Chapter Seven

Libraries and Interfaces

1

Libraries

- A *library* is a collection of *predefined functions* that perform specific operations
- The *standard libraries* are the libraries that should be provided by every C programming system
- Programmers can also build their *own libraries* using the mechanisms provided by C programming systems

2

Interfaces

- A *library interface* is the *boundary* between the *implementation* of a library and programs that *use* that library
- An *interface* represents a *shared understanding* about the nature of that boundary, providing both creators and users of a library with the *critical information* they need to know
- An *interface* gives *structure* to the *exchange of information* between the library and its users

3

An Example

- The **math** library defines the function **sqrt** to calculate a square root
- This function can be implemented by *Newton's algorithm*, *Taylor series expansion*, or *other algorithms*
- Knowing how to *call* the **sqrt** function and knowing how to *implement* it are both important skills

4

An Example

- The programmer (*implementer*) who implements a library cannot anticipate all the potential uses for that library
- The programmer (*client*) who calls functions in a library wouldn't know how to write them

5

Function Prototype

- For each function in the library, the client must know the following:
- Its *name*
- The *arguments* it requires and the *types* of those arguments
- The *type of result* it returns

6

Function Prototype

Client	interface	implementer
How a function is used	both sides agree on: function prototype	How a function works

7

Header Files

- C uses *header files* to provide a concrete realization of an *interface*
- The header file for a library contains the *prototypes* for all the functions *exported* by the library as well as associated *symbolic constants*, *macros*, and *types*

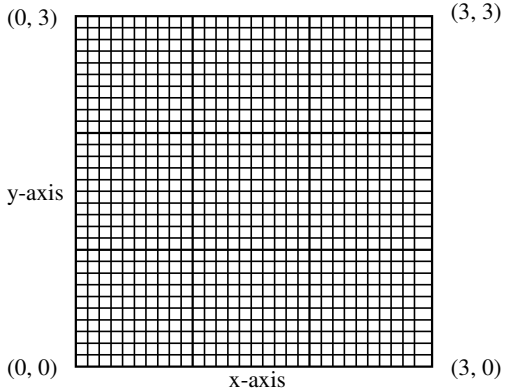
8

A Graphics Library

- When you start up the library, a *graphics window* is created on the screen and used as the drawing surface
- All drawing in the graphics window takes places on a conceptual *grid* of *points*
- *Points* are identified by specifying their position relative to the *origin*, which is the point at the *lower left corner* of the graphics window

9

Grid and Coordinates



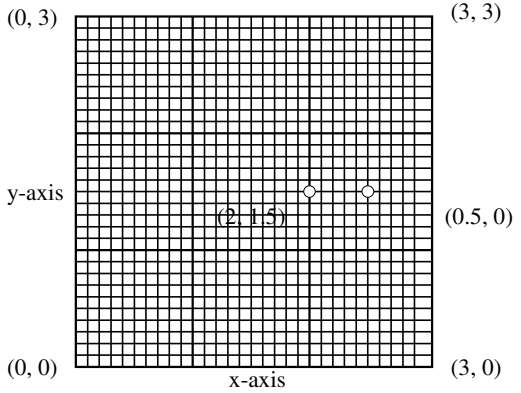
10

Coordinates

- *Absolute coordinates* specify a point in the window by giving its coordinates with respect to the origin
- *Relative coordinates* specify a point in the window by giving its coordinates with respect to the last position specified

11

Coordinates



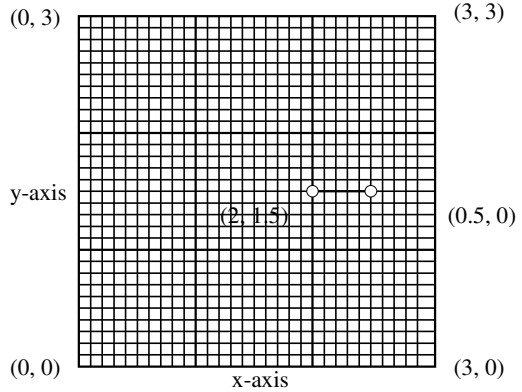
12

Drawing

- The *mental model* for the drawing process can be thought as a pen positioned and moved over the grid of points
- Each drawing process begins by positioning the pen to a particular position using the *absolute* coordinates
- You can then draw lines or arcs by moving the pen *relative to* its current position

13

Drawing



14

Functions

- The library contains the following functions:

initGraphics()	Initializes the library
movePen(x, y)	Moves the pen
drawLine(dx, dy)	Draws a line
drawArc(r, b, s)	Draws a arc
getWindowWidth()	Returns the width
getWindowHeight()	Returns the height
getCurrentX()	Returns x-coordinate
getCurrentY()	Returns y-coordinate

15

initGraphics

```
/*
 * Function: initGraphics
 * Usage: initGraphics();
 * ----------------------------
 * This function creates the graphics window on the
 * screen. The call to initGraphics must precede any
 * calls to other functions in the library and must also
 * precede any printf output. In most cases, the
 * initGraphics call is the first statement in the main.
 */

void initGraphics(void);
```

16

movePen

```
/*
 * Function: movePen
 * Usage: movePen(x, y);
 * ---------------------------
 * This function moves the current point to the
 * position (x, y), without drawing a line. The model
 * is that of the pen being lifted off the graphics
 * window surface and then move to its new position.
 */

void movePen(double x, double y);
```

17

drawLine

```
/*
 * Function: drawLine
 * Usage: drawLine(dx, dy);
 * ----------------------------
 * This function draws a line extending from the
 * current point by moving the pen dx inches in the x
 * axies and dy inches in the y axies. The final position
 * becomes the new position.
 */

void drawLine(double dx, double dy);
```

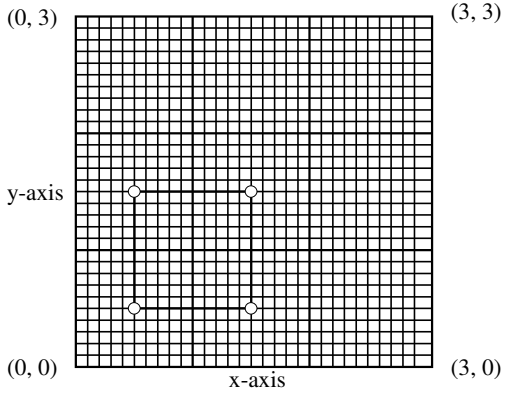
18

An Example

```
main()
{
   initGraphics();
   movePen(0.5, 0.5);
   drawLine(0.0, 1.0);
   drawLine(1.0, 0.0);
   drawLine(0.0, -1.0);
   drawLine(-1.0, 0.0);
}
```

19

An Example



20

drawArc

```
/*
 * Function: drawArc
 * Usage: drawArc(r, b, s);
 * ----------------------------
 * This function draws a circular arc, which always
 * begins at the current point. The arc has radius r, and
 * begins at the angle b and extends an angle of s. The
 * The angles are measured in degrees counterclockwise
 * from the 3 o'clock position along the x-axies.
 */

void drawArc(double r, double b, double s);
```

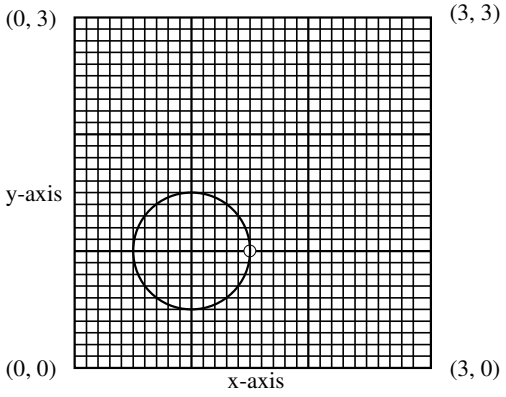
21

An Example

```
main()
{
   initGraphics();
   movePen(1.5, 1.0);
   drawArc(0.5, 0, 360);
}
```

22

An Example



23

getWindowWidth & Height

```
/*
 * Functions: getWindowWidth, getWindowHeight
 * Usage: width = getWindowWidth();
 *        height = getWindowHeight();
 * ------------------------------------------------------------
 * These functions return the width and height of the
 * graphics window, in inches.
 */

double getWindowWidth(void);
double getWindowHeight(void);
```

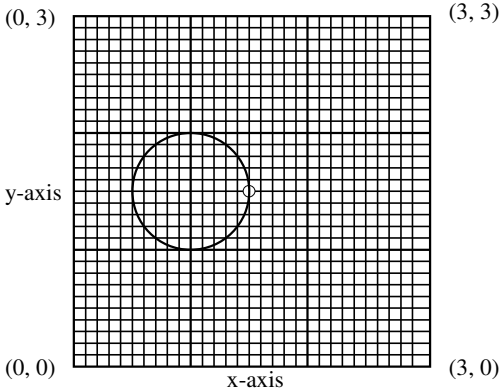
24

An Example

```
main()
{
   initGraphics();
   movePen(getWindowWidth()/2, getWindowHeight()/2);
   drawArc(0.5, 0, 360);
}
```

25

An Example



26

getCurrentX & Y

```
/*
 * Functions: getCurrentX, getCurrentY
 * Usage: x = getCurrentX();
 *        y = getCurrentY();
 * ----------------------------------------------
 * These functions return the current x and y positions.
 */

double getCurrentX(void);
double getCurrentY(void);
```

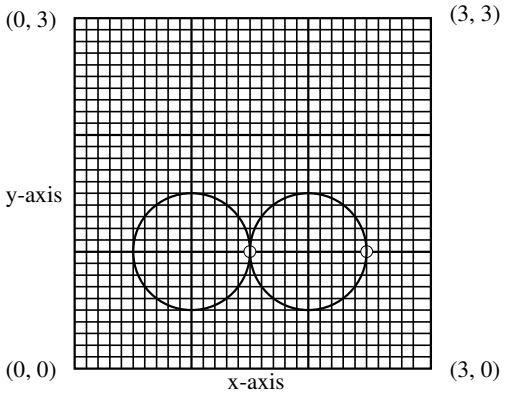
27

An Example

```
main()
{
   initGraphics();
   movePen(1.5, 1.0);
   drawArc(0.5, 0, 360);
   movePen(getCurrentX() + 1, getCurrentY() + 1);
   drawArc(0.5, 0, 360);
}
```

28

An Example



29

Documentation

```
/*
 * File: graphics.h
 * ----------------
 * This interface provides …
 */

/*
 * Overview
 * --------
 * This library provides …
 */
```

30

Build Your Own Tools

- Based on the simple functions provided in a library, you can construct *more sophisticated* functions
- For example, you can construct functions that draw boxes and circles
- You can build a *new library* that consists of these new functions and provides a *higher level of abstraction*

31

New Functions

- The following *new* functions are constructed

drawBox(x, y, w, h)	Draw a box
drawCenteredCircle(x, y, r)	Draw a circle
adjustPen(dx, dy)	Move a pen
drawLineTo(x, y)	Draw a line

- These functions can be *reused* over and over again

32

drawBox

```
/*
 * Function: drawBox
 * Usage: drawBox(x, y, width, height);
 * -----------------------------------------
 * This function draws a rectangle of the given width
 * and height with its lower left corner at (x, y).
 */

void drawBox(double x, double y, double w, double h);
```

33

drawBox

```
void drawBox(double x, double y, double w, double h)
{
   movePen(x, y);
   drawLine(0, h);
   drawLine(w, 0);
   drawLine(0, -h);
   drawLine(-w, 0);
}
```

34

drawCenteredCircle

```
/*
 * Function: drawCenteredCircle
 * Usage: drawCenteredCircle(x, y, radius);
 * ------------------------------------------------
 * This function draws a circle of the given radius
 * with its center at (x, y).
 */

void drawCenteredCircle(double x, double y, double radius);
```

35

drawCenteredCircle

```
void drawCenteredCircle(double x, double y, double radius)
{
   movePen(x + radius, y);
   drawArc(radius, 0, 360);
}
```

36

adjustPen

```
/*
 * Function: adjustPen
 * Usage: adjustPen(dx, dy);
 * ------------------------------------------
 * This function adjusts the current point by moving it dx
 * inches from its current x coordinate and dy inches from
 * its current y coordinate. No line is actually drawn.
 */

void adjustPen(double dx, double dy);
```

37

adjustPen

```
void adjustPen(double dx, double dy)
{
   movePen(getCurrentX() + dx, getCurrentY() + dy);
}
```

38

drawLineTo

```
/*
 * Function: drawLineTo
 * Usage: drawLineTo(x, y);
 * ------------------------------------------
 * This function is like drawLine, except that it uses the
 * absolute coordinates of the endpoint rather than the
 * relative displacement from the current point.
 */

void drawLineTo(double x, double y);
```

39

drawLineTo

```
void drawLineTo(double x, double y)
{
   drawLine(x - getCurrentX(), y - getCurrentY());
}
```

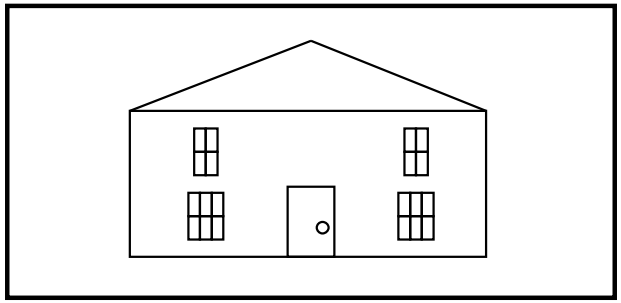
40

Solving a Larger Problem

- The best way to approach a large programming problem is to use the strategy of *stepwise refinement* to break the entire problem down into a set of simpler ones

41

Solving a Larger Problem



42

Symbolic Quantities

```
#define    HouseHeight          2.0
#define    HouseWidth           3.0
#define    AtticHeight          0.7
#define    DoorHeight           0.7
#define    DoorWidth            0.4
#define    DoorKnobRadius       0.03
#define    DoorKnobInset        0.07
#define    PaneHeight           0.25
#define    PaneWidth            0.2
#define    FirstFloorWindows    0.3
#define    SecondFloorWindows   1.25
```

43

main

```
main( )
{
   double cx, cy;

   initGraphics( );
   cx = getWindowWidth( ) / 2;
   cy = getWindowHeight( ) / 2;
   drawHouse(cx – HouseWidth/2,
             cy – (HouseHeight + AtticHeight)/2);
}
```

44

drawHouse

```
void drawHouse(double x, double y)
{
   drawOutline(x, y);
   drawDoor(x+(HouseWidth-DoorWidth)/2, y);
   drawWindows(x, y);
}
```

45

drawOutline

```
void drawOutline(double x, double y)
{
   drawBox(x, y, HouseWidth, HouseHeight);
   drawTriangle(x, y+ HouseHeight,
                HouseWidth, AtticHeight);
}
```

46

drawTriangle

```
void drawTriangle(double x, double y, double b, double h)
{
   movePen(x, y);
   drawLine(b, 0);
   drawLine(-b / 2, h);
   drawLine(-b / 2, -h);
}
```

47

drawDoor

```
void drawDoor(double x, double y)
{
   drawBox(x, y, DoorWidth, DoorHeight);
   drawCenteredCircle(x + DoorWidth - DoorKnobInset,
                      y + DoorHeight /2, DoorKnobRadius);
}
```

48

drawWindows

```
void drawWindows(double x, double y) {
   double xleft, xright;
   xleft = x + HouseWidth * 0.25;
   xright = x + HouseWidth * 0.75;
   drawGrid(xleft-PaneWidth*1.5, y+FirstFloorWindows,
            paneWidth, PaneHeight, 3, 2);
   drawGrid(xright-PaneWidth*1.5, y+FirstFloorWindows,
            paneWidth, PaneHeight, 3, 2);
   drawGrid(xleft-PaneWidth, y+SecondFloorWindows,
            paneWidth, PaneHeight, 2, 2);
   drawGrid(xright-PaneWidth, y+SecondFloorWindows,
            paneWidth, PaneHeight, 2, 2);
}
```

49

drawGrid

```
void drawGrid(double x, double y, double w, double h,
              double c, double r)
{
   int i, j;

   for (i = 0; i < c; i++) {
      for (j = 0; j < r; j++) {
         drawBox(x + i * w, y + j * h, w, h);
      }
   }
}
```

50