CS 210 **Sept. 1, 2016**
Day 3

Finding the Area and Circumference of a Circle

```
1.  /*
2.   * Calculates and displays the area and circumference of a circle
3.   */
4.
5.  #include <stdio.h> /* printf, scanf definitions */
6.  #define PI 3.14159
7.
8.  int
9.  main(void)
10. {
11.       double radius;    /* input - radius of a circle  */
12.       double area;      /* output - area of a circle   */
13.       double circum;    /* output - circumference      */
14.
15.       /* Get the circle radius */
16.
17.       /* Calculate the area */
18.          /* Assign PI * radius * radius to area. */
19.
20.       /* Calculate the circumference */
21.          /* Assign 2 * PI * radius to circum */
22.
23.       /* Display the area and circumference */
24.
25.       return (0);
26. }
```

Program Design

```c
/*
 * Calculates and displays the area and circumference of a circle
 */

#include <stdio.h> /* printf, scanf definitions */
#define PI 3.14159

int
main(void)
{
	double radius; /* input - radius of a circle */
	double area;   /* output - area of a circle  */
	double circum; /* output - circumference     */

	/* Get the circle radius */
	printf("Enter radius> ");
	scanf("%lf", &radius);

	/* Calculate the area */
	area = PI * radius * radius;

	/* Calculate the circumference */
	circum = 2 * PI * radius;

	/* Display the area and circumference */
	printf("The area is %.4f\n", area);
	printf("The circumference is %.4f\n", circum);

	return (0);
}
```

```
Enter radius> 5.0
The area is 78.5397
The circumference is 31.4159
```

Program Implementation

Square Root Program

```c
/*
 * Performs three square root computations
 */

#include <stdio.h> /* definitions of printf, scanf */
#include <math.h>  /* definition of sqrt           */

int
main(void)
{
      double first, second,    /* input - two data values        */
             first_sqrt,       /* output - square root of first  */
             second_sqrt,      /* output - square root of second */
             sum_sqrt;         /* output - square root of sum    */

      /* Get first number and display its square root. */
      printf("Enter the first number> ");
      scanf("%lf", &first);
      first_sqrt = sqrt(first);
      printf("The square root of the first number is %.2f\n", first_sqrt);
```

TABLE 3.1 Some Mathematical Library Functions

Function	Standard Header File	Purpose: Example	Argument(s)	Result
`abs(x)`	`<stdlib.h>`	Returns the absolute value of its integer argument: if `x is -5`, `abs(x)` is `5`	`int`	`int`
`ceil(x)`	`<math.h>`	Returns the smallest integral value that is not less than `x`: if `x` is `45.23`, `ceil(x)` is `46.0`	`double`	`double`
`cos(x)`	`<math.h>`	Returns the cosine of angle `x`: if `x` is `0.0`, `cos(x)` is `1.0`	`double` (radians)	`double`
`exp(x)`	`<math.h>`	Returns e^x where e = `2.71828...`: if `x` is `1.0`, `exp(x)` is `2.71828`	`double`	`double`
`fabs(x)`	`<math.h>`	Returns the absolute value of its type `double` argument: if `x` is `-8.432`, `fabs(x)` is `8.432`	`double`	`double`
`floor(x)`	`<math.h>`	Returns the largest integral value that is not greater than `x`: if `x` is `45.23`, `floor(x)` is `45.0`	`double`	`double`
`log(x)`	`<math.h>`	Returns the natural logarithm of `x` for `x > 0.0`: if `x` is `2.71828`, `log(x)` is `1.0`	`double`	`double`
`log10(x)`	`<math.h>`	Returns the base-10 logarithm of `x` for `x > 0.0`: if `x` is `100.0`, `log10(x)` is `2.0`	`double`	`double`
`pow(x, y)`	`<math.h>`	Returns x^y. If `x` is negative, `y` must be integral: if `x` is `0.16` and `y` is `0.5`, `pow(x,y)` is `0.4`	`double, double`	`double`
`sin(x)`	`<math.h>`	Returns the sine of angle `x`: if `x` is `1.5708`, `sin(x)` is `1.0`	`double` (radians)	`double`
`sqrt(x)`	`<math.h>`	Returns the nonnegative square root of `x` ($\sqrt{x}$) for `x ≥ 0.0`: if `x` is `2.25`, `sqrt(x)` is `1.5`	`double`	`double`
`tan(x)`	`<math.h>`	Returns the tangent of angle `x`: if `x` is `0.0`, `tan(x)` is `0.0`	`double` (radians)	`double`

P. 121 of the text

DrawCircle might look like this:

```
void DrawCircle()
{
    printf("   * *  \n");
    printf(" *     * \n");
    printf(" *     * \n");
    printf("   * *  \n");
}
```

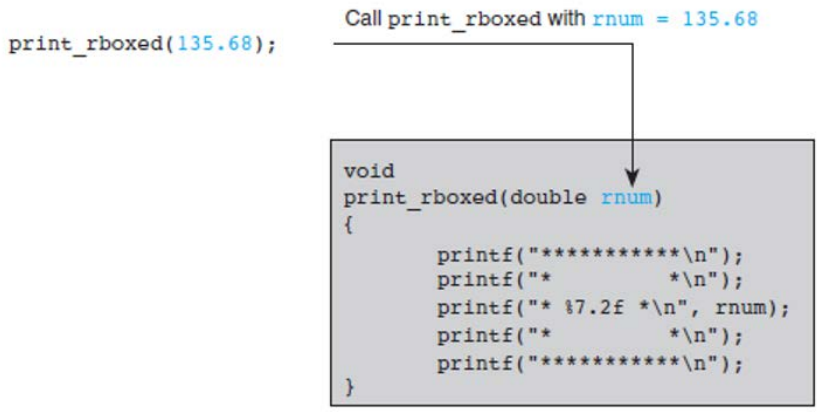
The DrawTriangle function can be broken down into two simpler pieces: DrawIntersectingLines and DrawBase. The complete program is shown below.

```c
#include<stdio.h>
void DrawCircle();           //function prototypes
void DrawTriangle();
void DrawIntersectingLines();
void DrawBase();

void main(void)
{
    DrawCircle();
    DrawTriangle();
}
void DrawCircle()
{
    printf("   * *  \n");
    printf(" *     * \n");
    printf(" *     * \n");
    printf("   * *  \n");
}
void DrawTriangle()
{
    DrawIntersectingLines();
    DrawBase();
}
void DrawIntersectingLines()
{
    printf("  / \\   \n");
    printf(" /   \\  \n");
    printf("/     \\ \n");
}
void DrawBase()
{
    printf("-------\n");
}
```

```
  * *
 *   *
 *   *
  * *
 /\
/  \
/   \
-------
```

Press any key to continue . . .



This function has a single input argument and returns nothing. Its return type is void.

```
#include "stdio.h"
void Fun1(double x);  //Prototype – note semicolon at end.

int main()
  {double x;
   x = 22;
   //Function call
   Fun1(x);      //x here is an argument
   return 0;
  }
//Function body
//This function takes one double argument and returns nothing.
//double x is called a formal parameter.
void Fun1(double x)  //Function heading (no semicolon)
  {double y;           //Function body
   y = 3*x*x + 9.7;
   printf("if x = %f, y = %f \n", x, y);
  }
```

Note that we could call x by a different name in the function since it is indeed a different variable.

We can change this function to one that accepts a double argument and returns a double result:

```
#include "stdio.h"
double Fun1(double x);  //Prototype – note semicolon at end.

int main()
  {double x, y;
   x = 22;
   //Function call
   y = Fun1(x);    //x here is an argument
   printf("if x = %f, y = %f \n", x, y);

   return 0;
  }
//Function body
//This function takes one double argument and returns a double.
//double x is called a formal parameter.
double Fun1(double x)  //Function heading (no semicolon)
  {double y;           //Function body
   y = 3*x*x + 9.7;
   return y;
  }
```

Example

Function scale

```
1.  /*
2.   * Multiplies its first argument by the power of 10 specified
3.   * by its second argument.
4.   * Pre : x and n are defined and math.h is included.
5.   */
6.  double
7.  scale(double x, int n)
8.  {
9.       double scale_factor;     /* local variable */
10.      scale_factor = pow(10, n);
11.
12.      return (x * scale_factor);
13. }
```

```
/*
 * Tests function scale.
 */
#include <stdio.h>            /* printf, scanf definitions */
#include <math.h>             /* pow definition */

/* Function prototype */
double scale(double x, int n);

int
main(void)
{
      double num_1;
      int num_2;

      /* Get values for num_1 and num_2 */
      printf("Enter a real number> ");
      scanf("%lf", &num_1);
      printf("Enter an integer> ");
      scanf("%d", &num_2);

      /* Call scale and display result. */
      printf("Result of call to function scale is %f\n",
             scale(num_1, num_2));          actual arguments

      return (0);
}                                           information flow

double
scale(double x, int n)                      formal parameters
{
      double scale_factor;      /* local variable - 10 to power n */

      scale_factor = pow(10, n);

      return (x * scale_factor);
}
```

```
Enter a real number> 2.5
Enter an integer> -2
Result of call to function scale is 0.025
```

Engr 123 **August 31, 2016**
Polynomial method

Write a console application that prompts the user for a value for x (a double). Evaluate and print the value of y where $y = x^4 - 3x^3 + 2x^2 + 1$. Put the function into a method called FindY which accepts an argument of type double and returns a double.

Turn in a printed copy of your source file.