CSE 142  
Computer Programming I  

Functions and Design

Overview

- Design process
- Functional decomposition
- Top down vs. bottom up
- Graphics primitives

Drawing a House

- House with windows and door

Drawing a (Similar) House

- House similar to the previous one

Draw House (Pseudo-code)

draw_house (color, ll_x, ll_y, num_windows)

draw body as a colored rectangle

draw roof as a colored triangle

if num_windows is one

draw door

draw window

if num_windows is two

draw door

draw window

draw window
**Functional Decomposition**

- Draw House
  - Draw Body
  - Draw Roof
  - Draw Door
  - Draw Window

  This is a “calling tree” or “static call graph.”
  Each function is shown, with an arrow down to each function called.

**Analysis to Design to Programming**

- Analyze the problem
- Then design a “big-picture” solution
  A functional decomposition shows how the pieces fit together
- Then design individual functions
  May depend on low-level (“primitive”) functions available
- Final programming may be very detailed

**Top Down or Bottom Up?**

- Which approach are we following with DrawHouse?

  **Answer:** Generally, top down. But we have to look ahead and know what low level functions will be available.
  Eventually, there will be graphics programming to do. Fortunately, most systems supply a library of graphics “primitives.”

**Top Down vs. Bottom Up**

- Sometimes designers start from the big picture
  Gradually work down to smaller pieces and then to fine details
  Called the “top down approach”
- Sometimes people start with small pieces
  Figure out how they can fit together pieces to solve ever larger and larger problems
  Called the “bottom up approach”

**Graphics Primitive**

- Typical functions: clearscreen, draw circle, rectangle, line, ellipse, etc.
- Typical parameters: location, color, fill, etc.
- Requires a coordinate system

  \[(0, 0) \quad (a, b)\]
Typical 'rectangle' and 'line'

```c
void rectangle (int color, int x1, int y1, int x2, int y2);
void line (int x1, int y1, int x2, int y2);
```

Big Picture Again

Fill in the pieces one at a time

Window Constants

Our analysis of how to describe a window

```c
void draw_window(int x, int y)
{ /* (x,y) is the lower left corner of the window */
    rectangle( WHITE, x, y, x + WIN_W, y + WIN_H);
    line( x+MID_X, y, x + MID_X,   y + WIN_H);
    line( x,y + MID_Y, x + WIN_W,  y + MID_Y);
}
```

Keep Filling in Pieces

Keep filling in pieces

```c
void draw_window(int x, int y)
{ /* (x,y) is the lower left corner of the window */
    rectangle( WHITE, x, y, x + WIN_W, y + WIN_H);
    line( x+MID_X, y, x + MID_X,   y + WIN_H);
    line( x,y + MID_Y, x + WIN_W,  y + MID_Y);
}
```

Implementing Independently:

Preconditions and Postconditions

Each function in the decomposition has a “contract”:
- precondition: what the caller must ensure
- postcondition: what the function promises

This gives you a way to assume that even unwritten functions work!
- establish the preconditions
- assume the postconditions!
**draw_house: preconds and postconds**

```c
void draw_house (int color, int ll_x, int ll_y, int windows) {
    preconds: what are they?
    postconds: what are they?
}
```

**Draw House (Gory Detail I)**

```c
void draw_house (int color, int ll_x, int ll_y, int windows) {
    int roof_ll_x, roof_ll_y;
    /* Draw Body */
    draw_body (color, ll_x, ll_y);
    /* Draw Roof */
    roof_ll_x = ll_x - OVERHANG;
    roof_ll_y = ll_y + BODY_HEIGHT;
    draw_roof (color, roof_ll_x, roof_ll_y);
}
```

**Draw House (Gory Detail II)**

```c
if (windows == 1) {
    draw_door (ll_x + DOOR_OFFSET_1, ll_y);
    draw_window (ll_x + WINDOW_OFFSET_1, ll_y + WINDOW_RAISE);
} else if (windows == 2) {
    draw_door (ll_x + DOOR_OFFSET_2, ll_y);
    draw_window (ll_x + WINDOW_OFFSET_2A, ll_y + WINDOW_RAISE);
    draw_window (ll_x + WINDOW_OFFSET_2B, ll_y + WINDOW_RAISE);
}
```

**Next Step: A Neighborhood**

We could write 6 different functions...

*Smarter* - call 1 function 6 times...

**Summary of Functional Decomposition**

Look for **common elements** (similarities)

Parameterize for **special features** (differences)

Determine which functions will **use** others

Draw a graph to show their relationships
QOTD: A Study in Decomposition

Which of these functions will include a loop?
Which will need to read data (scanf)/write data (printf)?
Which of them (really) returns a Boolean value?
Which ones might call "exit"?