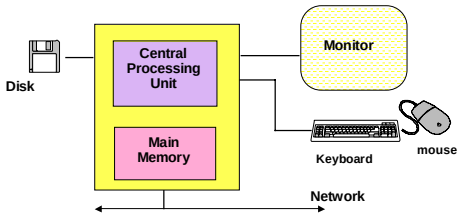
CSE / ENGR 142 Programming I

Display Input and Output (I/O)

Basic definitions

- **Input**: movement of data **into memory** from outside world (e.g., from keyboard).
  - Like assignment, 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

What's a Computer?



Examples of I/O Statements

```
printf("Enter a Fahrenheit temperature: ");

scanf("%lf",  &fahrenheit);

celsius =  (fahrenheit - 32.0) * 5.0 / 9.0;

printf("That equals %f degrees Celsius.",
                                  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.

printf( ): display output

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

output: Hello. Do 5 pushups.

%d is a placeholder ("conversion character") for an *int* value.

\n is an escape sequence for "newline" character.

Multiple Outputs

Basic rule:

% placeholders in format string match expressions in output list in number, order, and type.

```
int i;
double pi;
pi = 3.14 ;
i = 2;
printf("%d times %f is %f. \n", i , pi , (double) i * pi ) ;
```

Output: 2 times 3.14000 is 6.28000.

Formatting Output

- **A few of many 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**

Look in textbook or a reference manual!

Output Format Examples

%10.2f	_ _ _ _ 1 2 3 . 5 5	double
%10.4f	_ _ 1 2 3 . 5 5 0 0	
%.2f	1 2 3 . 5 5	
%10d	_ _ _ _ _ _ _ 4 7 5	int
%-10d	4 7 5 _ _ _ _ _ _ _	
%10c	_ _ _ _ _ _ _ _ _ a	char

scanf( ): read input

scanf ("control string", &input list) ;

```
int number ;
printf ( "Hello.  Do how many pushups? " ) ;
scanf ( " %d " ,  &number ) ;
printf ( "Do  %d  pushups. \n",  number ) ;
```

output:

```
Hello.  Do how many pushups?  5
Do  5  pushups.
```

input list variables *MUST* be preceded by an &.
input list variables *MUST* be preceded by an &.

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 student_id ;
double grade ;
scanf ("%d %lf", &student_id , &grade ) ;
```


Whitespace

- **"Whitespace": space ' ', tab '\t', newline '\n'**
- **Skipped by scanf for int "%d", and double "%lf"**
  - **user can type spaces before a number and they are ignored**
- **Not skipped for char input "%c"**
  - **each character typed, including spaces, is used**
- **It can get confusing, especially when char and numeric variables are intermixed**
  - **We'll avoid mixing them for now.**

Format Items Summary

Type	scanf()	printf()	
char	%c	%c	
int	%d	%d	%i also works
double	%lf	%f	← (long) float

What happens if types don't match?

printf -- garbled output

scanf -- unpredictable errors and don't forget the & !

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

```
int product, i;
  /*declarations without
initializers */

product = 40;
i = 5;
  /*initialization via
assignment statements
*/
```

```
int product = 40, i =5;
  /*declaration  with
initializers, */

i = 6;
        /*not an
          inititialization! */
```

Initializers are part of the declaration; not considered assignment statements.

Initialization Quiz

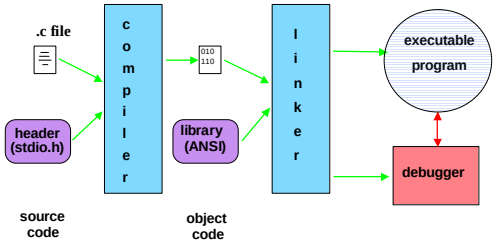
```
int main (void){        /*line 1*/
   int a, b, c, d=10;   /*line 2*/
   b=5;                 /*line 3*/
   d=6;                 /*line 4*/
   scanf("%d %d", &b, &c);      /*line 5*/
   }
```

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

I/O 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

Compilers, Linkers, etc.



Syntax vs Semantics/Logic

- **Syntax**: the required form of the program
  - punctuation, keywords, word order, etc.
  - The C compiler always catches these "syntax errors" or "compiler errors"
- **Semantics and logic**: what the program means
  - what you want it to do
  - The C compiler cannot catch these kinds of errors!
  - They can be extremely difficult to find
  - Logic errors may not show up right away

Syntax or logic errors?

```
#include <stdio.h>
int main (void) {
double far, cel;

far = 56.0;
cel = (far-32.0)*5.0/9.0

printf ('Celsius is %f ', cell);

retrun (0);
}
```

```
#include <stdio.h>
int main (void) {
double far, cel;

far = 56.0;
cel = far-32.0*5.0/9.0;

printf ("Celcius is %d", far);

return (0);
}
```


Syntax or logic errors?

```
#include <stdio.h>
int main (void) {
double far, cel;

far = 56.0;
cel = (far-32.0)*5.0/9.0;

printf ('Celsius is %f ', cell);

retrun (0);
}
```

```
#include <stdio.h>
int main (void) {
double far, cel;

far = 56.0;
cel = far-32.0*5.0/9.0;

printf ("Celcius is %d", far);

return (0);
}
```