Reading Assignment #4

Topic: C

Textbook: Problem Solving & Program Design in C
by Jeri R. Hanly and Elliot B. Koffman

Chapters (3, 4)

Top-Down Design with Functions
Selection Structures – if and switch statements
Flowcharts of if Statements with (a) Two Alternatives and (b) One Alternative
if Statement (One Alternative)

FORM: \[
\text{if (condition)} \\
\text{statement}_T;
\]

EXAMPLE: \[
\text{if (x > 0.0)} \\
\text{pos}_\text{prod} = \text{pos}_\text{prod} \times \text{x};
\]

INTERPRETATION: If \text{condition} evaluates to \text{true}, then \text{statement}_T is executed; otherwise, \text{statement}_T is skipped.
To test $x = 0$:
if ( $x == 0$ ) {
    pos_prod = pos_prod * x ;
}

Let’s say we have $x, y$ (two variables already declared), we want to order them in increasing order.
if ( $x > y$ ) {
    $x = y$ ;
    $y = x$ ;
}
if Statement to Order $x$ and $y$

```c
if (x > y) {
    temp = x;  /* Store old x in temp */
    x = y;    /* Store old y in x */
    y = temp; /* Store old x in y */
}
```

A step-by-step simulation of the execution of the `if` statement when $x$ is 12.5 and $y$ is 5.0. The table shows that `temp` is initially undefined (indicated by ?). Each line of the table shows the part of the `if` statement that is

<table>
<thead>
<tr>
<th>Statement Part</th>
<th>$x$</th>
<th>$y$</th>
<th>temp</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>if ($x &gt; y$){</td>
<td>12.5</td>
<td>5.0</td>
<td>?</td>
<td>12.5 &gt; 5.0 — true</td>
</tr>
<tr>
<td></td>
<td>temp = x;</td>
<td></td>
<td>12.5</td>
<td>Store old x in temp</td>
</tr>
<tr>
<td></td>
<td>x = y;</td>
<td>5.0</td>
<td></td>
<td>Store old y in x</td>
</tr>
<tr>
<td></td>
<td>y = temp;</td>
<td>12.5</td>
<td></td>
<td>Store old x in y</td>
</tr>
</tbody>
</table>
A block of statements to be executed:
if ( x > 0.0 ) {
    pos_prod = pos_prod * x ;
    printf("%f",pos_prod );
    scanf("%lf",&x ) ;
}

double temp ;
if ( x > y ) {
    temp = x ; /* store old x in temp */
    x = y ; /* store old y in x */
    y = temp ; /* store old x in y */
}
<table>
<thead>
<tr>
<th>Operator</th>
<th>Condition</th>
<th>English Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=</td>
<td><code>x &lt;= 0</code></td>
<td>x less than or equal to 0</td>
<td>1 (true)</td>
</tr>
<tr>
<td>&lt;</td>
<td><code>power &lt; MAX_POW</code></td>
<td>power less than MAX_POW</td>
<td>0 (false)</td>
</tr>
<tr>
<td>&gt;=</td>
<td><code>x &gt;= y</code></td>
<td>x greater than or equal to y</td>
<td>0 (false)</td>
</tr>
<tr>
<td>&gt;</td>
<td><code>item &gt; MIN_ITEM</code></td>
<td>item greater than MIN_ITEM</td>
<td>1 (true)</td>
</tr>
<tr>
<td>==</td>
<td><code>mom_or_dad == 'M'</code></td>
<td>mom_or_dad equal to 'M'</td>
<td>1 (true)</td>
</tr>
<tr>
<td>!=</td>
<td><code>num != SENTINEL</code></td>
<td>num not equal to SENTINEL</td>
<td>0 (false)</td>
</tr>
</tbody>
</table>

C Relational and Equality Operators and Sample Conditions
We have

char letter_1, letter_2, letter_3 ;

if ( letter_1 == 'B' ) {
    printf("Name starts with a B");
}

Ascii value of A = ‘A’

Ascii value of B = ‘B’

Ascii value of A = 32

Ascii value of a = ‘a’

Ascii value of b = ‘b’
<table>
<thead>
<tr>
<th>operand1</th>
<th>operand2</th>
<th>operand1 &amp;&amp; operand2 (read operand1 and operand2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonzero</td>
<td>nonzero</td>
<td>1</td>
</tr>
<tr>
<td>nonzero</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>nonzero</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

&& Operator

| operand1 | operand2 | operand1 || operand2 (read operand1 or operand2) |
|----------|----------|----------|---------------------------------------------|
| nonzero  | nonzero  | 1        |                                             |
| nonzero  | 0        | 1        |                                             |
| 0        | nonzero  | 1        |                                             |
| 0        | 0        | 0        |                                             |

|| Operator

<table>
<thead>
<tr>
<th>operand1</th>
<th>!operand1 (read not operand1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonzero</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

! Operator
Ex: Write a conditional test using letter_1, letter_2 and letter_3 variables from the last example for reading in nicknames to test whether the person’s identity is "Bob".

If (( letter_1 == ‘B’) && ( letter_2 == ‘o’) && ( letter_3 == ‘b’))
{
    printf("Hello Bob \n");
    printf("You have access\n.");
}

To handle upper and lower cases replace ( letter_1 == 'B' ) with

(( letter_1 == ‘B’) || (letter_1 == ‘b’))
if Statement (Two Alternatives)

FORM: \[
\text{if (condition)} \\
\quad \text{statement}_T; \\
\text{else} \\
\quad \text{statement}_F;
\]

EXAMPLE: \[
\text{if (} x \geq 0.0 \text{)} \\
\quad \text{printf("positive\n");} \\
\text{else} \\
\quad \text{printf("negative\n");}
\]

INTERPRETATION: If \text{condition} evaluates to \text{true}, then \text{statement}_T is executed and \text{statement}_F is skipped; otherwise, \text{statement}_T is skipped and \text{statement}_F is executed.
Structure Chart for Finding the Alphabetically First Letter

Structure Chart for Step 2 of the First Letter Problem
Selection Structures: if and switch Statements

Refined Algorithm

1. Get three letters for ch1, ch2, and ch3.
2. Save the alphabetically first of ch1, ch2, and ch3 in alpha_first.
   2.1 Save the alphabetically first of ch1 and ch2 in alpha_first.
      2.1.1 if ch1 precedes ch2
      2.1.2 alpha_first gets ch1.
      else
      2.1.3 alpha_first gets ch2.
   2.2 Save the alphabetically first of ch3 and alpha_first in alpha_first.
      2.2.1 if ch3 precedes alpha_first
      2.2.2 alpha_first gets ch3.
3. Display the alphabetically first letter.

Table 4.3 shows a trace of the algorithm for the data string THE. Each step is listed at the left in order of its execution. If an algorithm step changes the value of a variable, the table shows the new value. The effect of each step is described at the far right. For example, the table shows that the step

1. Get three letters for ch1, ch2, and ch3

stores the letters T, H, and E in the variables ch1, ch2, and ch3.

Trace of Algorithm for Program

<table>
<thead>
<tr>
<th>Algorithm Step</th>
<th>ch1</th>
<th>ch2</th>
<th>ch3</th>
<th>alpha_first</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Get three letters...</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>Gets the data</td>
</tr>
<tr>
<td>2.1.1 if ch1 precedes ch2</td>
<td>T</td>
<td>H</td>
<td>E</td>
<td></td>
<td>Is 'T' &lt; 'H'?</td>
</tr>
<tr>
<td>2.1.3 alpha_first gets ch2.</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td>Value is false 'H' is first so far</td>
</tr>
<tr>
<td>2.2.1 if ch3 precedes alpha_first</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Is 'E' &lt; 'H'?</td>
</tr>
<tr>
<td>2.2.2 alpha_first gets ch3.</td>
<td></td>
<td></td>
<td>E</td>
<td></td>
<td>Value is true 'E' is first</td>
</tr>
<tr>
<td>3. Display alpha_first.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Prints E is the first</td>
</tr>
</tbody>
</table>
Finding the Alphabetically First Letter

/* Finds and displays the alphabetically first letter */

#include <stdio.h>

int main (void)
{
   char chl, ch2, ch3; /* three letters (all uppercase or all lowercase) */
   char alpha_first; /* alphabetically first letter */

   /* Get three letters. */
   printf ("Enter three uppercase or three lowercase letters> ");
   scanf ("%c%c%c", &chl, &ch2, &ch3);

   /* Save the alphabetically first of ch1 and ch2 in alpha_first. */
   if (chl < ch2)
      alpha_first = ch1; /* ch1 comes before ch2 */
   else
      alpha_first = ch2; /* ch2 comes before ch1 */

   /* Save the alphabetically first of ch3 and alpha_first in alpha_first. */
   if (ch3 < alpha_first)
      alpha_first = ch3; /* ch3 comes before alpha_first */

   /* Display result. */
   printf ("%c is the first letter alphabetically. \n", alpha_first);

   return (0);
}

Enter three uppercase or three lowercase letters> EBK
B is the first letter alphabetically.
Multiple-Alternative Decision

SYNTAX: if (condition_1)
        statement_1
else if (condition_2)
        statement_2
        ...
        ...
else if (condition_n)
        statement_n
else
        statement_e

EXAMPLE: /* increment num_pos, num_neg, or num_zero depending on x */
        if (x > 0)
            num_pos = num_pos + 1;
        else if (x < 0)
            num_neg = num_neg + 1;
        else /* x equals 0 */
            num_zero = num_zero + 1;
if ( x > 0 ) {
    num_pos = num_pos + 1 ;
    printf("One more positive number \n");
}
else if ( x < 0 ) {
    num_neg = num_neg + 1 ;
    printf("One more negative number \n");
}
else {
    num_zero = num_zero + 1 ;
    printf("Another zero \n");
}
Function comp_tax

/*@  
* Computes the tax due on a tax table  
* Pre  :  salary is defined.  
* Post:  Returns the tax due for 0.0 <= salary <= 150,000.00;  
*       returns –1.0 if salary is outside the table range.  
*/

double
comp_tax (double salary)
{
    double tax;

    if (salary < 0.0)
        tax = -1.0;
    else if (salary < 15000.00)  /* first range */
        tax = 0.15 * salary;
    else if (salary < 30000.00)  /* second range */
        tax = (salary – 15000.00 * 0.18 + 2250.00;
    else if (salary < 50000.00)  /* third range */
        tax = (salary – 30000.00 * 0.22 + 5400.00;
    else if (salary < 80000.00)  /* fourth range */
        tax = (salary – 50000.00 * 0.27 + 11000.00;
    else if (salary < 150000.00)  /* fifth range */
        tax = (salary – 80000.00 * 0.33 + 21600.00;
    else
        tax = -1.0;

    return (tax);
}
**switch Statement**

SYNTAX:  
```
switch (controlling expression) {
  label set₁
    statements₁
    break;

  label set₂
    statements₂
    break;

  ...

  label setₙ
    statementsₙ
    break;

default:
  statementsₐ
}
```

EXAMPLE: /* Determine life expectancy of a standard light bulb */
```
switch (watts) {
  case 25:
    life = 2500;
    break;

  case 40:
  case 60:
    life = 1000;
    break;
  case 75:
  case 100:
    life = 750;
    break;

default:
    life = 0;
}
```
int watts, life;

/* Determine average life expectancy of a standard light bulb */
switch (watts) {
  case 25:
    life = 2500;
    break;

  case 40:
  case 60:
    life = 1000;
    break;

  case 75:
  case 100:
    life = 750;
    break;

  default:
    life = 0;
}
printf("Life expectancy of %d-watt bulb: ", watts);
if (life == 0)
  printf("unknown\n");
else
  printf("%d hours\n", life);
char grade;
int units;

scanf("%d", &units);

switch (grade) {
    case 'A':
        credit_pts = credit_pts + (4 * units);
        break;
    case 'B':
        credit_pts = credit_pts + (3 * units);
        break;
    case 'C':
        //
        //
        //
Common Programming Errors

printed for all nonnegative values of x.

```c
if (0 <= x <= 4)
    printf("Condition is true\n");
```

For example, let's consider the case when x is 5. The value of 0 <= 5 is 1, and 1 is certainly less than or equal to 4! In order to check if x is in the range 0 to 4, we could write a nested if statement such as

```c
if (0 <= x)
    if (x <= 4)
        printf("x is in the range 0-4\n");
```

```c
if (x = 10)
    printf("x is 10");
```

```c
if (x > 0)
    sum = sum + x;
    printf("Greater than zero\n");
else
    printf("Less than zero\n");
```
/* incorrect perception of noise loudness */

if (noise_db <= 110)
    printf("%d-decibel noise is very annoying.\n", noise_db);
else if (noise_db <= 90)
    printf("%d-decibel noise is annoying.\n", noise_db);
else if (noise_db <= 70)
    printf("%d-decibel noise is intrusive.\n", noise_db);
else if (noise_db <= 50)
    printf("%d-decibel noise is quiet.\n", noise_db);
else
    printf("%d-decibel noise is uncomfortable.\n", noise_db);
if ( noise_db <= 50 )
    printf ( "%d - decibel noise is quiet\n", noise_db );
else if ( noise_db <= 70 )
    printf ( "%d - decibel noise is intrusive\n", noise_db );
else if ( noise_db <= 90 )
    printf ( "%d - decibel noise is annoying\n", noise_db );
else if ( noise_db <= 110 )
    printf ( "%d - decibel noise is very annoying\n", noise_db );
else
    printf ( "%d - decibel noise is very uncomfortable\n", noise_db );