Chapter 3

Top-Down Design with Functions

Figure 3.1 Edited Data Requirements and Algorithm for Conversion Program

```
/*
 * Converts distance in miles to kilometers.
 */

#include <stdio.h>                  /* printf, scanf definitions */
#define KMS_PER_MILE 1.609          /* conversion constant */

int
main(void)
{
      double miles;   /* input - distance in miles.          */
      double kms;     /* output - distance in kilometers     */

      /* Get the distance in miles.                          */

      /* Convert the distance to kilometers.                 */
         /* Distance in kilometers is
               1.609 * distance in miles.                    */

      /* Display the distance in kilometers.                 */

      return (0);
}
```


Figure 3.2 Outline of Program Circle

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

#include <stdio.h>
#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 */

      /* Calculate the area */
         /* Assign PI * radius * radius to area. */

      /* Calculate the circumference */
         /* Assign 2 * PI * radius to circum. */

      /* Display the area and circumference */

      return (0);
}
```


Figure 3.3 Calculating the Area and the Circumference of a Circle

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

#include <stdio.h>
#define PI 3.14159

int
main(void)
```

(continued)

Figure 3.3 Calculating the Area and the Circumference of a Circle (cont'd)

```
{
      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
```


Figure 3.4 Computing the Rim Area of a Flat Washer

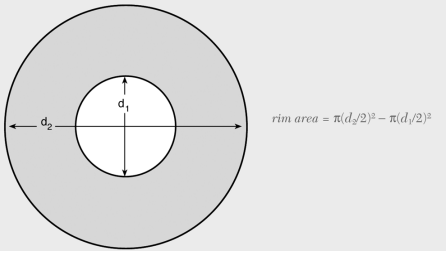

Figure 3.5 Flat Washer Program

```
/*
 * Computes the weight of a batch of flat washers.
 */

#include <stdio.h>
#define PI 3.14159

int
main(void)
{
      double hole_diameter; /* input - diameter of hole        */
      double edge_diameter; /* input - diameter of outer edge  */
      double thickness;     /* input - thickness of washer     */
      double density;       /* input - density of material used */
      double quantity;      /* input - number of washers made  */
      double weight;        /* output - weight of washer batch */
      double hole_radius;   /* radius of hole                  */
      double edge_radius;   /* radius of outer edge            */
      double rim_area;      /* area of rim                     */
      double unit_weight;   /* weight of 1 washer              */

      /* Get the inner diameter, outer diameter, and thickness.*/
      printf("Inner diameter in centimeters> ");
      scanf("%lf", &hole_diameter);
      printf("Outer diameter in centimeters> ");
      scanf("%lf", &edge_diameter);
      printf("Thickness in centimeters> ");
      scanf("%lf", &thickness);

      /* Get the material density and quantity manufactured. */
      printf("Material density in grams per cubic centimeter> ");
      scanf("%lf", &density);
      printf("Quantity in batch> ");
      scanf("%lf", &quantity);

      /* Compute the rim area. */
      hole_radius = hole_diameter / 2.0;
      edge_radius = edge_diameter / 2.0;
      rim_area = PI * edge_radius * edge_radius -
                 PI * hole_radius * hole_radius;

      /* Compute the weight of a flat washer. */
      unit_weight = rim_area * thickness * density;
```

(continued)

Figure 3.5 Flat Washer Program (cont'd)

```
      /* Compute the weight of the batch of washers. */
      weight = unit_weight * quantity;

      /* Display the weight of the batch of washers. */
      printf("\nThe expected weight of the batch is %.2f", weight);
      printf(" grams.\n");

      return (0);
}
```

```
Inner diameter in centimeters> 1.2
Outer diameter in centimeters> 2.4
Thickness in centimeters> 0.1
Material density in grams per cubic centimeter> 7.87
Quantity in batch> 1000

The expected weight of the batch is 2670.23 grams.
```


Figure 3.6 Function sqrt as a "Black Box"

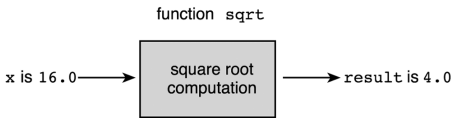

Figure 3.7 Square Root Program

```
/*
 * 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);
```

(continued)

Figure 3.7 Square Root Program (cont'd)

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

      /* Display the square root of the sum of the two numbers. */
      sum_sqrt = sqrt(first + second);
      printf("The square root of the sum of the two numbers is %.2f\n",
             sum_sqrt);

      return (0);
}
```

```
Enter the first number> 9.0
The square root of the first number is 3.00
Enter the second number> 16.0
The square root of the second number is 4.00
The square root of the sum of the two numbers is 5.00
```


Figure 3.8 Triangle with Unknown Side a

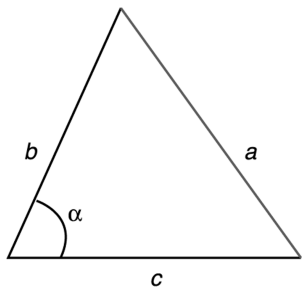

Figure 3.9 House and Stick Figure

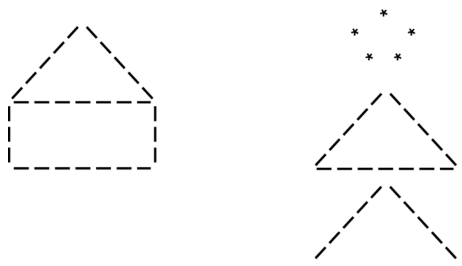

Figure 3.10 Structure Chart for Drawing a Stick Figure

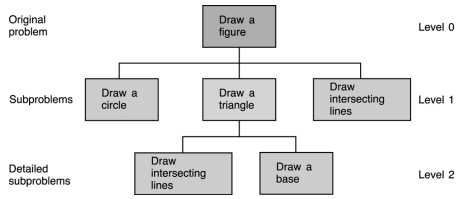

Figure 3.11 Function Prototypes and Main Function for Stick Figure

```
/*
 * Draws a stick figure
 */

#include <stdio.h>

/* function prototypes                                  */

void draw_circle(void);      /* Draws a circle          */

void draw_intersect(void);   /* Draws intersecting lines */

void draw_base(void);        /* Draws a base line       */

void draw_triangle(void);    /* Draws a triangle        */

int
main(void)
{
      /* Draw a circle.  */
      draw_circle();

      /* Draw a triangle.  */
      draw_triangle();

      /* Draw intersecting lines.  */
      draw_intersect();

      return (0);
}
```


Figure 3.12 Function draw_circle

```
/*
 * Draws a circle
 */
void
draw_circle(void)
{
      printf("   *  \n");
      printf(" *   *\n");
      printf("  * * \n");
}
```


Figure 3.13 Function draw_triangle

```
/*
 * Draws a triangle
 */
void
draw_triangle(void)
{
      draw_intersect();
      draw_base();
}
```


Figure 3.14 Program to Draw a Stick Figure

```
/* Draws a stick figure */

#include <stdio.h>

/* Function prototypes */
void draw_circle(void);          /* Draws a circle              */
void draw_intersect(void);       /* Draws intersecting lines    */
void draw_base(void);            /* Draws a base line           */
void draw_triangle(void);        /* Draws a triangle            */

int
main(void)
{

      /* Draw a circle.                */
      draw_circle();

      /* Draw a triangle.              */
      draw_triangle();

      /* Draw intersecting lines.      */
      draw_intersect();

      return (0);
}
```

(continued)

Figure 3.14 Program to Draw a Stick Figure (cont'd)

```
/*
 * Draws a circle
 */
void
draw_circle(void)
{
      printf("   *   \n");
      printf(" *   * \n");
      printf(" * *  \n");
}

/*
 * Draws intersecting lines
 */
void
draw_intersect(void)
{
      printf("  / \\  \n"); /* Use 2 \'s to print 1 */
      printf(" /   \\ \n");
      printf("/     \\\n");
}

/*
 * Draws a base line
 */
void
draw_base(void)
{
      printf("-------\n");
}

/*
 * Draws a triangle
 */
void
draw_triangle(void)
{
      draw_intersect();
      draw_base();
}
```


Figure 3.15 Flow of Control Between the main Function and a Function Subprogram

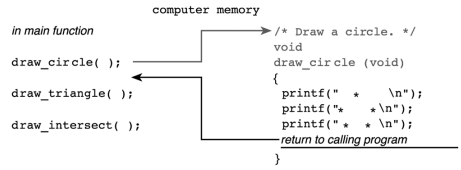

Figure 3.16 Function instruct and the Output Produced by a Call

```
/*
 * Displays instructions to a user of program to compute
 * the area and circumference of a circle.
 */
void
instruct(void)
{
      printf("This program computes the area\n");
      printf("and circumference of a circle.\n\n");
      printf("To use this program, enter the radius of\n");
      printf("the circle after the prompt: Enter radius>\n");
}
```

```
This program computes the area
and circumference of a circle.

To use this program, enter the radius of
the circle after the prompt: Enter radius>
```


Figure 3.17 Lego® Blocks

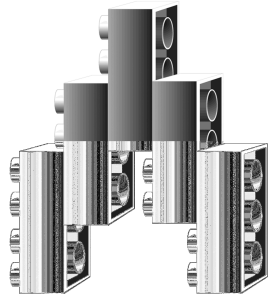

Figure 3.18 Function print_rboxed and Sample Run

```
/*
 * Displays a real number in a box.
 */

void
print_rboxed(double rnum)
{
      printf("***********\n");
      printf("*         *\n");
      printf("* %7.2f *\n", rnum);
      printf("*         *\n");
      printf("***********\n");
}
```

```
***********
*         *
*  135.68 *
*         *
***********
```


Figure 3.18 Function print_rboxed and Sample Run (cont'd)

```
***********
*         *
*  135.68 *
*         *
***********
```

Figure 3.19 Effect of Executing print_rboxed (135.68);

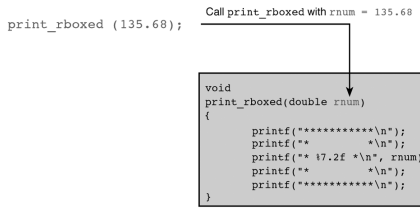

Figure 3.20 Function with Input Arguments and One Result

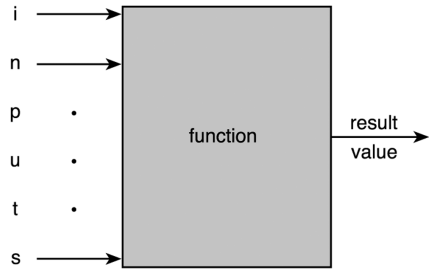

Figure 3.21 Functions find_circum and find_area

```
1.  /*
2.   * Computes the circumference of a circle with radius r.
3.   * Pre:  r is defined and is > 0.
4.   *       PI is a constant macro representing an approximation of pi.
5.   */
6.  double
7.  find_circum(double r)
8.  {
9.      return (2.0 * PI * r);
10. }
11.
12. /*
13.  * Computes the area of a circle with radius r.
14.  * Pre:  r is defined and is > 0.
15.  *       PI is a constant macro representing an approximation of pi.
16.  *       Library math.h is included.
17.  */
18. double
19. find_area(double r)
20. {
21.     return (PI * pow(r, 2));
22. }
```


Figure 3.22 Effect of Executing circum = find_circum (radius);

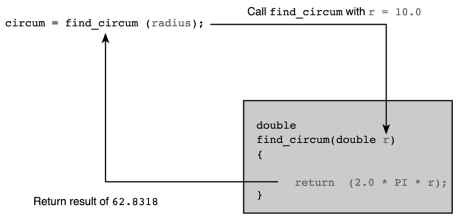

Figure 3.23 Function scale

```
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. }
```


Figure 3.24 Testing Function scale

```
1.  /*
2.   * Tests function scale.
3.   */
4.
5.  #include <math.h>
6.
7.  /* Function prototype */
8.  double scale(double x, int n);
9.
10. int
11. main(void)
```

(continued)

Figure 3.24 Testing Function scale (cont'd)

```
{
    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
```


Figure 3.25 Data Areas After Call scale(num_1, num_2);