16.070 Introduction to Computers and Programming

February 14 | **Recitation 2** | **Spring 2002**

Topics:

Proper Homework Format

- Turn in requirements for software design problems
- Special formatting guidelines for homework
- Programming guide

Steps for Programming Algorithms

- Problem Statement
- Input/Output Description
- Hand Example
- Algorithm Development
- Testing
- Maintenance

Turn in requirements for software design problems

When doing the homework you should follow the Spiral Model of the Software Design Process. This model calls for you to perform each of the steps at least twice. We will require that you only turn in the final version of your work for each stage. The material that must be turned in for each stage is listed in the example that we will step through today. The format of each item can be found in the next section.

Each Problem Set will contain Turn In requirements associated with each of its problems. These requirements should be adhered to. In general however, you will be required to turn in at least your program code (hardcopy) and a screen dump of its output. Some problems might require you to draw up a flow chart or write pseudo code only. Always use some common sense too, turning in information that clarifies your answers whenever you feel that its necessary.

Step 1: Problem Statement

This portion of the software process is done for you in the form of the problem statement. Here, the concepts of the software engineering project are described.

Example
Design a program that inputs integer headings between –360 and +360 degrees (counterclockwise positive with respect to 0 degrees pointing from the origin to the right) and determines the smallest angle between them. The following output could be observed during a sample run.

```
Enter heading 1 (in degrees): 15
Enter heading 2 (in degrees): 20

The smallest angle between the headings is: 5 degrees

Press any key to continue
```

Step 2: Input/Output Description[1]

1. *Detailed Requirements;* list that contains each individual software requirement

2. *Analysis of Requirements*; list all of the inputs, outputs, constraints, and formulas that are required for the software

Example

Detailed Requirements

- Prompt the user to enter the two headings, in degrees. Check for valid input according to problem constraints.
- Determine the angle between the headings
- Determine if that is the smallest possible angle between the headings
- Display the smallest possible angle between the headings

Analysis

- Inputs: two headings
- Outputs: smallest angle between the two headings
- Formulas: for an angle < 0: Angle + 360 = equivalent angle
  for headings between 0 and 360 degrees (inclusive), the angle between two headings is the larger heading – the smaller heading
  Two headings always have two possible angle separations: *x*, and 360 - *x*

Step 3: Hand Example *(Test Planning Phase from Lecture #2)*

Test Cases
Generate sample input headings and manually calculate the angles you expect to be output by your program. Be sure to test normal cases and boundary conditions (we do).

Heading 1 (degrees)	Heading 2 (degrees)	Smallest Angle (degrees)
0	90	90
90	270	180
355	5	10
-30	350	20
5	5	0

Step 4: Algorithm Development

An algorithm can be broken up into conceptually different steps that are interrelated. Together these steps make up the whole of the algorithm. Each step may have its own algorithm.

1. *Decomposition Outline;* list that contains the names of all the steps you intend to use and the problems that those steps solve

2. *Algorithms for Decomposition Steps*; write out the algorithms that you will use to solve the problem in each of the steps (Pseudocode, flow chart/state diagrams – next week)

[1] often called Requirements

Decomposition Outline
You should write out Pseudo-Code for each of the algorithms you intend to use. Pseudo-code (required for our example) should contain a description of each of the algorithm steps written out in English. The vocabulary that you use to write your pseudo-code should be limited. It may contain words such as:

begin, end, if, then, while, print, read, etc…

Each pseudo-code statement should be written on a separate line and if a statement takes more than one line, the continuation lines should be indented. Different blocks of pseudo-code should be separated by blank lines from one and other.

Example

- **get_user_input** prompt user for a number reads in value
- **check_input_validity** checks that input is between –360 and 360 degrees
- **normalize_headings** recalculate both headings between 0 and 360 degrees
- **calculate_angle_between** calculate the angle between the normalized headings
- **find_smallest_angle** calculates the smallest angle between the headings
- **display_results** displays the smallest angle between the input headings

Algorithms

```
• get_user_input
  PRINT "Enter heading 1 (in degrees):"
  READ number Heading1
  PRINT "Enter heading 2 (in degrees):"
  READ number Heading2

• check_input_validity
  Error = FALSE
  IF (Heading1 < -360) OR (Heading1 > 360) THEN
     PRINT "First input invalid."
     Error  = TRUE
  IF (Heading2 < -360) OR (Heading2 > 360) THEN
     PRINT "Second input invalid."
     Error  = TRUE

  IF Error = FALSE  THEN

  • normalize_headings
    IF Heading1 < 0 THEN
       Heading1 = Heading1 + 360
    IF Heading2 < 0 THEN
       Heading2 = Heading2 + 360

  • calculate_angle_between
    IF Heading1 >= Heading2 THEN
       AngleBetween = Heading1 – Heading2
    ELSE IF Heading1 < Heading2 THEN
       AngleBetween = Heading2 – Heading1
    ELSE
       PRINT "Unknown angle condition."

  • find_smallest_angle
    IF AngleBetween > (360 – AngleBetween) THEN
       AngleBetween = 360 - AngleBetween

  • display_results
    print "The smallest angle between the two headings is AngleBetween degrees"
```

Step 4 (continued): Programming

1. Source code printout
2. Screen dump of sample run
3. Electronic submission of source code, when required

Example

```
/********************************/
/* This program determines the  */
/*   smallest angle between two */
/*   headings.                  */
/* by Louis Breger, Feb 2002    */
/* For 16.070 Recitation 2      */
/********************************/

#include <stdio.h>

#define TRUE 1   /* standard symbolic constants for Boolean values */
#define FALSE 0

/*** Main Function ***/
int main(void)  {

  /* Variable Declarations */
  int Heading1 = 0; /* contains first user heading */
  int Heading2 = 0; /* contains second user heading */

  /* both heading variables are measured in degrees using a counterclockwise
     positive coordinate system with 0 degrees indicating a heading to the
     right
  */

  int AngleBetween = 0;  /* contains the angle between the headings */
                         /* measured in degrees                      */

  int Error = FALSE;  /* indicates whether an input error occurred */
                      /* FALSE == No Error,    TRUE == Error       */


  /***** get_user_input *****/

  /* Prompt user for two headings */
  printf("Enter Heading 1 (in degrees): ");
  scanf("%d", &Heading1);

  printf("Enter Heading 2 (in degrees): ");
  scanf("%d", &Heading2);


  /***** check_input_validity *****/

  if ((Heading1 < -360) || (Heading1 > 360)) {
    printf("\nFirst input is invalid.\n\n");
    Error = TRUE;
  } /* end if Heading1 invalid */

  if ((Heading2 < -360) || (Heading2 > 360)) {
    printf("\nSecond input is invalid.\n\n");
    Error = TRUE;
  } /* end if Heading2 invalid */
  if (Error == FALSE) {

    /***** normalize_headings *****/

    if (Heading1 < 0) {
      Heading1 = Heading1 + 360;
    } /* end if Heading1 is negative */

    if (Heading2 < 0) {
      Heading2 = Heading2 + 360;
    } /* end if Heading2 is negative */


    /***** calculate_angle_between *****/

    /* calculate the angle between the
       normalized headings              */

    if (Heading1 >= Heading2) {
      AngleBetween = Heading1 – Heading2;
    } /* end if Heading1 is larger than Heading 2 */

    else if (Heading1 < Heading2) {
      AngleBetween = Heading2 – Heading1;
    } /* end else if Heading2 is larger than Heading 2 */

    else {
      printf(“\n\nError, not all angle conditions accounted for\n\n”);
    } /* end else error condition */


    /***** find_smallest_angle *****/

    if ( AngleBetween > (360 – AngleBetween) ) {
      AngleBetween = 360 – AngleBetween;
    } /* end if AngleBetween was not the smallest angle possible */


    /***** display_results *****/

    printf(“\nThe smallest angle between “);
    printf(“the two headings is %d degrees.\n\n”, AngleBetween);

  } /* end if the inputs were valid */

  return 0;
} /* end main */
```

Step 5: Testing *(Test Execution Phase from Lecture #2)*

Screendumps of the test cases and their outputs.

Example

```
Enter heading 1 (in degrees): 350
Enter heading 2 (in degrees): -20

The smallest angle between the two headings is 10 degrees.

Press any key to continue
```

Step 6: Maintenance Phase

For this example: This is not a large program, but a correct coding style, keeping readability in mind is required. Remember to add comments and indent your code. This enables a different programmer to follow your code and make additions or changes. (See extract of the programming guide on the next page.) For our purposes, you don't need to explicitly include a discussion about maintenance.

Programming Guide

Readability of Code

• A comment should come before each block of code that describes its function
• Insert blank lines between different blocks of code
• One statement per line is allowed (a ; should be followed by an enter)
• Meaningful variable names should be used
• Avoid running statements over multiple lines
• Use whitespace in expressions (before and after operators)

Homework 2 Design Problem Notes

In the design problem (ps2p5), you are asked to design a program that determines whether an underwater beacon is visible to a submarine. When you are planning your algorithm, consider the parts of the problem carefully… is there any way to separate the problem into pieces that are more readily solved?

Consider how you would find the angle between two points. Are there any exceptional cases to be considered? Boundary conditions to be tested? Hint: what would happen if you divided a number by 0? What if you add a degree to 360 degrees? There are many such conditions to consider in this problem and you are urged to test your program extensively.