Lecture 5

Selection Statements

Review from last week

- cin and cout
- directives
- escape sequences
- member functions cout.width(20);
- formatting flags cout.setf(ios::fixed);
- manipulators setwidth(20);

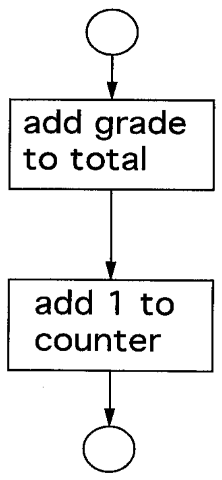
What we will learn about this week

- Simple Flow of Control
  - Sequential
  - Branching mechanisms
    - *If / Else*
    - *Switch*
- Program Style

“If you come to a fork in the road, take it.” – Yogi Berra

Sequence

```
total = total + grade;
counter = counter + 1;
```



Branching (Selection)
Chose between alternatives

Calculate an employee's pay:

pay = rate * 40 + 1.5 * rate * (hours – 40)
if the employee has worked overtime

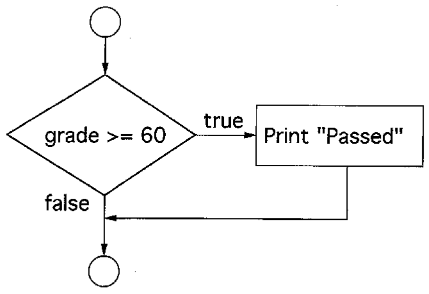
OR

pay = rate * hours
if the employee did not work overtime

If Statement

if

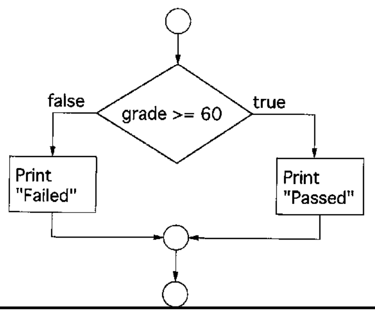
```
if (grade >= 60 )
  cout << "Passed";
```



If - Else Statement

if/else

```
if (grade >= 60 )
  cout << "Passed";
else
  cout << "Failed";
```



Syntax of the **if** – **else** statement

(See display 2.7 page 68 of book)

```
if (Boolean_Expression)
        Yes_Statement;
else
        No_Statement;
```

```
if (Boolean_Expression)
{
            yes_statement1;
             yes_statement2;
            …
}
else
{
             no_statement1;
             no_statement2;
            …
}
```

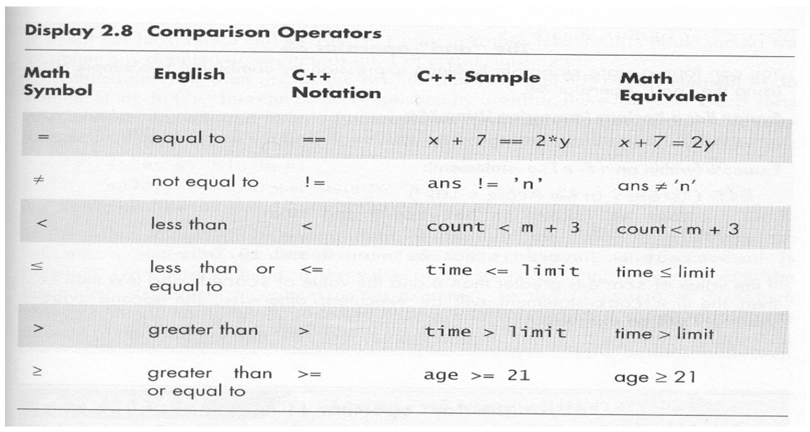
Comparison Operators

==	is equal to	if(x == 6)
!=	is not equal to	if(count != 0)
<	is less than	if(max < 100)
<=	is less than or equal to	if(data <= 17)
>	is greater than	if(left > 2)
>=	is greater than or equal to	if(a>=b)

(be careful not to use a single =)

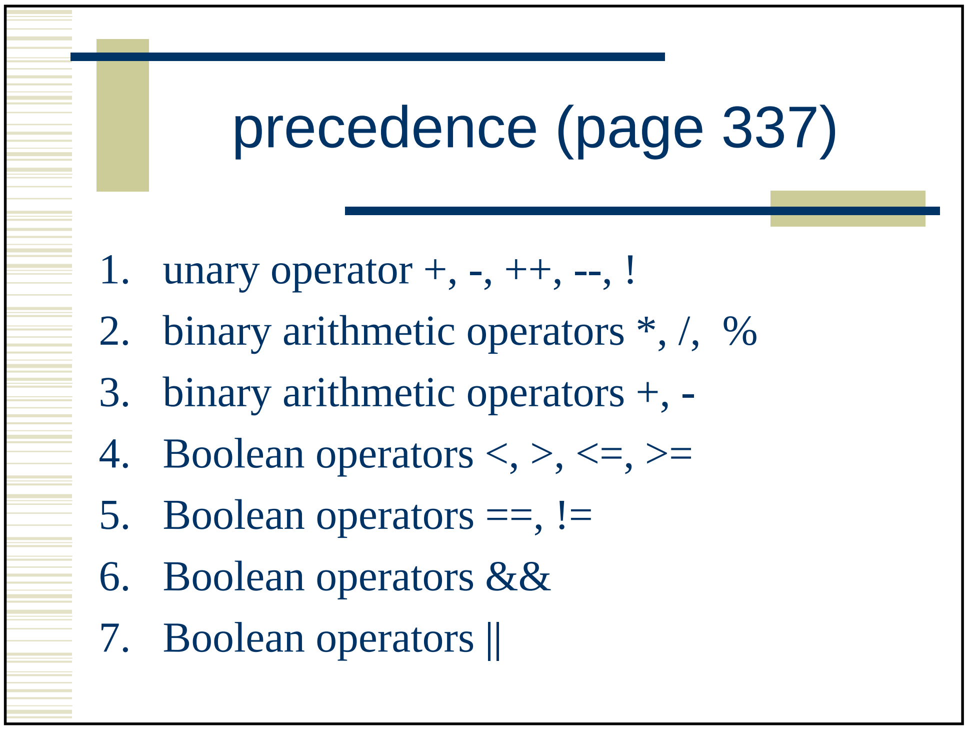
Should not have a space between the two symbols!!!!

Comparison Operators
(Display 2.8 page 69)



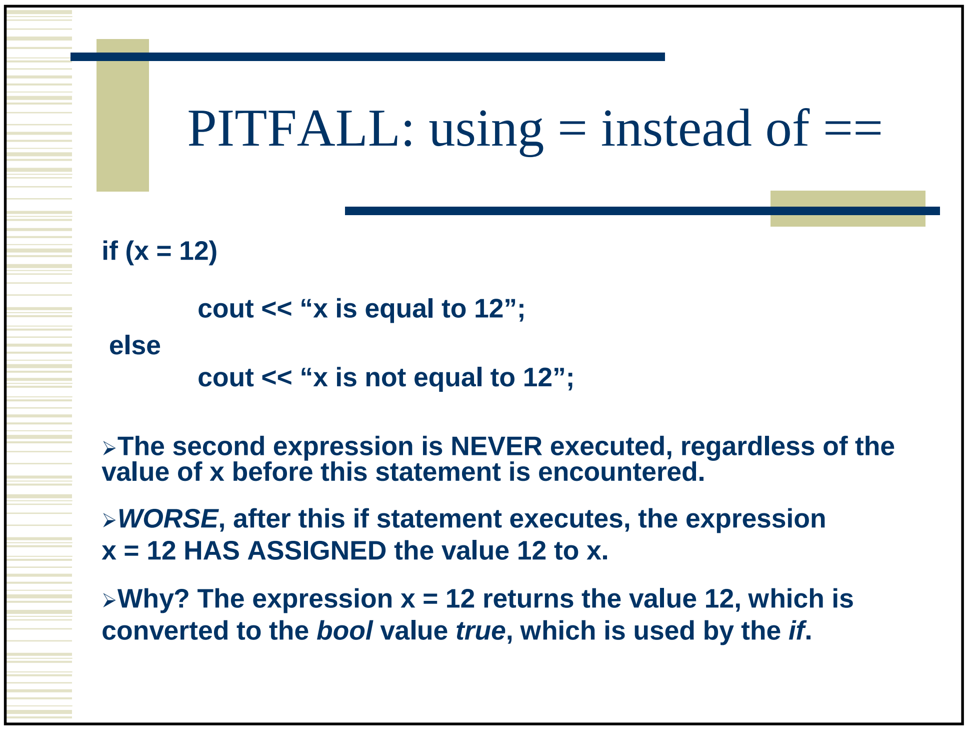
Display 2.8 Comparison Operators

Math Symbol	English	C++ Notation	C++ Sample	Math Equivalent
=	equal to	==	`x + 7 == 2*y`	$x + 7 = 2y$
≠	not equal to	!=	`ans != 'n'`	ans ≠ 'n'
<	less than	<	`count < m + 3`	count < m + 3
≤	less than or equal to	<=	`time <= limit`	time ≤ limit
>	greater than	>	`time > limit`	time > limit
≥	greater than or equal to	>=	`age >= 21`	age ≥ 21



precedence (page 337)

1. unary operator +, -, ++, --, !
2. binary arithmetic operators *, /, %
3. binary arithmetic operators +, -
4. Boolean operators <, >, <=, >=
5. Boolean operators ==, !=
6. Boolean operators &&
7. Boolean operators ||



PITFALL: using = instead of ==

```
if (x = 12)

            cout << "x is equal to 12";
 else
            cout << "x is not equal to 12";
```

- **The second expression is NEVER executed, regardless of the value of x before this statement is encountered.**
- ***WORSE*, after this if statement executes, the expression x = 12 HAS ASSIGNED the value 12 to x.**
- **Why? The expression x = 12 returns the value 12, which is converted to the *bool* value *true*, which is used by the *if*.**

Boolean Expressions

- An expression that can be thought of as true or false
- Boolean expressions use !<>=||&&, and evaluate to true and false
- Integers in Boolean Expressions:
  - 0 is converted to false
  - Any non-zero integer is converted to true

```
if (integer_value)
    cout << "Not zero";
else
    cout << "Is zero";
```

Programming example

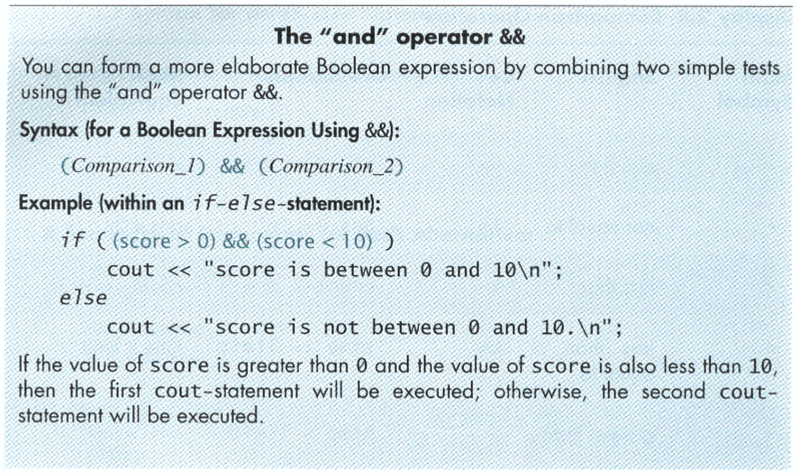
(Display 2.6 page 67)

```
#include <iostream>
int main ()
{
    int hours;
    double gross_pay, rate;

    cout << "Enter the hourly rate of pay: $";
    cin >> rate;
    cout << Enter the number of hours worked, \n"
            << rounded to a whole number of hours: ";
    cin >> hours;

    if (hours > 40)
            gross_pay = rate*40 + 1.5*rate*(hours-40);
    else
             gross_pay = rate*hours;
```

if – else continued

```
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);

    cout << “Hours = “ << hours << endl;
    cout << “Hourly pay rate = $” << rate << endl;
    cout << “Gross pay = $” << gross_pay << endl;

    return 0;
}
```

AND &&



The “and” operator &&

You can form a more elaborate Boolean expression by combining two simple tests using the “and” operator &&.

Syntax (for a Boolean Expression Using &&):

(Comparison_1) && *(Comparison_2)*

Example (within an *if-else*-statement):

```
if ( (score > 0) && (score < 10) )
    cout << "score is between 0 and 10\n";
else
    cout << "score is not between 0 and 10.\n";
```

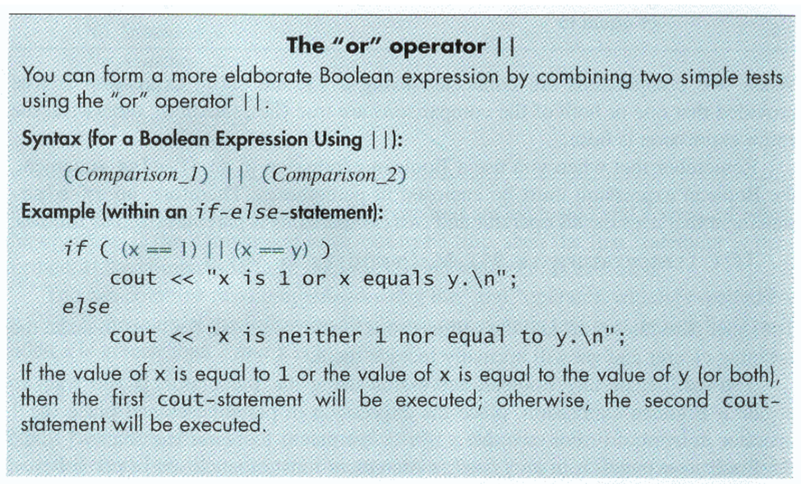
If the value of score is greater than 0 and the value of score is also less than 10, then the first cout-statement will be executed; otherwise, the second cout-statement will be executed.

Boolean AND operator

AND &&   `if (a>b && count == 0)`

B1	B2	B1 && B2
False	False	False
False	True	False
True	False	False
True	True	True

OR ||



The "or" operator ||

You can form a more elaborate Boolean expression by combining two simple tests using the "or" operator ||.

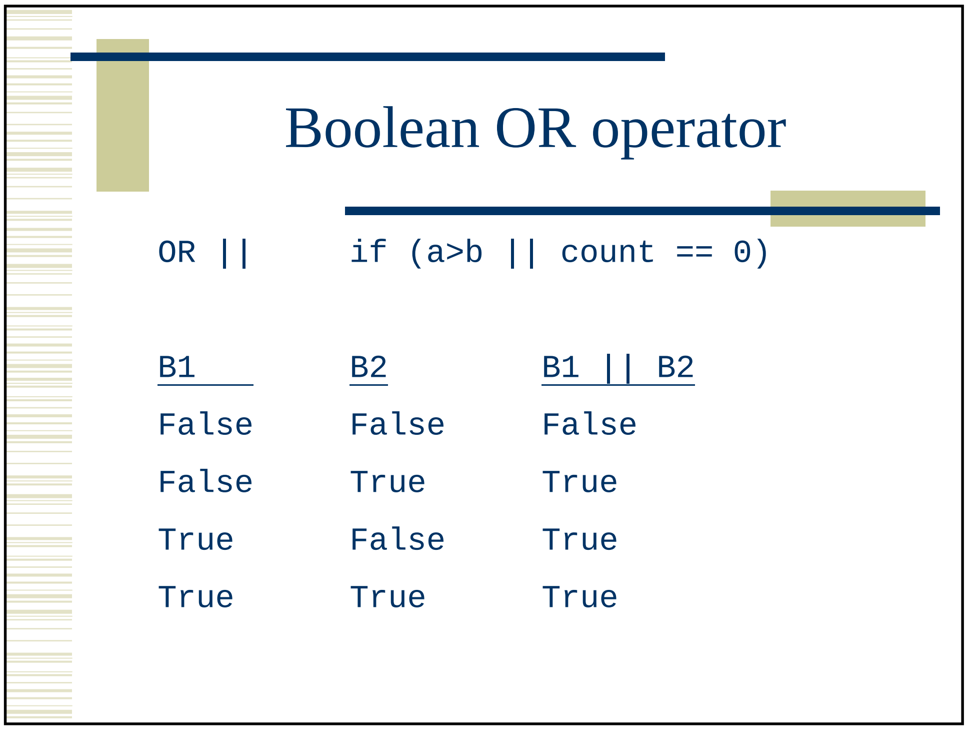
Syntax (for a Boolean Expression Using ||):

(*Comparison_1*) || (*Comparison_2*)

Example (within an if-else-statement):

```
if ( (x == 1) || (x == y) )
    cout << "x is 1 or x equals y.\n";
else
    cout << "x is neither 1 nor equal to y.\n";
```

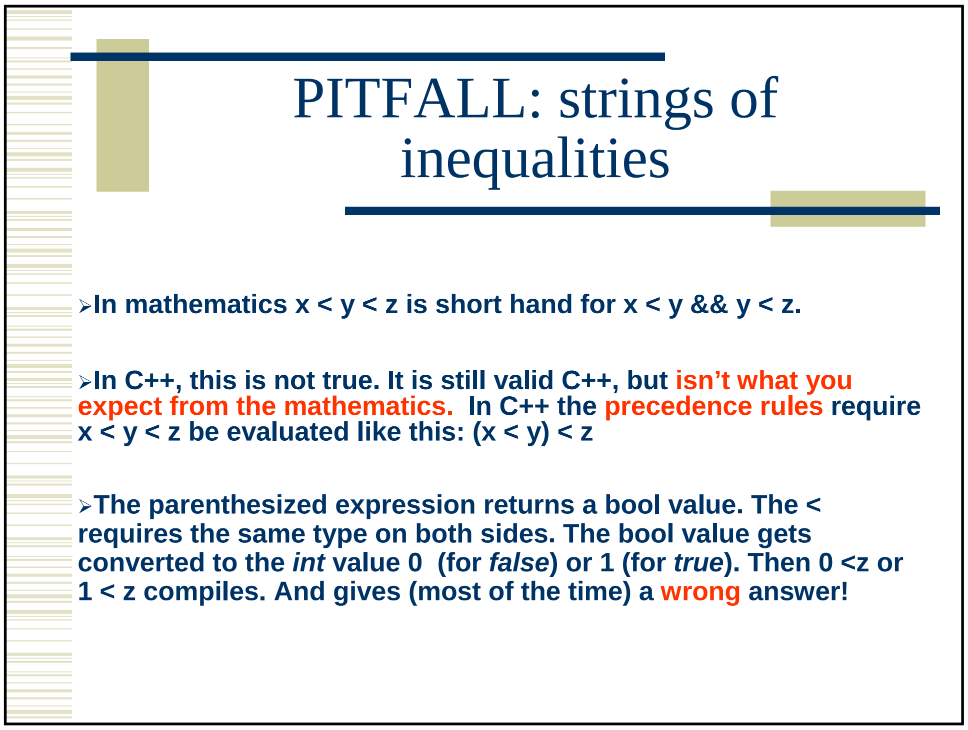
If the value of x is equal to 1 or the value of x is equal to the value of y (or both), then the first cout-statement will be executed; otherwise, the second cout-statement will be executed.



Boolean OR operator

```
OR ||      if (a>b || count == 0)
```

B1	B2	B1 || B2
False	False	False
False	True	True
True	False	True
True	True	True



PITFALL: strings of inequalities

- In mathematics x < y < z is short hand for x < y && y < z.
- In C++, this is not true. It is still valid C++, but isn't what you expect from the mathematics. In C++ the precedence rules require x < y < z be evaluated like this: (x < y) < z
- The parenthesized expression returns a bool value. The < requires the same type on both sides. The bool value gets converted to the *int* value 0 (for *false*) or 1 (for *true*). Then 0 <z or 1 < z compiles. And gives (most of the time) a wrong answer!

Nested Ifs

```
if (guess > number)
    cout << "Too high.";
else
    if (guess < number)
        cout << "Too low.";
    else
        if (guess == number)
            cout << "Correct!";
```

More readable if indentation rules are broken:

```
if (guess > number)
   cout << "Too high.";
else if (guess < number)
   cout << "Too low.";
else if (guess == number)
   cout << "Correct!";
```

This format works because the conditions are mutually exclusive:

```
if (guess > number)
    cout << "Too high.";
else if (guess < number)
    cout << "Too low.";
else // (guess == number)
    cout << "Correct!";
```

Dangling Else

```
if (fuel < 0.75)
   {
        if (fuel < 0.25)
                cout << “Fuel low!\n”;
   }
else
        cout “Fuel over ¾ full.\n”;
```

Multi-way statements

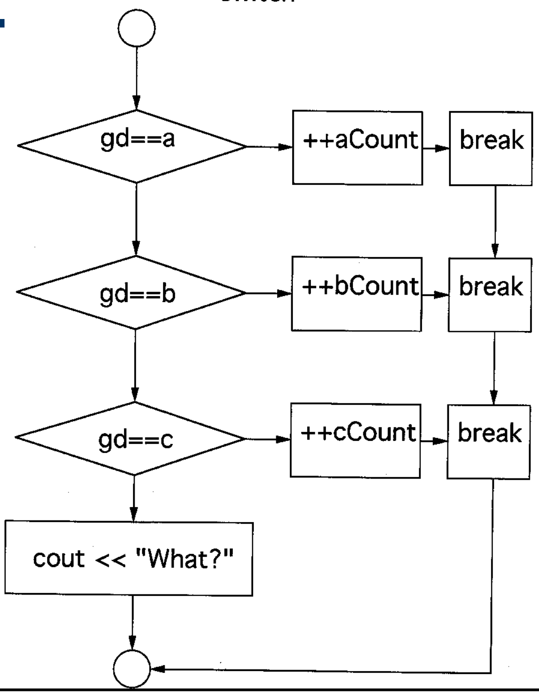
An If statement is a two-way branch.

When you want three or four way branches, you can nest the If statements.

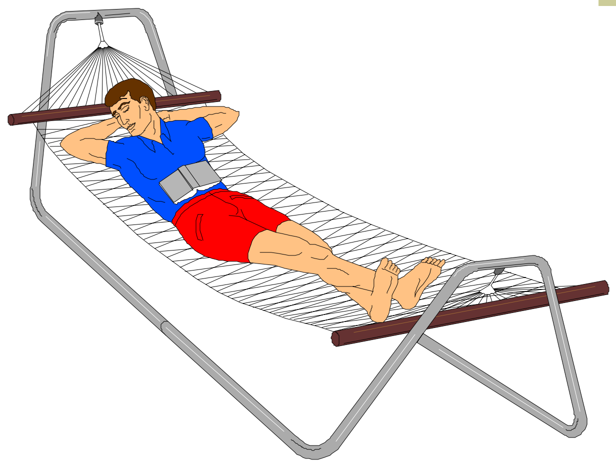
A switch statement is another way to implement a multi-way branch.

```
switch (gd) {
  case 'a':
    ++ aCount;
    break;
  case 'b':
    ++ bCount;
    break;
  case 'c':
    ++cCount;
    break;
  default:
    cout << "What?";
    break;
}
```

switch

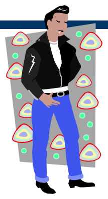


Take a Break!



Programming Style
(or how not to write confusing code)

Clarity and Style



- Your goal is not just to get a program to work.
- You are graded on both program **clarity** and correctness:
  - Comments are important!
  - A correct header is important!
  - Output identification is important!
  - Spacing and Indentation are important!
  - Clear and concise code is important! (KISS)

Programming Style

- Group like things together
  - indent
  - leave a blank line
- Comments
  - // comment follows until end of line
  - /* multi-line comment ends with */
- Use Headers (I will no longer grade programs without names in source and **output**)

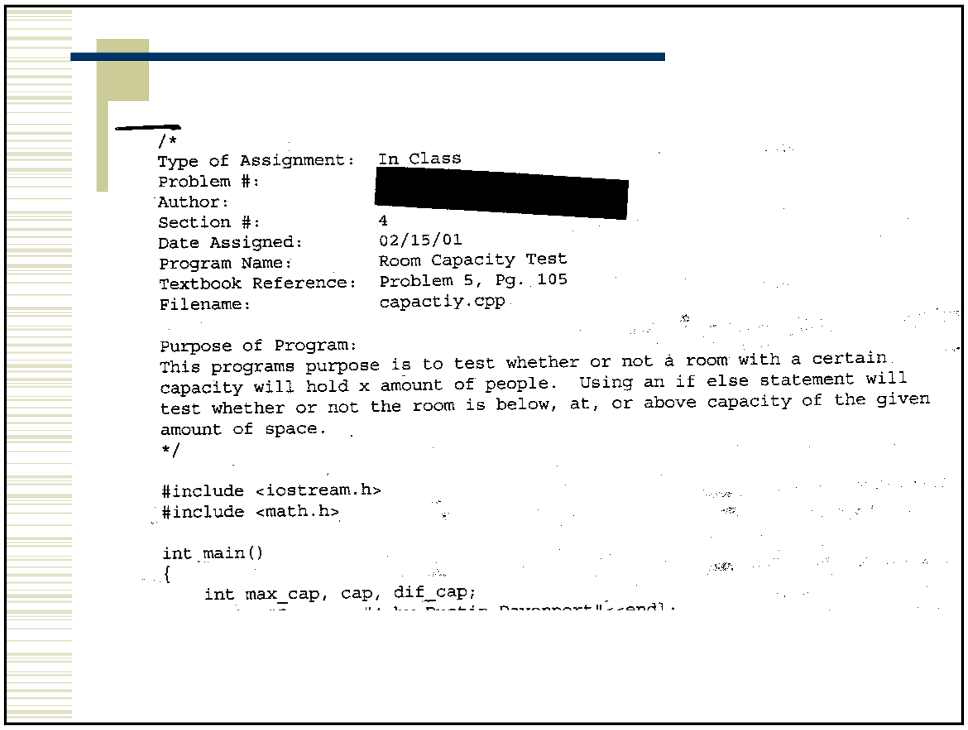
Writing good code

- If the problem is complex, make sure you have a game plan (algorithm).
- Write one section of code at a time, test it, and when it works move on to the next section.
- Comment the main sections of code, then after the program is done, go back and add additional comments to clarify what you are doing.
- Remember to use blank lines to group like statements together.
- Proper indentation also makes the code more legible.

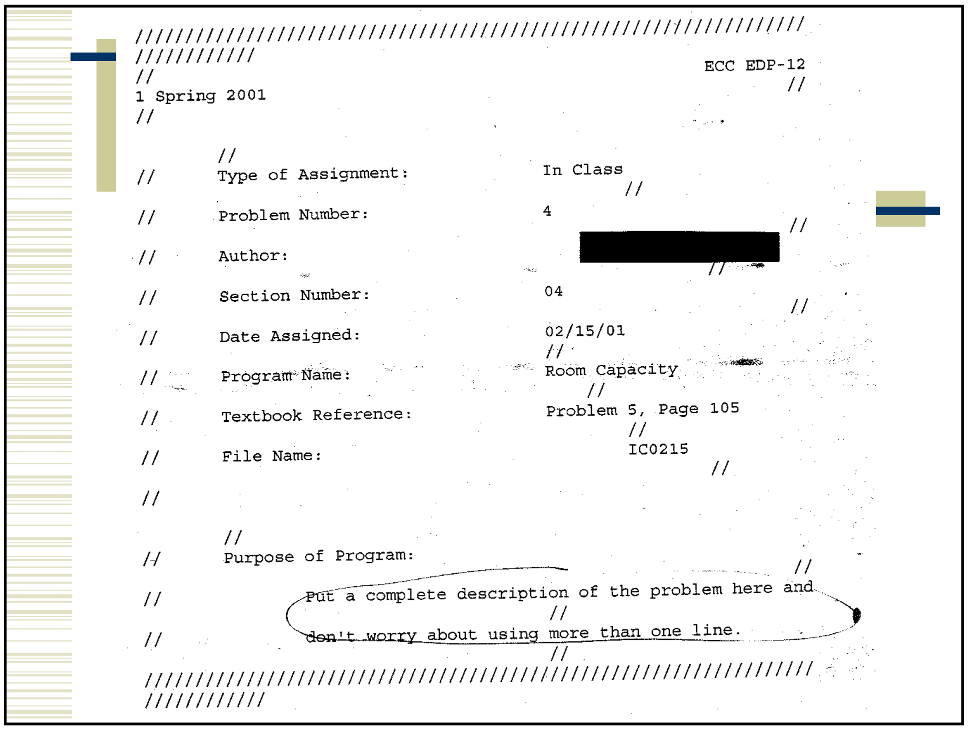
pi.cpp

```
/////////////////////////////////////////////////////////////////////////////
//                          ECC EDP-121 Spring 2001                        //
//                                                                         //
//   Type of Assignment:        In Class                                   //
//   Problem Number:            4                                          //
//   Author:                    Glenn Mayer                                //
//   Section Number:            01                                         //
//   Date Assigned:             01/08/01                                   //
//   Program Name:              Calculate PI                               //
//   Textbook Reference:        Problem 23, Page 321                       //
//   File Name:                 pi.cpp                                     //
//                                                                         //
//   Purpose of Program:                                                   //
//       To calculate PI using an infinite series and show the increase in //
//     precision as you use more terms.                                    //
/////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>
#include <constrea.h>

// Main Program
int main( )
{
    // Variable Declations
    float pi;                  // the calculated value of pi
    long int terms;            // the maximum number of terms
    long int i;                // a loop variable indicating the current term
```

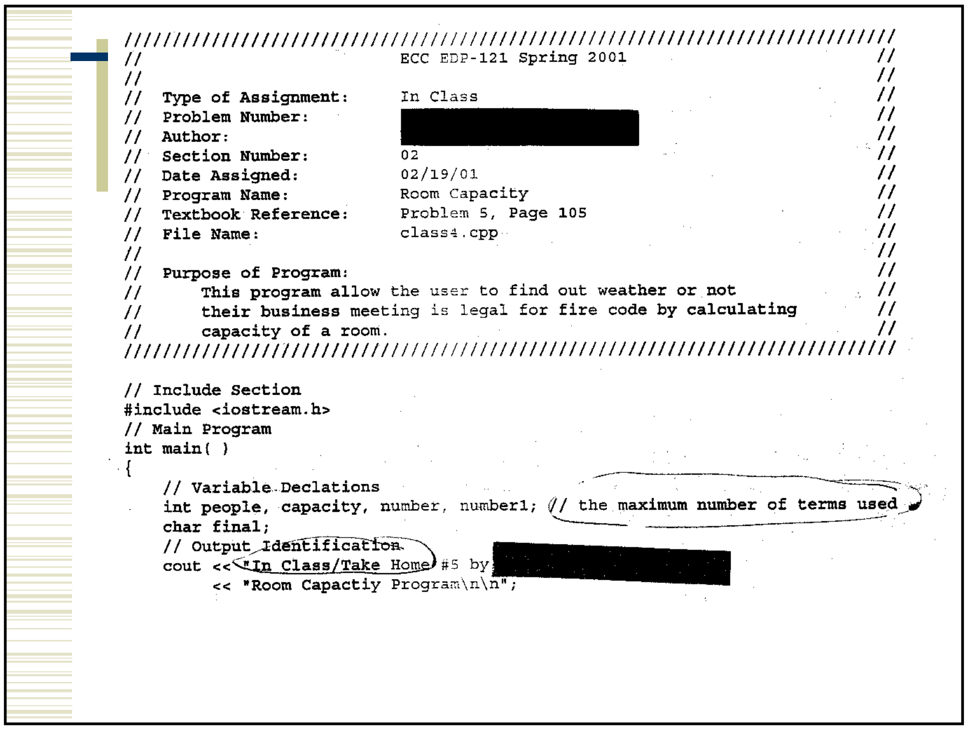
```
/////////////////////////////////////////////////////////////////////////////
//                      ECC EDP-121 Spring 2001                           //
//                                                                         //
//   Type of Assignment:      In Class                                    //
//   Problem Number:          [redacted]                                  //
//   Author:                                                               //
//   Section Number:          04                                           //
//   Date Assigned:           02/15/01                                     //
//   Program Name:            Example Program                              //
//   Textbook Reference:      Problem 5, Page 105                          //
//   File Name:               inclass 4.cpp                                //
//                                                                         //
//   Purpose of Program:                                                   //
//       To find out how many people can fit in the room without           //
//           overdoing it.                                                 //
//                                                                         //
/////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>

// Main Program
int main( )
{
    // Variable Declations
    int number_of_people, room_capacity, people_person;     // the maximum number of te
rms used
```



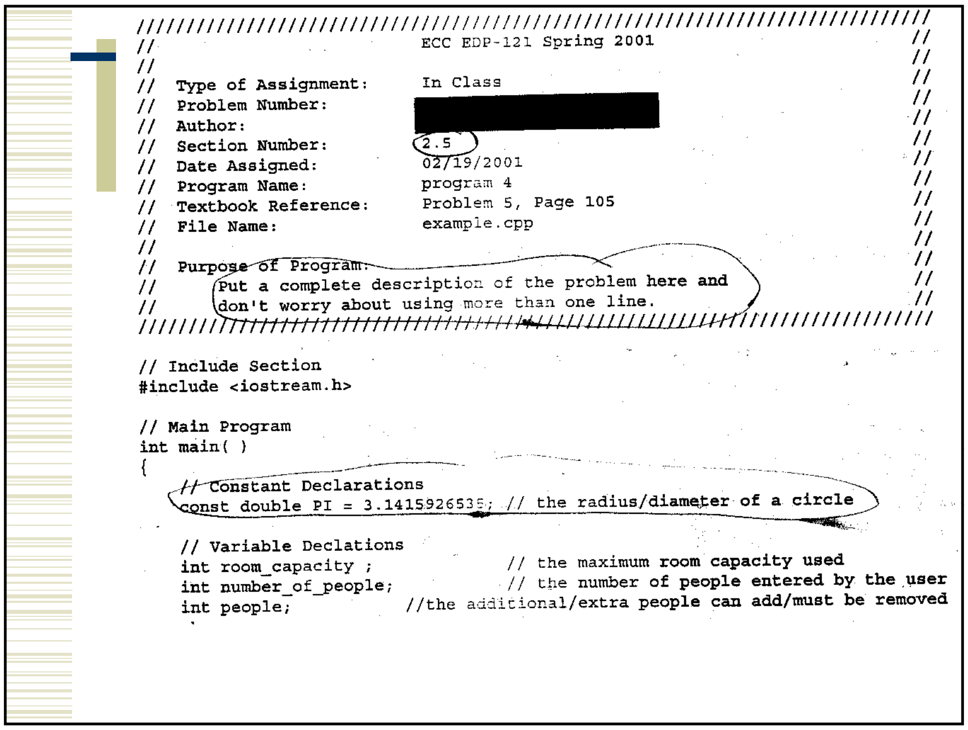
```
/*
Type of Assignment:   In Class
Problem #:            [redacted]
Author:
Section #:            4
Date Assigned:        02/15/01
Program Name:         Room Capacity Test
Textbook Reference:   Problem 5, Pg. 105
Filename:             capactiy.cpp

Purpose of Program:
This programs purpose is to test whether or not a room with a certain
capacity will hold x amount of people.  Using an if else statement will
test whether or not the room is below, at, or above capacity of the given
amount of space.
*/

#include <iostream.h>
#include <math.h>

int main()
{
     int max_cap, cap, dif_cap;
     [illegible]
```



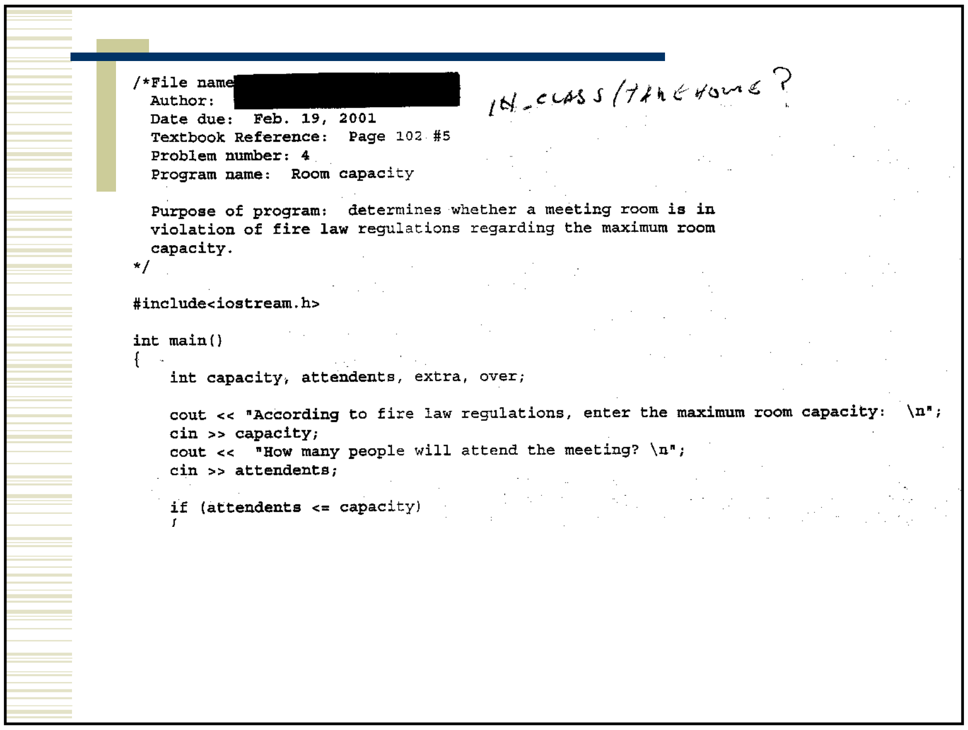
```
//////////////////////////////////////////////////////////////////////
////////////
//                                                        ECC EDP-12
1 Spring 2001                                                   //
//

        //
//      Type of Assignment:               In Class
                                                  //
//      Problem Number:                   4
                                                                //
//      Author:                           [redacted]
                                                        //
//      Section Number:                   04
                                                                //
//      Date Assigned:                    02/15/01
                                          //
//      Program Name:                     Room Capacity
                                              //
//      Textbook Reference:               Problem 5, Page 105
                                                  //
//      File Name:                                IC0215
                                                          //
//

        //
//      Purpose of Program:                                     //
//              Put a complete description of the problem here and
                                          //
//              don't worry about using more than one line.
                                          //
//////////////////////////////////////////////////////////////////////
////////////
```



```
/////////////////////////////////////////////////////////////////////////////
//                              ECC EDP-121 Spring 2001                     //
//                                                                          //
//  Type of Assignment:         In Class                                    //
//  Problem Number:             [redacted]                                  //
//  Author:                                                                 //
//  Section Number:             02                                          //
//  Date Assigned:              02/19/01                                    //
//  Program Name:               Room Capacity                               //
//  Textbook Reference:         Problem 5, Page 105                         //
//  File Name:                  class4.cpp                                  //
//                                                                          //
//  Purpose of Program:                                                     //
//      This program allow the user to find out weather or not              //
//      their business meeting is legal for fire code by calculating        //
//      capacity of a room.                                                 //
/////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>
// Main Program
int main( )
{
    // Variable Declations
    int people, capacity, number, number1; // the maximum number of terms used
    char final;
    // Output Identification
    cout << "In Class/Take Home #5 by [redacted]
         << "Room Capactiy Program\n\n";
```

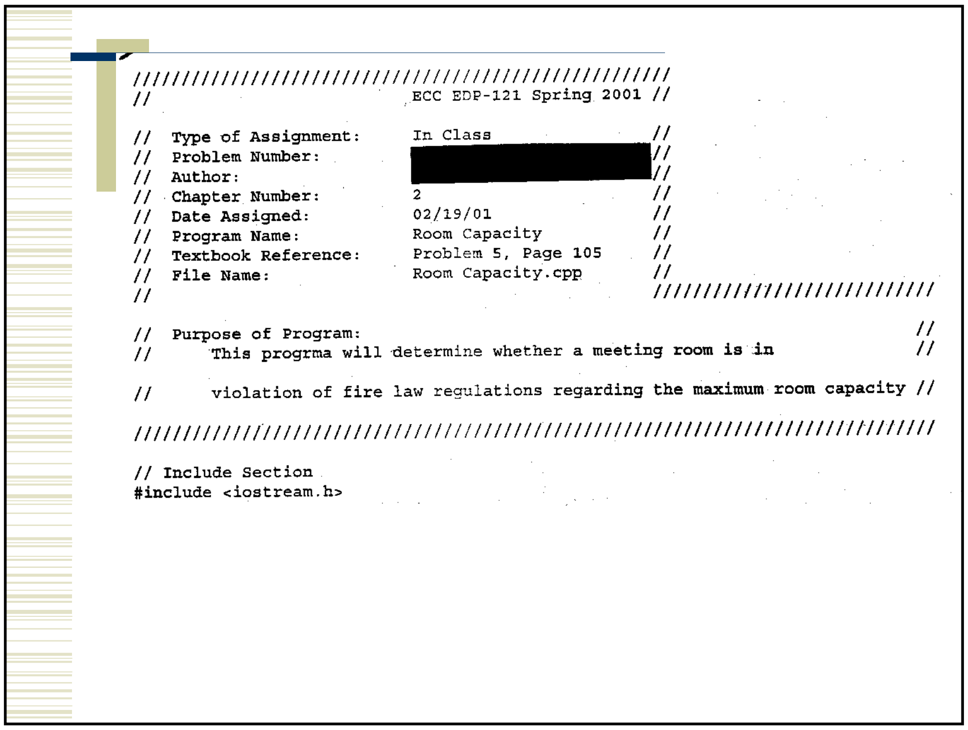


```
/////////////////////////////////////////////////////////////////////////////
//                              ECC EDP-121 Spring 2001                     //
//                                                                          //
//  Type of Assignment:         In Class                                    //
//  Problem Number:             [redacted]                                  //
//  Author:                                                                 //
//  Section Number:             2.5                                         //
//  Date Assigned:              02/19/2001                                  //
//  Program Name:               program 4                                   //
//  Textbook Reference:         Problem 5, Page 105                         //
//  File Name:                  example.cpp                                 //
//                                                                          //
//  Purpose of Program:                                                     //
//      Put a complete description of the problem here and                  //
//      don't worry about using more than one line.                         //
/////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>

// Main Program
int main( )
{
    // Constant Declarations
    const double PI = 3.14159265358; // the radius/diameter of a circle

    // Variable Declations
    int room_capacity ;              // the maximum room capacity used
    int number_of_people;            // the number of people entered by the user
    int people;              //the additional/extra people can add/must be removed
```

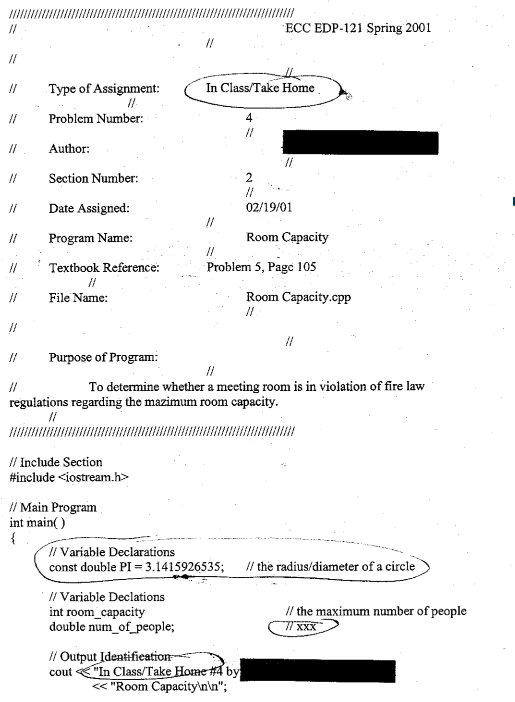


IN-CLASS/TAKE HOME?

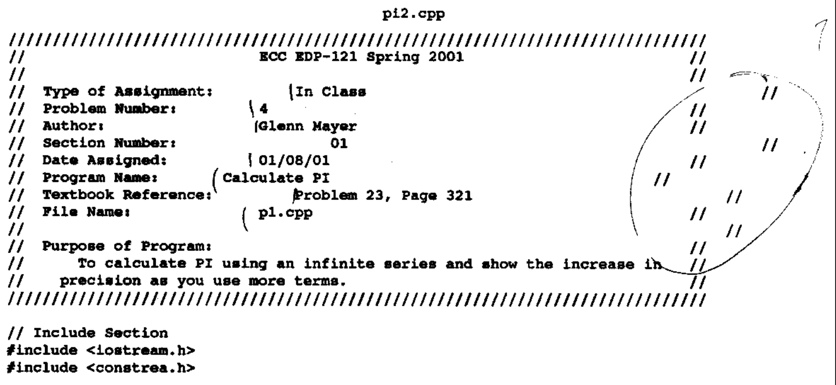
```
/*File name [redacted]
  Author:
  Date due:  Feb. 19, 2001
  Textbook Reference:  Page 102 #5
  Problem number: 4
  Program name:  Room capacity

  Purpose of program:  determines whether a meeting room is in
  violation of fire law regulations regarding the maximum room
  capacity.
*/

#include<iostream.h>

int main()
{
    int capacity, attendents, extra, over;

    cout << "According to fire law regulations, enter the maximum room capacity:  \n";
    cin >> capacity;
    cout << "How many people will attend the meeting? \n";
    cin >> attendents;

    if (attendents <= capacity)
    {
```



```
//////////////////////////////////////////////////////////
//                             ECC EDP-121 Spring 2001 //

//  Type of Assignment:        In Class                //
//  Problem Number:            [redacted]              //
//  Author:                                            //
//  Chapter Number:            2                       //
//  Date Assigned:             02/19/01                //
//  Program Name:              Room Capacity           //
//  Textbook Reference:        Problem 5, Page 105     //
//  File Name:                 Room Capacity.cpp       //
//                                                     ///////////////////////////

//  Purpose of Program:                                                         //
//      This progrma will determine whether a meeting room is in                //

//      violation of fire law regulations regarding the maximum room capacity //

////////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>
```



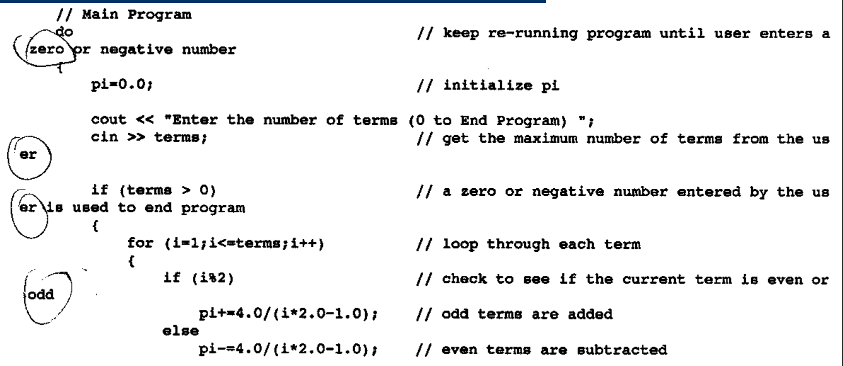
```
/////////////////////////////////////////////////////////////////////////////
//                                   ECC EDP-121 Spring 2001
                    //
//
                                            //
//     Type of Assignment:     In Class/Take Home
                //
//     Problem Number:                4
                                      //
//     Author:                          [redacted]
                                            //
//     Section Number:                2
                                      //
//     Date Assigned:                 02/19/01
                                  //
//     Program Name:                  Room Capacity
                                  //
//     Textbook Reference:       Problem 5, Page 105
          //
//     File Name:                     Room Capacity.cpp
                                      //
//
                                            //
//     Purpose of Program:
                                  //
//            To determine whether a meeting room is in violation of fire law
regulations regarding the mazimum room capacity.
     //
/////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>

// Main Program
int main( )
{
     // Variable Declarations
     const double PI = 3.1415926535;       // the radius/diameter of a circle

     // Variable Declations
     int room_capacity                       // the maximum number of people
     double num_of_people;                   // xxx

     // Output Identification
     cout << "In Class/Take Home #4 by [redacted]
          << "Room Capacity\n\n";
```



pi2.cpp

```
///////////////////////////////////////////////////////////////////////////////
//                        ECC EDP-121 Spring 2001                           //
//                                                                          //
//  Type of Assignment:         In Class                                        //
//  Problem Number:          4                                              //
//  Author:                  Glenn Mayer                                    //
//  Section Number:                  01                                         //
//  Date Assigned:           01/08/01                                       //
//  Program Name:        Calculate PI                                   //
//  Textbook Reference:          Problem 23, Page 321                           //
//  File Name:               p1.cpp                                         //
//                                                                              //
//  Purpose of Program:                                                     //
//       To calculate PI using an infinite series and show the increase in  //
//    precision as you use more terms.                                      //
///////////////////////////////////////////////////////////////////////////////

// Include Section
#include <iostream.h>
#include <constrea.h>
```

?



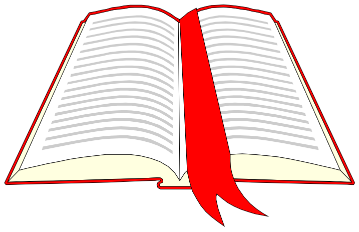
```
        // Main Program
        do                                      // keep re-running program until user enters a
zero or negative number
        {
            pi=0.0;                             // initialize pi

            cout << "Enter the number of terms (0 to End Program) ";
            cin >> terms;                       // get the maximum number of terms from the us
er

            if (terms > 0)                      // a zero or negative number entered by the us
er is used to end program
            {
                for (i=1;i<=terms;i++)          // loop through each term
                {
                    if (i%2)                    // check to see if the current term is even or
odd
                        pi+=4.0/(i*2.0-1.0);    // odd terms are added
                    else
                        pi-=4.0/(i*2.0-1.0);    // even terms are subtracted
```

Re-submit Programs

- Remember that you can re-submit your program and I will average the two grades as the final grade for the project.
- Turn in both the original and updated programs when you re-submit.
- Remember to change the header on the updated program.
- Turn the programs in to the re-submit folder!

Reading Assignment

Sections 3.1, 3.2, 3.3
(pg 99-117)



Labs

In Class:

Room Capacity

Programming Project #6 page 95

Simple Version – Runs Once

Take Home:

Days of the Week