) : 8%
  - Lab (programming) Assignments(due weekly) : 20%
  - Projects (programming assignments) : 20%
  - Midterm exam: : 20%
  - Final exam : 30%

- No makeups for exams. Make sure you have no scheduling conflicts with exams
- You have 48 hours grace period to submit the labs – choose wisely. DO NOT contact the instructor or TAs for extensions unless you have a real emergency
- ATTENDENCE in sections and lectures is REQUIRED!
- To complete the labs you need a college of engineering account. If you don’t have one yet, send an email to help@engineering.ucsb.edu

Clickers out – frequency AB

About you…

What is your familiarity/confidence with programming in C++?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

About you…

What is your familiarity/confidence with C++ pointers?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

About you…

What is your familiarity/confidence with C++ classes?

A. Know nothing or almost nothing about it.
B. Used it a little, beginner level.
C. Some expertise, lots of gaps though.
D. Lots of expertise, a few gaps.
E. Know too much; I have no life.

About you…

What is your familiarity/confidence with using version control – git or subversion?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

iClickers: You must bring them

- Buy an iClicker at the Bookstore
- Register it on GauchoSpace (wait for my announcement on Piazza)
- Bring your iclicker to class

Required textbook

- Michael Main and Walter Savitch. *Data Structures and Other Objects Using C++ (4th edition)*, Addison-Wesley, 2011.

Recommended textbook

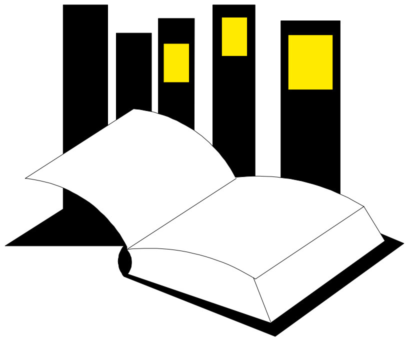
- Problem Solving with C++, Walter Savitch, Edition 9

You must attend class and lab sections
You must prepare for class
You must participate in class

Clickers, Peer Instruction, and PI Groups

- Find 1-2 students sitting near you. If you don't have any move.
- Introduce yourself.
- This is your initial PI group (at least for today)

Intro to Object Oriented Programming

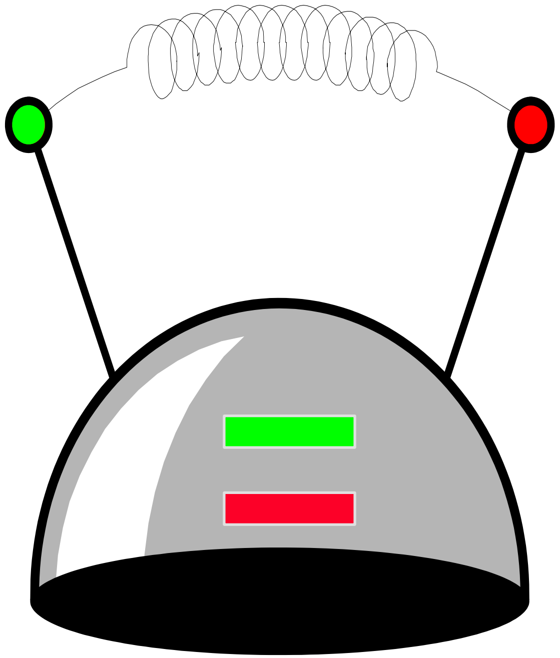


Data Structures
and Other Objects
Using C++

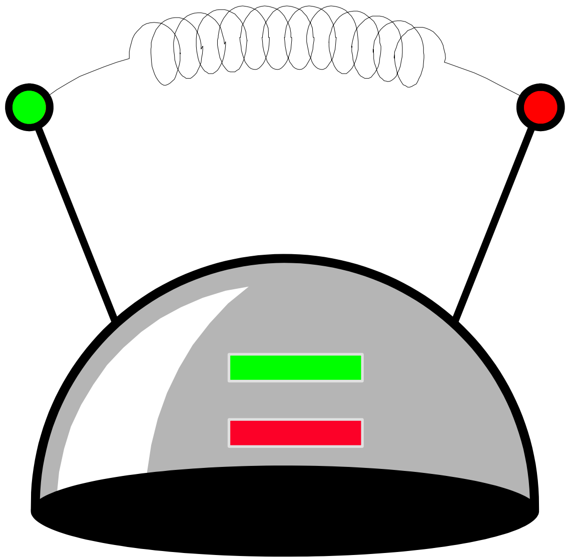
- Chapter 2 introduces Object Oriented Programming.
- OOP is an approach to programming which supports the creation of new data types and operations to manipulate those types.

What is this Object ?

- There is no real answer to the question, but we’ll call it a “thinking cap”.
- The plan is to describe a thinking cap by telling you what actions can be done to it.

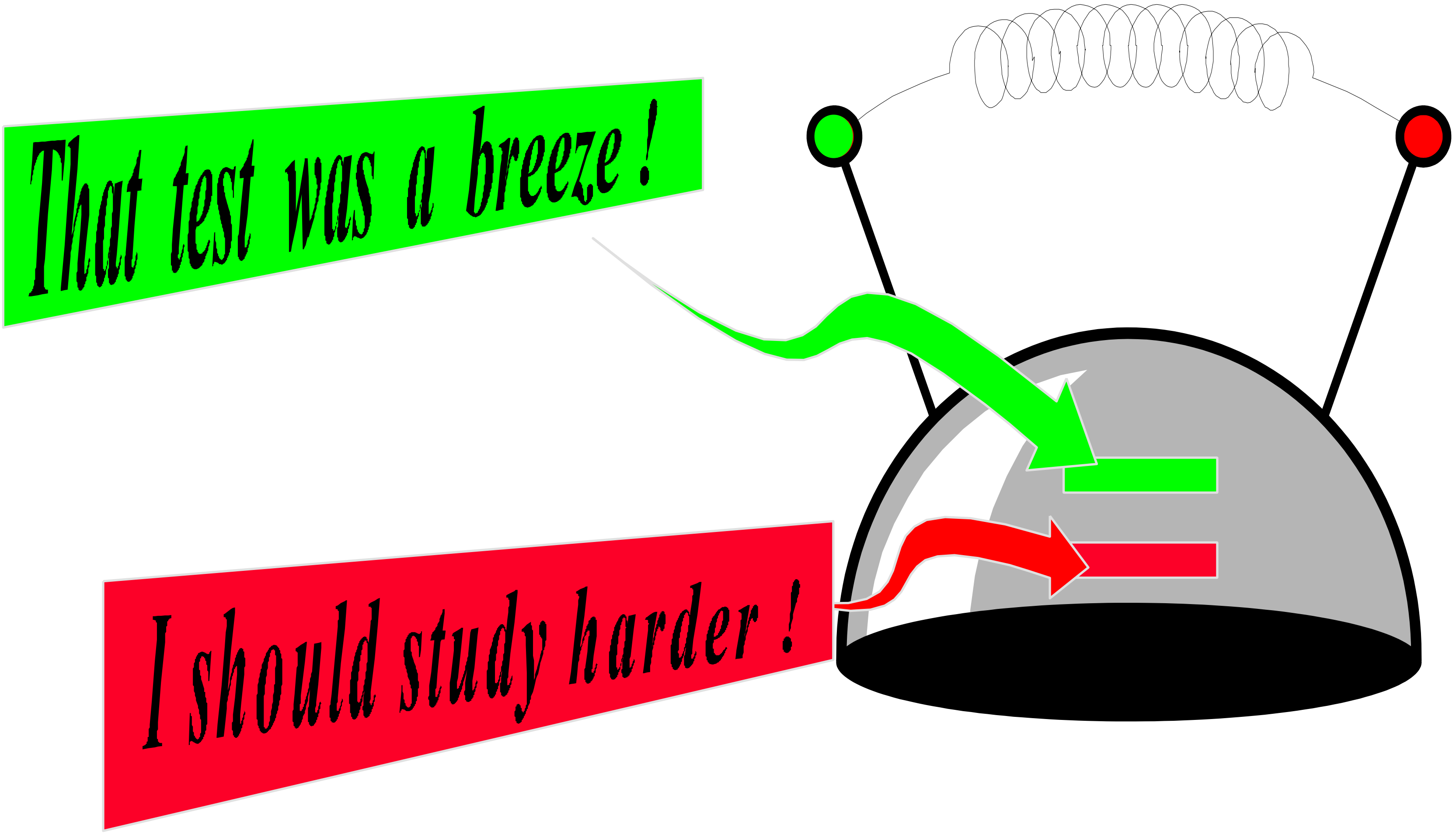


Description of the thinking cap



- You may put a piece of paper in each of the two slots (green and red), with a sentence written on each.
- You may push the green button and the thinking cap will speak the sentence from the green slot’s paper.
- And same for the red button.

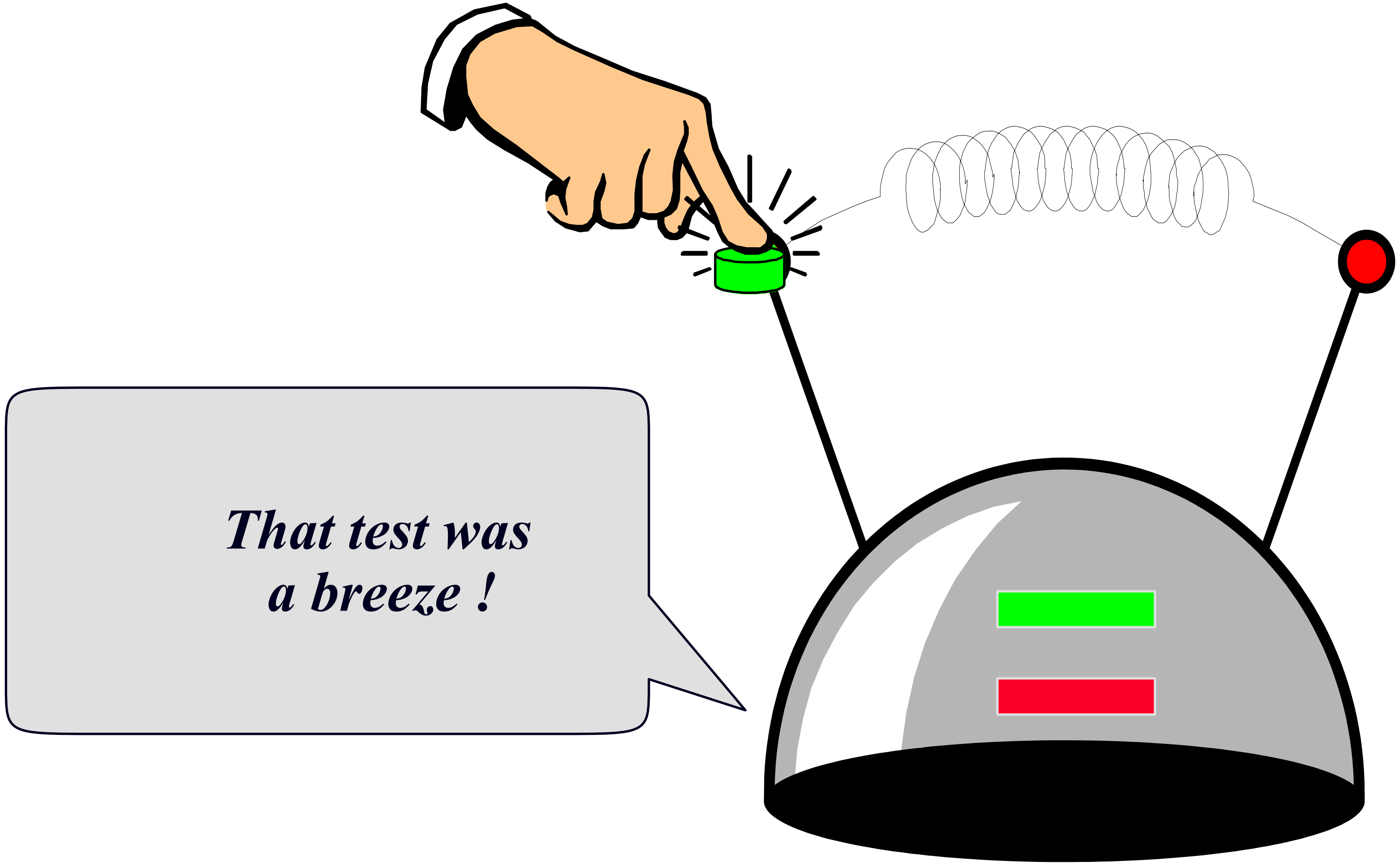
Example



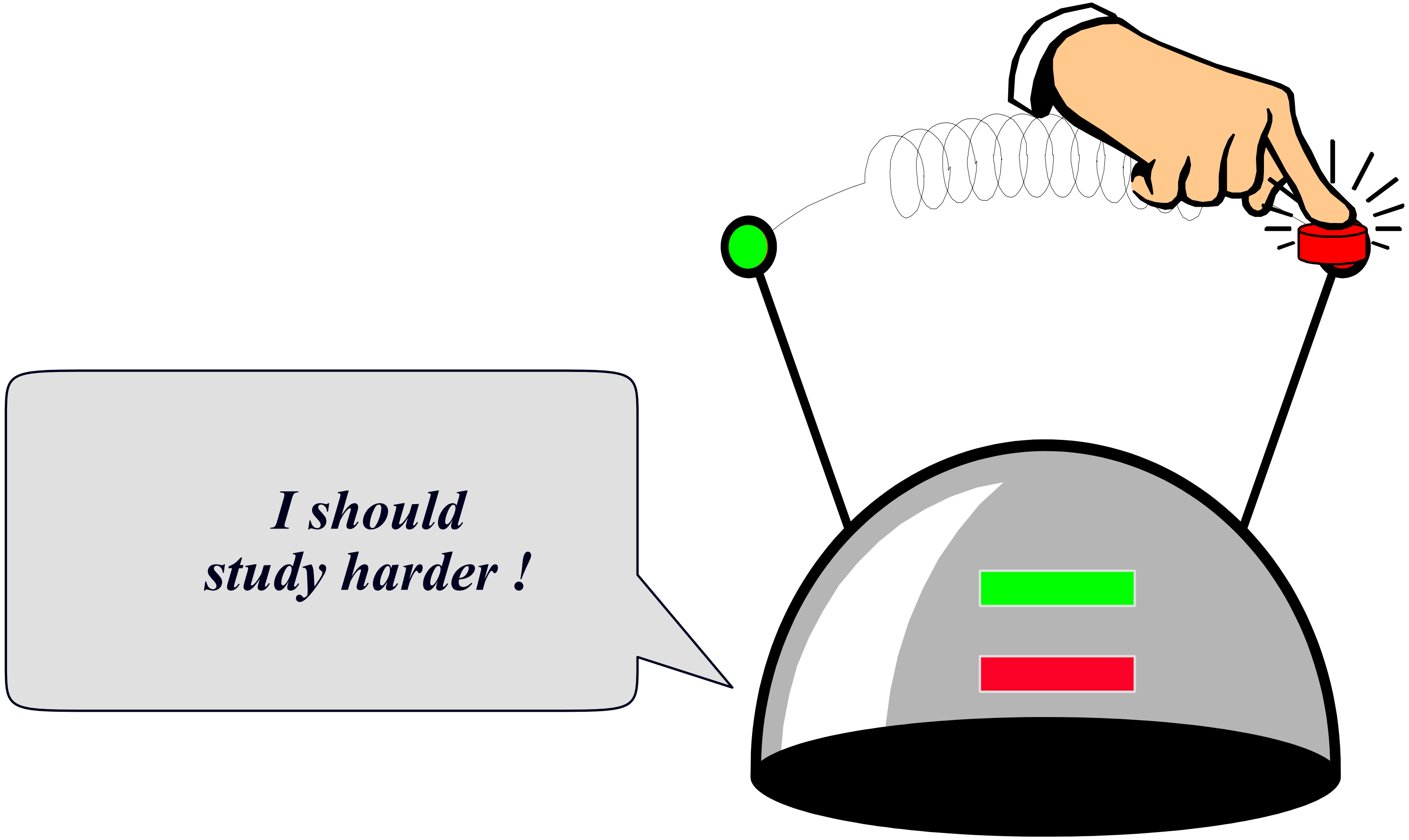
That test was a breeze !

I should study harder !

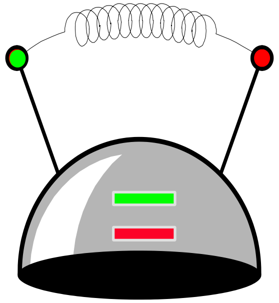
Example



Example



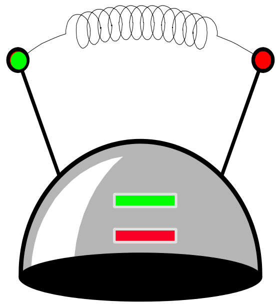
Thinking Cap Definition



- We can define the thinking cap using a data type called a class.

```
class thinking_cap
{

          . . .

 };
```

Components of the thinking cap

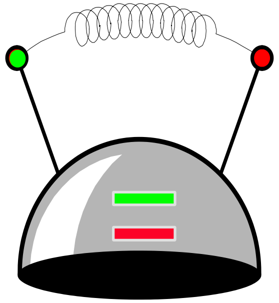


- The class will have the data that store its internal state:
- green_string and red_string..

Is it similar to a struct so far?

```
class thinking_cap
{

      char green_string[50];
      char red_string[50];
      . . .
};
```

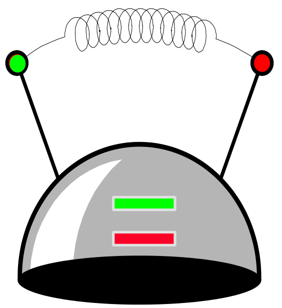
Thinking Cap as an Abstract Data Type



- The two components will be private member variables. This ensures that nobody can directly access this information. The only access is through functions that we provide for the class.

```
class thinking_cap
{

private:
    char green_string[50];
    char red_string[50];
};
```

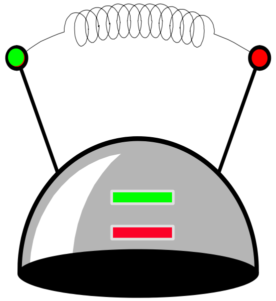
Thinking Cap as an Abstract Data Type



- Public interface – can be accessed by the user of the class
  - List member function (methods) that manipulate data here
  - Provides a clear interface to data

```
class thinking_cap
{
public:
      . . .
private:
     char green_string[50];
     char red_string[50];
};
```

Thinking Cap Implementation



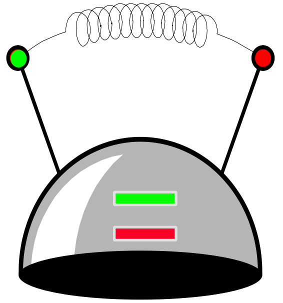
- Public interface – can be accessed by the user of the class
  - List member function (methods) that manipulate data here!
  - Provides a clear interface to data!!

Our thinking cap has at least three member functions:

```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

Function bodies will be elsewhere.

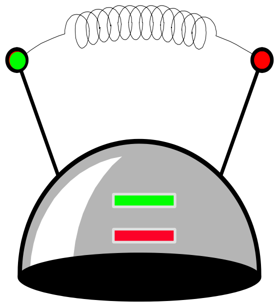
Thinking Cap Implementation



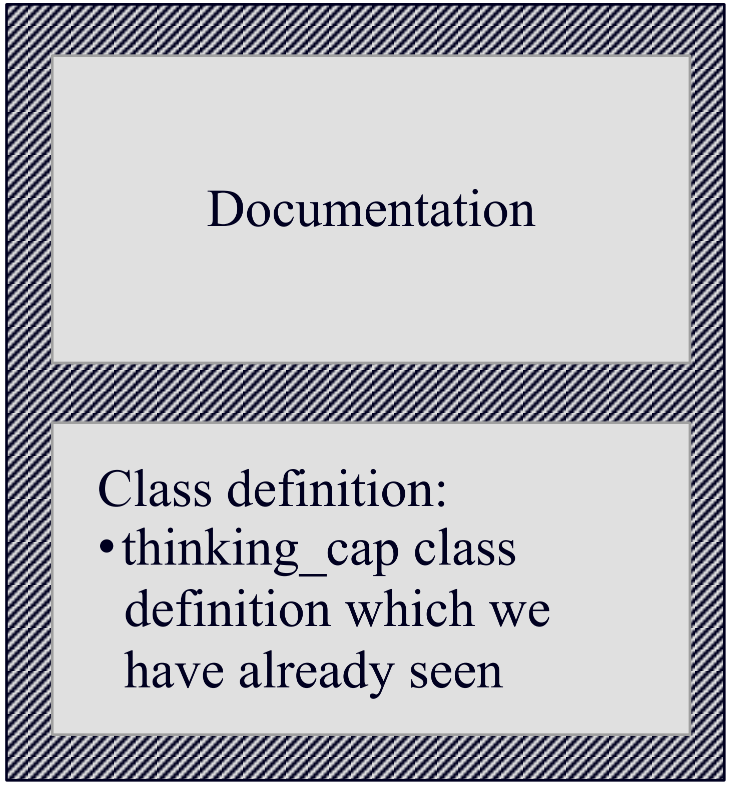
```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

This means that these functions will not change the data stored in a thinking_cap.

Files for the Thinking Cap



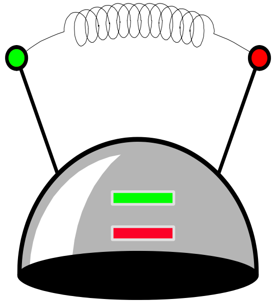
- The thinking_cap class definition, which we have just seen, is placed with documentation in a file called thinker.h, outlined here.
- The implementations of the three member functions will be placed in a separate file called thinker.cxx, which we will examine in a few minutes.



Documentation

Class definition:
- thinking_cap class definition which we have already seen

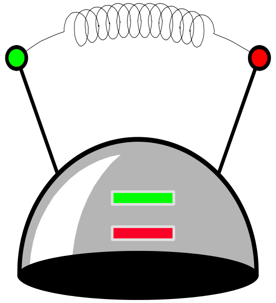
Using the Thinking Cap



- A program that wants to use the thinking cap must **include** the thinker header file (along with its other header inclusions).

```
#include <iostream>
#include <cstdlib>
#include "thinker.h"

...
```

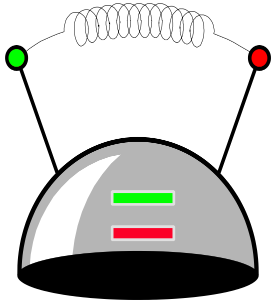
Using the Thinking Cap



```
#include <iostream.h>
#include <stdlib.h>
#include "thinker.h"

int main( )
{
    thinking_cap student:
    thinking_cap fan;
}
```

- How is student different from “thinking_cap”?

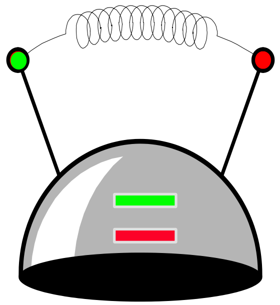
Using the Thinking Cap



```
#include <iostream.h>
#include <stdlib.h>
#include "thinker.h"

int main( )
{
    thinking_cap student:
    thinking_cap fan;
```

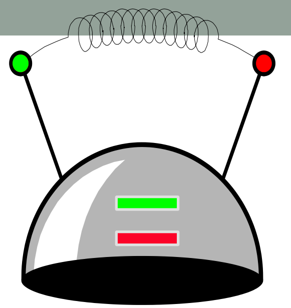
- What happens in memory after this code is executed?

Using the Thinking Cap



- Activating the student's slot method

```
#include <iostream.h>
#include <stdlib.h>
#include "thinker.h"

int main( )
{
    thinking_cap student;
    thinking_cap fan;
    student.slots( "Hello",  "Goodbye");

}
```



Quiz

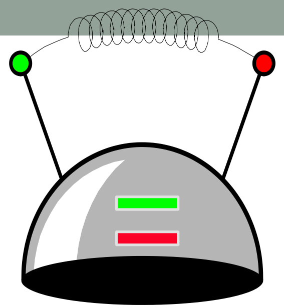
How would you activate student's push_green member function ?

(Write your answer)

(After that discuss with your peer group)

```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

```
int main( )
{
    thinking_cap student;
    thinking_cap fan;
    student.slots( "Hello",  "Goodbye");
}
```



Quiz

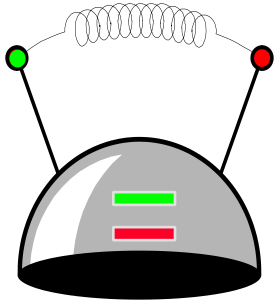
What would be the output of student's push_green member function at this point in the program ?

```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

```
int main( )
{
   thinking_cap student;
   thinking_cap fan;
   student.slots( "Hello",  "Goodbye");
   student.push_green();
}
```

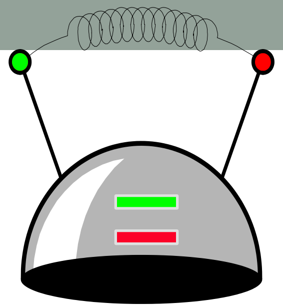
Quiz

```
int main( )
{
    thinking_cap student;
    thinking_cap fan;
    student.slots( "Hello",  "Goodbye");
    fan.slots( "Go Cougars!", "Boo!");
    student.push_green( );
    fan.push_green( );
    student.push_red( );
}
```



Trace through this program, and write the complete output.

Review: Thinking Cap Definition



```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( );
    void push_red( );
private:
    char green_string[50];
    char red_string[50];
};
```

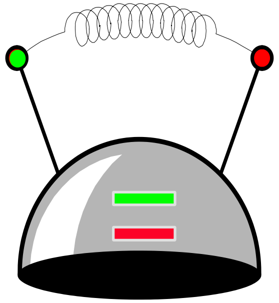
When are the data members (green_string and red_string) created in memory

A. When the compiler compiles the class definition (above)

B. When an object of type thinking_cap is created in the program (at run-time)

C. When the slots() member function is activated

Thinking Cap Implementation

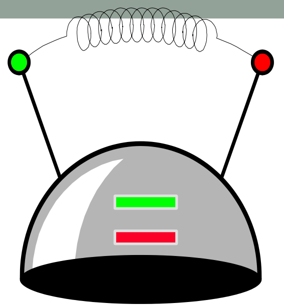


- Usually we implement the class in a separate .cpp file.

```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( );
    void push_red( );
private:
    char green_string[50];
    char red_string[50];
};
```

Function bodies will be in .cxx file.

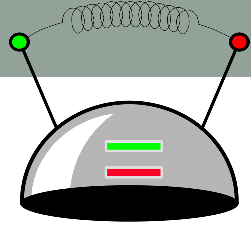
Thinking Cap Implementation



There are two special features about a member function's implementation . . .

```
void thinking_cap::slots(char new_green[ ], char new_red[ ])
{




}
```

Thinking Cap Implementation

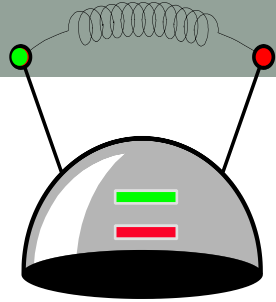


There are two special features about a member function's implementation . . .

1. The class name is included in the function's heading using the :: operator
2. The function can refer to any of the member variables

```
void thinking_cap::slots(char new_green[ ], char new_red[ ])
{
    assert(strlen(new_green) < 50);
    assert(strlen(new_red) < 50);
    strcpy(green_string,  new_green);
    strcpy(red_string, new_red);
}
```

Thinking Cap Implementation



Within the body of the function, the class’s member variables and other methods may all be accessed.

```
void thinking_cap::slots(char new_
{
    assert(strlen(new_green) < 50).
    assert(strlen(new_red) < 50);
    strcpy(green_string,  new_gree
    strcpy(red_string, new_red);
}
```

But, whose member variables are these? Are they

student.green_string

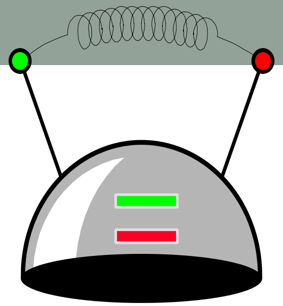
student.red_string

fan.green_string

fan.red_string

?

Thinking Cap Implementation



Within the body of the function, the class’s member variables and other member functions may all be accessed.

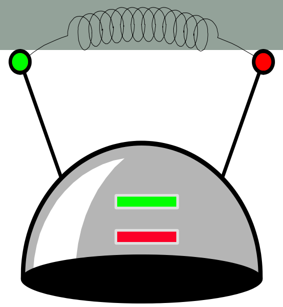
```
void thinking_cap::slots(char new_
{
    assert(strlen(new_green) < 50)
    assert(strlen(new_red) < 50);
    strcpy(green_string,  new_gree
    strcpy(red_string, new_red);
}
```

If we activate student.slots:

student.green_string

student.red_string

Thinking Cap Implementation



Within the body of the function, the class’s member variables and other member functions may all be accessed.

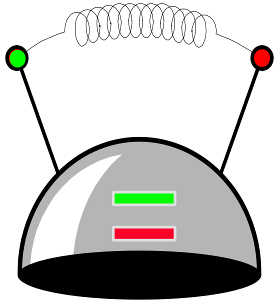
```
void thinking_cap::slots(char new_
{
    assert(strlen(new_green) < 50);
    assert(strlen(new_red) < 50);
    strcpy(green_string,  new_gree
    strcpy(red_string, new_red);
}
```

If we activate fan.slots:

fan.green_string

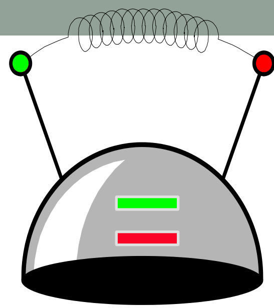
fan.red_string

Thinking Cap Implementation



Here is the implementation of the push_green() member function, which prints the green message:

```
void thinking_cap::push_green( )
{

    cout << green_string << endl;

}
```



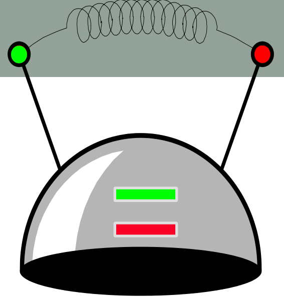
A Quiz

What is the output of this code?

A. *Some junk value*

B. *Program error at run time*

```
class thinking_cap
{
public:
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};

int main( )
{
    thinking_cap student;
    student.push_green( );
}
```

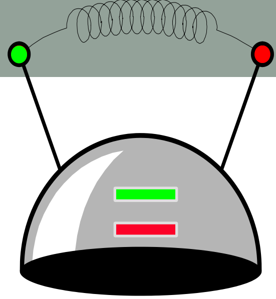


Constructor

An “initialization” function that is guaranteed to be called when an object of the class is created

```
class thinking_cap
{
public:
    thinking_cap(char new_green[], char new_red[]);
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

Constructor



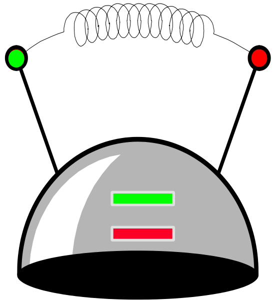
An “initialization” function that is guaranteed to be called when an object of the class is created

```
class thinking_cap
{
public:
    thinking_cap(char new_green[], char new_red[]);
    void slots(char new_green[ ], char new_red[ ]);
    void push_green( ) const;
    void push_red( ) const;
private:
    char green_string[50];
    char red_string[50];
};
```

Which distinction(s) do you see between the constructor and other methods of the class?

- ***A. The constructor has the same name as the class***
- ***B. It doesn’t have a return type***
- ***C. It has formal parameters***
- ***D. A and B***
- ***E. None of the above***

Implementation of the constructor



Do you expect the body of the constructor to be different from the slots() method in this example? Discuss with your group why or why not.

A. Yes

B. No

```
thinking_cap::thinking_cap(char new_green[], char new_red[] )
{
    //Code for initializing the member variables of


}
```

What we have spoken about so far?

- Class = Data + Member Functions.
- Abstract Data Type = Class + information hiding
- How to define a new class type, and place the definition in a header file.
- How to use the header file in a program which declares instances of the class type.
- How to activate member functions.

⍰But you still need to learn how to write the bodies of a class’s methods.

Next time

- C++ classes continued
- Intro to PA1