Reading Assignment #5

Topic: C

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

Chapter (5)

Repetition and Loop Statements
for loops, conditional loops, while loops, do-while loops
Flowchart of a while Loop

- Count emp < 7
  - True
    - Get data
    - Compute pay
    - Display pay
    - Increase count_emp by 1
  - False
    - Exit loop
while Statement

SYNTAX: while (loop repetition condition)
         statement

EXAMPLE: /* Display N asterisks. */
          count_star = 0;
          while (count_star < N) {
              printf("*");
              count_star = count_star + 1;
          }

INTERPRETATION: The loop repetition condition (a condition to control the loop process) is tested; if it is true, the statement is executed, and the loop repetition condition is retested. The statement is repeated as long as (while) the loop repetition condition is true. When this condition is tested and found to be false, the while loop is exited, and the next program statement after the while statement is executed.
/* Multiply data while product remains less than 10000 */
product = 1;
while (product < 10000) {
    printf("%d\n", product); /* Display product so far */
    printf("Enter next item> ");
    scanf("%d", &item);
    product = product * item; /* Update product */
}
Ex: Write a program to compute $x^n$ where $n$ is a positive integer, and $x$ and $n$ are provided by the user.

$x^n = 1 \cdot x \cdot x \cdot x \cdot x \cdot x \cdots x$

-------------------------

$n$ times

```c
int main (void)
{
    int i, n ;
    double x, power ;
    printf(" Enter x and n > ");
    scanf("%lf %d", &x, &n );
    i = 0 ;
    power = 1 ;
    while(i < n ) {
        power = power * x ;
        i ++ ;
    }
    printf("Power = %f", power );
}
```
/* Writing $x^n$ with a while loop */

void main(void)
{
    int i,n ;
    double x, power;
    printf("Enter x and n >");
    scanf("%lf %d", &x, &n);
    power =1;
    while (n > 0) {
        power = power * n ;
        n = n – 1 ; /* or n - - or n -= 1 */
    } /* while */
    printf ("Power = %lf", power);
} /* main */
/* Converting from celsius to farenheit using a sentinel loop */

#define SENTINEL -1000

int main ( void ) {
    double fahr, celsius;
    printf("Enter celsius temp or –1000 to quit >" ) ;
    scanf("%lf",&celsius);
    while ( celsius != SENTINEL ) {
        fahr = 1.8 * celsius + 32.0 ;
        printf("Farenheit equivalent = %f\n", fahr );
        printf("Enter celsius temp or –1000 to quit >" ) ;
        scanf("%lf",&celsius);
    } /* while */
} /* main */
/* Converting from farenheit temperature to celsius temperature */

#define SENTINEL -1000

int main ( void ) {
    double fahr, celsius ;
    printf("Enter farenheit temp or -1000 to quit >" ) ;
    scanf("%lf",&fahr);
    while ( fahr != SENTINEL ) {
        celsius = ( 5.0/9.0) * ( fahr - 32.0 ) ;
        printf("Celsius equivalent = %f\n", celsius );
        printf("Enter farenheit temp or -1000 to quit >" );
        scanf("%lf",&fahr);
    } /* while */
} /* main */
for Statement

SYNTAX: for (initialization expression;
    loop repetition condition;
    update expression)
          statement

EXAMPLE: /* Display N asterisks. */
    for (count_star = 0;
        count_star < N;
        count_star = count_star + 1)
          printf("*");

INTERPRETATION: First, the initialization expression is executed. Then, the loop repetition condition is tested. If it is true, the statement is executed, and the update expression is evaluated. Then the loop repetition condition is retested. The statement is repeated as long as the loop repetition condition is true. When this condition is tested and found to be false, the for loop is exited, and the next program statement after the for statement is executed.

Caution: Although C permits the use of fractional values for counting loop control variables of type double, we strongly discourage this practice. Counting loops with type double control variables will not always execute the same number of times on different computers.
Ex: Write a program to compute \( x^n \) where \( n \) is a positive integer, and \( x \) and \( n \) are provided by the user.

\[
x^n = \quad 1 \cdot x \cdot x \cdot x \cdot x \cdot x \ldots \times
\]

-------------------------

\( n \) times

```c
int main (void)
{
    int i, n;
    double x, power;
    printf(" Enter x and n > ");
    scanf("%lf %d", &x, &n);
    power = 1;
    for ( i = 0; i < n; ++ i )
        power = power * x;
    printf("Power = %f", power);
}
```
Printing a Celsius-to-Fahrenheit Conversion Table

/* Conversion of Celsius to Fahrenheit temperatures */

#include <stdio.h>

/* Constant macros */
#define CBEGIN 10
#define CLIMIT -5
#define CSSTEP 5

int
main(void)
{
    /* Variable declarations */
    int    celsius;
    double fahrenheit;

    /* Print the table heading */
    printf("  Celsius   Fahrenheit\n");

    /* Print the table */
    for (celsius = CBEGIN;
         celsius >= CLIMIT;
         celsius = celsius - CSSTEP) {
        fahrenheit = 1.8 * celsius + 32.0;
        printf("    %3d     %9.2f\n", celsius, fahrenheit);
    }

    return 0;
}

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>50.00</td>
</tr>
<tr>
<td>5</td>
<td>41.00</td>
</tr>
<tr>
<td>0</td>
<td>32.00</td>
</tr>
<tr>
<td>-5</td>
<td>23.00</td>
</tr>
</tbody>
</table>
Sentinel Loop

1. Initialize sum to zero.
2. Get first score.
3. while score is not the sentinel
   4. Add score to sum.
   5. Get next score.

One is tempted to try the following algorithm that reverses the order of Steps and 5 so as to be able to omit the duplication of Step 5 in Step 2.
sum = 0;

printf("Enter first score (or %d to quit) > ",
    SENTINEL);
for (scanf("%d", &score);
    score != SENTINEL;
    scanf("%d", &score)) {
    sum = sum + score;
    printf("Enter next score (%d to quit) > ",
        SENTINEL);
}

printf("\nSum of exam scores is %d\n", sum);

return (0);
```c
#include <stdio.h>

#define SENTINEL 0
#define NUM_MONTHS 12

int main(void) {
    int month, /* number of month being processed */
        mem_sight, /* one member's sightings for this month */
        sightings; /* total sightings so far for this month */

    printf("BALD EAGLE SIGHTINGS\n");
    for (month = 1;
         month <= NUM_MONTHS;
         month = month + 1) {
        sightings = 0;

        for (scanf("%d", &mem_sight);
             mem_sight != SENTINEL;
             scanf("%d", &mem_sight)) {
            sightings = sightings + mem_sight;
        }

        printf(" month %2d: %2d\n", month, sightings);
    }

    return (0);
}
```

Input data:

```
2 1 4 3 0
1 2 0
0
$ 4 1 0
...
```

Results:
```
BALD EAGLE SIGHTINGS
    month 1: 10
    month 2: 3
    month 3: 0
    month 4: 10
```
do-while Statement

SYNTAX:  do  
        statement  
        while (loop repetition condition);  

EXAMPLE: /* Find first capital letter input */  
          do  
            status = scanf("%c", &alpha);  
            while (status >= 0 && (alpha < 'A' || alpha > 'Z'));  

INTERPRETATION: First, the statement is executed. Then, the loop repetition condition is tested and if it is true, the statement is re-executed and the condition retested. The statement is executed repeatedly as long as the loop repetition condition is true. When this condition is tested and found to be false, the loop is exited and the next statement after the do-while is executed.

NOTE: If the loop body contains more than one statement, the group of statements must be surrounded by braces.
The loop

do {
    printf(“Enter a letter from A through E >”);
    scanf(“%c”, &letter_choice);
} while (letter_choice < ‘A’ || letter_choice > ‘E’);

prompts the user to enter one of the letters A through E. After scanf gets a data character, the loop repetition condition tests to see whether letter_choice contains one of the letters requested. If so, the repetition condition is false, and the next statement after the loop executes. If letter_choice contains some other letter, the condition is true and the loop body is repeated.

char skip_ch ;

/* Skip rest of data line */

do
    scanf(“%c”, &skip_ch);
while (skip_ch != ‘\n’);

Recall :

1997 \n
Bob \n
Common Programming Errors

while (x > xbig)
    x = x - 2;
    xbig = xbig + 1;
/* end while */

really executes as

while (x > xbig)
    x = x - 2; /* only this statement is repeated */

xbig = xbig + 1;
printf("Experiment successful?\(Y/N\) > ");
scanf("%c", &ans);

if (ans == 'Y') {
    printf("Enter data one value per line (\$d to quit)\n\> ", SENT);
    for (scanf("%d", &data);
        data != SENT;
        scanf("%d", &data)) {
        sum = sum + data;
        printf("> ");
        /* <= missing */
    }
} else {
    printf("Try it again tomorrow.\n\> ");
    printf("Now follow correct shutdown procedure.\n\> ");
}
Be very careful when using tests for inequality to control repetition of a loop. The following loop is intended to process all transactions for a bank account while the balance is positive.

```c
for (scanf (“% d % lf”, &code, &amount) ;
    balance != 0.0 ;
    scanf (“% d % lf”, &code, &amount) { 
    . . .
}
```

If the bank balance goes from a positive to a negative amount without ever being exactly 0.0, the loop will not terminate as planned. The loop that follows would be much safer:

```c
for (scanf (“% d % lf”, &code, &amount) ;
    balance > 0.0 ;
    scanf (“% d % lf”, &code, &amount) { 
    . . .
}
```