Goals

Understand how to use format statements to create tabular output. Understand how to execute a basic loop.

Review

Arithmetic:

- Reading integer values from the keyboard:

  ```c
  int x; // receives input value
  int howmany; // howmany values were read successfully
  howmany = fscanf(stdin "%d", &x);
  x = x + 5;
  ```

- Printing integer values to the screen:

  ```c
  fprintf(stdout, "The value of x is: %d\n", x);
  ```

- Printing \( x \) first in decimal and then in hexadecimal on the same line with specified field widths: (To have the output of multiple fprintfs() on the same line, omit the \n character from all but the final fprintf().)

  ```c
  fprintf(stdout, "X in decimal: %8d", x);
  fprintf(stdout, "X in hexadecimal: %6x\n", x);
  ```
Iterative execution

While it is often necessary to repetitively execute a statement or block of statements while or until some condition remains or becomes true or false.

Because of all of the possible formulations of while/until, remains/becomes, and true/false, it is always the case that there are always multiple correct ways to construct such iterations. And, alas, there are even more incorrect ways to construct them.

For this lab we will start with while some condition remains true. This structure is commonly called the "while loop."

Structure:

```c
while (condition)
{
    one-or-more statements;
}
```

or

```c
while (condition)
    exactly-one-statement;
```

Important notes

- the statement or statements controlled by the while() are called the body of the loop.
- It is necessary that the body of the loop modify the value of the condition. Why?
- Proper use of indentation is critical for human readability
- But indentation is completely irrelevant to the C compiler.
• Example: Print the integer values between 10 and 1 in decreasing order. (Note: This is not a complete C program. It needs to include stdio.h and it needs a proper main() function.)

```c
int value;
value = 10;

fprintf(stdout, " Table of values \n");

while (value > 0)
{
    fprintf (stdout, "%3d\n", value);
    value = value - 1;
}
```

• Note we only want the heading to be printed one time. So that fprintf() must be placed outside the loop.

Thought experiment: What does the following code segment do:

- Refuse to compile because of missing {} ?
- Compile fine and work fine?
- Compile fine but work not so fine?; if so how would it work not so fine??

```c
int value;
value = 10;

fprintf(stdout, " Table of values \n");

while (value > 0)
{
    fprintf (stdout, "%3d\n", value);
    value = value - 1;
}
```

**Iterative execution: For**

Another loop structure is commonly called the “for loop."

```c
for (init-expression; continue-condition; update-expression)
{
    loop-body
}
```

The init-expression is executed one time
The continue-condition is evaluated each iteration of the loop before the loop-body is executed. The update-expression is executed after the loop-body is executed. The loop-body is executed if and only if the continue condition is true.

**Example:**
The earlier while loop is equivalent to:

```c
for (value=10 ; value>0; value=value-1)  
{
   fprintf(stdout, "%3d\n", value);
}
```

As before, the statement or statements controlled by the for () are called the body of the loop.

Assignment:

Write a program called lab3.c that does the following:

- Read an integer value \( x \) from the standard input.
- Using a `while` loop compute \( x^n \) for the \( n = 1, 2, \ldots, 10 \) and print a table of the following format containing the values \( n \) and \( x^n \) in both decimal and hexadecimal representation. Do not use any C language facilities or C library functions (e.g. `pow()`) that were not discussed in this or previous lab documents!
- You will need three int variables in this program. I would call them \( n \), \( x \), and \( x^n \) (\( x^n \)).
- Which ones need to change inside the loop? In what way do they change?
- Your output should look EXACTLY like this table if (and only if) the input value is 2.

### Powers of 2 \(<----\) print value of \( x \) (not the constant 2)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x )</th>
<th>( x^n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>200</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>400</td>
</tr>
</tbody>
</table>

- To ensure that your output remains properly aligned, your first column should be 3 digits wide, the second 12, and the third 8. You should experimentally position the heading as shown.
- Modify your program, saving it as `lab3a.c` so that it uses a `for` loop instead of a `while` loop.
- Experimentally determine what is the largest input value that produces mathematically correct output. Add a comment to your program warning potential users that it will not work for values larger than the one you identify.

**Turn In Work**

Show your TA that you completed the assignment. Then submit your `lab3.c` program using the handin page: http://handin.cs.clemson.edu