

Feb 11, 10 7:23 **test_Circle.cpp** Page 1/1

```
// file: test_Circle.cpp
//
// This program calculates the area of a circle, given the radius.
//
// Programmer: Dick Furnstahl  furnstahl.1@osu.edu
//
// Revision history:
// 05-Feb-2009  original version, for 780.20 Computational Physics
// 11-Feb-2010  added output statements
//
// Notes:
// * Use make test_Circle to compile and link with the Circle class
// * Compare to area.cpp from Session 1
//
// To do:
// * Test additional methods for the Circle class
//
//*****//
// include files
#include <iostream>
using namespace std; // otherwise prepend std:: to cin, cout, endl

#include "Circle.h" // include the Circle class definition
//*****//

int
main ()
{
    double my_radius;
    // Have to specify std namespace now with before cout, cin, endl
    cout << "Enter the radius of a circle: "; // ask for radius
    cin >> my_radius;

    Circle my_first_circle(my_radius); // create a Circle object

    double my_area = my_first_circle.area();
    cout << "radius=" << my_radius << ",area=" << my_area << endl;

    // Now create another circle in a block (i.e., between {}'s)
    {
        Circle my_second_circle(2.*my_radius);
        cout << "Made a 2nd circle of radius: "
             << my_second_circle.get_radius() << endl;
        cout << endl << "Last statement inside {}'s..." << endl;
    }
    cout << "Now outside {}'s." << endl << endl;

    return (0); // successful completion
}
//*****//
```

Feb 11, 10 7:26 **area.cpp** Page 1/1

```
// file: area.cpp
//
// This program calculates the area of a circle, given the radius.
//
// Programmer: Dick Furnstahl  furnstahl.1@osu.edu
//
// Revision history:
// 02-Jan-2004  original version, for 780.20 Computational Physics
// 01-Jan-2010  updates to "To do" wishlist
//
// Notes:
// * compile with: "g++ -o area area.cpp"
//
// To do:
// 1. output the answer with higher precision (more digits)
// 2. use the "predefined" value of pi or atan
// 3. define an inline square function
// 4. split the calculation off into a function (subroutine)
// 5. output to a file (and/or input from a file)
// 6. rewrite using a Circle class
//
//*****//
// include files
#include <iostream> // this has the cout, cin definitions
using namespace std; // if omitted, then need std::cout, std::cin
//*****//

const double pi = 3.1415926535897932385; // define pi as a constant

int
main ()
{
    double radius; // every variable is declared as int or double or ...

    cout << "Enter the radius of a circle: "; // ask for radius
    cin >> radius;

    double area = pi * radius * radius; // area formula

    cout << "radius=" << radius << ", area=" << area;

    return 0; // "0" for successful completion
}
//*****//
```

Feb 11, 10 7:20 **Circle.h** Page 1/1

```
// file: Circle.h
//
// Header file for the Circle C++ class.
//
// Programmer: Dick Furnstahl  furnstahl.1@osu.edu
//
// Revision history:
//   02/05/09  original version
//
// Notes:
//   *
//
// To do:
//   *
//
//*****

// The ifndef/define macro ensures that the header is only included once
#ifndef CIRCLE_H
#define CIRCLE_H

// include files

class Circle
{
public:
    Circle (const double rad); // constructor
    ~Circle (); // destructor

    // accessor functions
    double get_radius ();
    void set_radius (const double rad);
    double area ();

private:
    double radius; // the circle radius
    inline double sqr (const double x) {return x*x;}; // inline function
};

#endif
```

Feb 11, 10 7:23 **Circle.cpp** Page 1/1

```
// file: Circle.cpp
//
// Definitions for the Circle C++ class.
//
// Programmer: Dick Furnstahl  furnstahl.1@osu.edu
//
// Revision history:
//   02/05/09  Original version.
//
//*****
// include files
#include "Circle.h" // include the header for this class

//*****

#include <iostream>

const double pi = 3.1415926535897932385; // define pi

Circle::Circle (const double rad) // Constructor for Circle
{
    radius = rad; // set the internal (private) radius to the passed value

    // For debugging, print a message when we make a circle
    std::cout << "A circle of radius " << radius
                << " is created ..." << std::endl;
}

Circle::~Circle () // Destructor for Circle (not much to do here!)
{
    // For debugging, print a message when we make a circle
    std::cout << "A circle of radius " << radius
                << " is destroyed ..." << std::endl;
}

void Circle::set_radius(const double rad)
{
    radius = rad;
}

double Circle::get_radius()
{
    return radius;
}

double Circle::area()
{
    return pi * sqr(radius);
}

//*****
```