SOURCE CODE EXAMPLES: C

This appendix contains the code examples referred to in the main text. There is a version of the appendix for each language. This is the C version.

#1 EXAMPLE FOR THE CALLING DIAGRAM

```
int main() {
  function_one();
  function_two();
}

int function_one() {
  function_three();
}

int function_two() {
  function_four();
  function_one();
  function_two();
}

int function_three() {}

int function_four() {
  function_three();
}
```

#2 SMALL EXAMPLE FOR HAND SIMLUATION

```
  char v1 = 10;
  int v2 = v1 * 100;

  while (v2 > 500) {
    printf("Excess over 500 = %d\n", v2 - 500);
    v2 = --v1 * 100;
  }
```

#3 EXAMPLE FOR SCOPE REGIONS

```
void A() {
  int B;
}

void E() {
  int C;
   {
      int D;
   }
}
```

#4 LARGE EXAMPLE FOR HAND SIMULATION

```
#include <stdio.h>

int g1 = 100, g2 = 20;

void f1(int xx) {
  int w = 10;
  w = w + g1;
  printf("%d", w / xx);
}

int f2() {
   int w = 5;
   g1 = g2 + w;
   return w + 2;
}

int main() {
   int r;
   r = f2();
   f1(r);
   printf("%d", g1);
   return 0;
}
```

#5 TOP-DOWN DESIGN EXAMPLE

```
/**********************************
 The paint estimator
**********************************/

#include <stdio.h>
const float paintCost = 12.5;
const float height = 8.0;

float costOfPainting, length, width,
      coverage, area; /* global variables */

void inputValues() {
   printf("Type in length, width, coverage: ");
   scanf("%f%f%f", &length, &width, &coverage);
}

void calculateArea() {
   area = 2 * (length + width) * height + length * width;
}

void calculateCostFromArea() {
   costOfPainting = area / coverage * paintCost;
}

void calculateCost() {
   calculateArea();
   calculateCostFromArea();
}
void displayCost() {
   printf("Cost of painting a room is %f\n",
           costOfPainting);
}
int main() {
   inputValues();
   calculateCost();
   displayCost();
}
```

#6 TOP-DOWN DESIGN EXAMPLE, REWORKED

```
/**********************************
 The paint estimator
**********************************/

#include <stdio.h>
const float paintCost = 12.5;
const float height = 8.0;

float costOfPainting, length, width, coverage;

void inputValues() {
   printf("Type in length, width, coverage: ");
   scanf("%f%f%f", &length, &width, &coverage);
}
float calculateArea() {
   return 2 * (length + width) * height + length * width;
}

float calculateCostFromArea() {
   return area / coverage * paintCost;
}

void calculateCost(float l, float w, float c) {
   float area;

   area = calculateArea(l, w);
   calculateCostFromArea(area, c);
}
void displayCost(float l, float w, float c, float cp) {
   printf("Cost of painting a room %f by %f with\n
           paint of coverage %f is %f\n",
           l, w, c, cp);
}

int main() {
   float costOfPainting;

   inputValues();
   costOfPainting = calculateCost(length, width, coverage);
   displayCost(length, width, coverage, costOfPainting);
}
```

#7 THE MAIN PROGRAM FOR THE TOP-DOWN DESIGN EXAMPLE

```
int main() {
   float costOfPainting, length, width, coverage;

   length = inputLength();
   width = inputWidth();
   coverage = inputCOverage();
   calculateCost(length, width, coverage);
   displayCost(length, width, coverage, costOfPainting);
}
```

#8 THE PAINT ESTIMATOR DESIGN IN CLASSES

Since C does not have classes, we must simulate the encapsulation of functions through naming conventions and the passing of a pointer to the "object" for every call to a method. We can, however, encapsulate the data in structures, even though we cannot make it private.

```
typedef struct Room {
  float Width, Length;  // in ft.
  float Height;   // in ft.
} Room;

typedef struct Paint {
  float CostOfPaint;// in $/gallon
  float Coverage;       // in sq.ft./gallon
} Paint;

typedef struct Estimator {
  Room  room;
  Paint paint;
  float CostOfPainting; // in $
} Estimator;
```

#9 THE PAINT ESTIMATOR DESIGN WITH CLASS METHODS

The methods will be functions whose name begin with the "class" name, i.e. the struct type. There is no need for accessor methods or the constructor in this case, since we can use the variable fields in the structs directly. The constructor becomes an initializer function.

```
/************** Room methods *********************/
void RoomRoom(Room* room) { /* the constructor */
  room->Height = 8.0;
}

void RoomInput(Room* room) {
  printf("Type width and length: ");        /* input method prompts */
  scanf("%f", &room->Width, &room->Length); /* the user             */
}

/************** Paint methods *********************/
void PaintPaint(Paint* paint) {
    paint->CostOfPaint = 12.5  /* a constructor to set the constant */
}
```

```
void PaintInput(Paint* paint) { /* input method */
    printf("Type coverage: ");
    scanf("%f", &paint->Coverage);
}

/************** Estimator methods ********************/
void EstimatorEstimator(Estimator* estimator) {
  RoomRoom(&estimator->room);
  PaintPaint(&estimator->paint);
}

void EstimatorEstimate(Estimator* estimator); /* function defined below */

void EstimatorDisplay(Estimator* estimator) {
  printf("Cost is %f\n", estimator->CostOfPainting);
}

void EstimatorRun(Estimator* estimator);      /* function defined below */
```

#10 THE PAINT ESTIMATOR TOP LEVEL METHODS

```
void EstimatorEstimate(Estimator* estimator) {
  float wallArea, ceilingArea;

  wallArea =
    2 * (estimator->room.Width + estimator->room.Length) *
         estimator->room.Height;
  ceilingArea = estimator->room.Width * estimator->room.Length;
  estimator->CostOfPainting =
     (wallArea + ceilingArea) / estimator->paint.Coverage *
       estimator->paint.CostOPaint;
}

void EstimatorRun(Estimator* estimator) {
  RoomInput(&estimator->room);
  PaintInput(&estimator->paint);
  EstimatorEstimate(estimator);
  EstimatorDisplay(estimator);
}


int main() {
  Estimator estimator;  /* make an estimator object */
  EstimatorEstimator(&estimator); /* initialize it */
  EstimatorRun(&estimator);
}
```

#11 SIMPLE EXAMPLE FOR DEBUGGING

```
while (num != 9999) {
  total += num;
  scanf("%d", &num);
```

```
}
```

#12 DEBUGGING WITH TRACE ADDED

```
while (num != 9999) {
  printf("Trace total: %d\n", total);
  total += num;
  scanf("%d", &num);
}
```

#13 ADDING AN ASSERTION

Add a line with #include <assert.h> at the top of the file.

```
assert(i >= 0 && i <= 9);
```

#14 AN ASSERTION IN STRAIGHT CODE

```
if (x < 0 || x > 9) {
  printf("Error: x is out of range (%d)\n", x);
  exit(1);
}
```

#15 A DEBUGGING EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  int c, lastc, totalIEs = 0, totalEIs = 0;

  printf("Type any number of lines followed by EOF\n");

  while ((c = getchar()) != EOF) {
    c = tolower(c);
    if (c == 'i' && lastc == 'e')
      ++totalEIs;
    else if (c == 'e' && lastc == 'i')
      ++totalIEs;
    else
      lastc = c;
  }
  printf("\nthe number of IEs is %d\n", totalIEs);
  printf("\nand the number of EIs = %d\n", totalEIs);
}
```

#16 THE EXAMPLE WITH TRACE ADDED

```
int main() {
  int c, lastc, totalIEs = 0, totalEIs = 0;

  printf("Type any number of lines followed by EOF\n");

  while ((c = getchar()) != EOF) {
    c = tolower(c);
    if (c == 'i' && lastc == 'e')
      ++totalEIs;
    else if (c == 'e' && lastc == 'i')
      ++totalIEs;
    else
      lastc = c;
    printf(
    "c = %c lastc = %c totalIEs = %d totalEIS = %d\n",
       c, lastc, totalIEs, totalEIs);
  }
```

#17 THE EXAMPLE WITH ASSERTION ADDED

```
else
      lastc = c;
  /*****************************************/
    assert(lastc == c);
  /*****************************************/
  }
  ...
```

#18 THE EXAMPLE ASSERTION AND TRACE

```
...
    else
      lastc = c;
  /*//////   a trace statement   ///////*/
    printf("lastc = %c, c = %c\n", lastc, c);
  /*//////////////////////////////////////*/
  /*******   an assertion  *************/
  /*  assert(lastc == c);              */
  /************************************/
  }
```

#19 THE EXAMPLE DEBUGGED

```
...
#include <stdio.h>
#include <stdlib.h>

int main() {
  int c, lastc, totalIEs = 0, totalEIs = 0;

  printf("Type any number of lines followed by EOF\n");

  while ((c = getchar()) != EOF) {
    c = tolower(c);
    if (c == 'i' && lastc == 'e')
      ++totalEIs;
    else if (c == 'e' && lastc == 'i')
      ++totalIEs;
    /*else*/
    lastc = c;
  }
}
```