CSE 142
Computer Programming I
Input and Output (I/O)

Overview
Topics
Output: printf
Input: scanf
Basic format codes
More on initializing variables

Writing Useful Programs
It's hard to write useful programs using only variables and assignment statements
Even our Fahrenheit to Celsius program needed more:
   - Needed a way to get data into and out of the program
   - We'll learn more about doing this today
     - Lots of terminology and messy details, but worthwhile.

What's a Computer?

Basic Definitions
Input: movement of data into memory from outside world (e.g., from keyboard).
   - Changes the value of a variable
   - "read" operation

Output: movement of data from memory to outside world (e.g., to monitor)
   - "write" operation
   - Does not change value of memory

Text Output
I/O Statements from a Familiar Program

```c
printf("Enter a Fahrenheit temperature: ");
scanf("%lf", &fahrenheit);
celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
printf("That equals %f degrees Celsius.\n", celsius);
```

Display Input and Output

The functions `printf` and `scanf` provide basic display I/O services.

- `printf("control string", list of expressions)`;
- `scanf("control string", list of &variables)`;

Control string gives the format of output or input.
Expressions are what to output.
Variables are where to store the input.
*`&` is magic (that is REQUIRED for `scanf`)*

Display Input and Output

```c
printf("control string", list of expressions) ;
scanf("control string", list of &variables) ;
```

Printf( ): Display Output

```c
int numPushups;
numPushups = 5 ;
printf("Hello. Do %d pushups. \n", numPushups);
```

Output: Hello. Do 5 pushups.

What Does the ‘\n’ Do?

```c
int numPushups;
numPushups = 5 ;
printf("Hello.  Do %d pushups.  \n", numPushups);
```

Output: Hello.  Do 5 pushups.

Getting a Little Fancier

```c
printf("control string", list of expressions) ;
```

Printf might have more than one expression in its list:

```c
printf("%d times %f is %f. \n", multiplier , pi , (double) multiplier * pi);
```

Output: 2 times 3.14000 is 6.28000.
Advanced Output Formatting
This is only the beginning! A few of many other things you can do:
- Control number of decimals
  3.1 vs 3.100000
- Exponential (scientific) or decimal notation
  3.1 vs 3.1E0
- Control total width (including spaces)
  _____3.1 vs __3.1

How?
Look in textbook or a reference manual, or online help!

Output Format Examples

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>%10.2f</td>
<td>_ _ _ _ 1 2 3 . 5 5</td>
<td>double</td>
</tr>
<tr>
<td>%10.4f</td>
<td>_ _ 1 2 3 . 5 5 0 0</td>
<td>double</td>
</tr>
<tr>
<td>%2f</td>
<td>1 2 3 . 5 5</td>
<td></td>
</tr>
<tr>
<td>%10d</td>
<td>_ _ _ _ _ _ _ 4 7 5</td>
<td>int</td>
</tr>
<tr>
<td>%-10d</td>
<td>4 7 5 _ _ _ _ _ _ _</td>
<td>int</td>
</tr>
<tr>
<td>%10c</td>
<td>_ _ _ _ _ _ _ _ _ a</td>
<td>char</td>
</tr>
</tbody>
</table>

scanf(): Read Input

```c
scanf ( "control string", &input list ) ;
int numPushups ;
printf ( "Hello.  Do how many pushups? " ) ;
scanf ( " %d " ,  &numPushups) ;
printf ( "Do %d pushups.\n",  numPushups) ;
output: Hello.  Do how many pushups?  5
Do 5 pushups.
```

If You Forget the ‘&’

The program will compile, but when you execute...

Whitespace

- `space ( )`, `tab (\t)`, `newline (\n)` are “whitespace”

Whitespace is skipped by scanf for int ("%d"), and double ("%lf")

This means the user can type spaces before a number and they are ignored

Not skipped for char input "%c"

each character typed, including spaces, is used

Multiple Inputs

Basic rule:

- % placeholders in the format must match variables in the input list
- MUST! match one-for-one in number, order, and type.
- int    studentID ;
- double grade ;
- scanf (" %d %lf", &studentID , &grade ) ;

```c
scanf (" %d %lf", &studentID, &grade) ;
```
Format Items Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>scanf()</th>
<th>printf()</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>%c %c</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>%d %d %i</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>%lf %f (long) float</td>
<td></td>
</tr>
</tbody>
</table>

What happens if types don’t match?
- printf -- garbled output
- scanf -- unpredictable errors

printf/scanf Summary

Output: printf("control string", output list);
- output list – expressions; values to be printed
- control string – types and desired format
  for now, NO "&", ever!

Input: scanf("control string", &input list);
- input list – variables; values to be read
- control string – types and expected format
  can be a way of initializing variables
  for now, YES "&", always!

Both: %x’s, I/O list match in number, order, type

I/O Summary

Input is the movement of data into memory
- In C, we use scanf for input from the keyboard
Output is the movement of data from memory
- In C, use printf for output to the screen

Know the basic printf/scanf rules, and know them well

Be aware that advanced formatting options exist and can be looked up when needed

Bonus Topic: More on Initializing Variables

Review: Initialization means giving something a value for the first time.

Potential ways to initialize:
- Assignment statement
  scanf
- Yet another way: initializer with declaration

Initializing when Declaring

Declarations without initializers
- int product, i;
- product = 40;
- i = 5;

Initializers are part of the declaration; they are not assignment statements (despite the = sign).

Declarations with initializers
- int product = 40, i = 5;

Initialization Quiz

int main (void) {
    /* line 1 */
    int a, b, c, d=10; /* line 2 */
    b=5; /* line 3 */
    d=6; /* line 4 */
    scanf("%d %d", &b, &c); /* line 5 */
    return 0; /* line 6 */
}

Q: Where is each of a, b, c, and d initialized?
Next Time

We’ll learn about a powerful new type of statement, the conditional or “if” statement

Please join me then!