CSE 142
Computer Programming I

Input and Output (I/O)

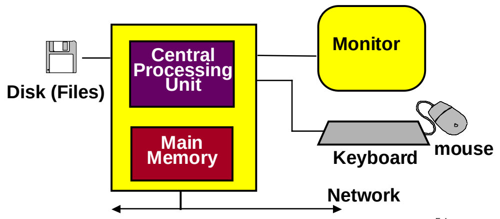
Overview

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?

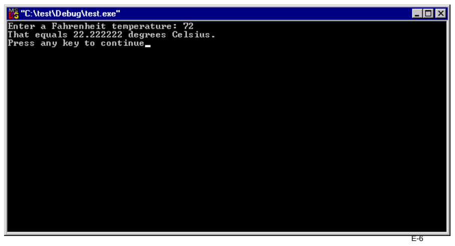


Basic Definitions

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



```
Enter a Fahrenheit temperature: 72
That equals 22.222222 degrees Celsius.
Press any key to continue_
```

I/O Statements from a Familiar Program

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

scanf("%lf",  &fahrenheit);

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

printf("That equals %f degrees Celsius.",
                                   celsius);
```

E-7

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!)

E-8

printf( ): Display Output

```
int      numPushups;

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

```
output:  Hello.  Do 5 pushups.
```

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

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

What Does the '\n' Do?

```
int      numPushups;

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

```
output:  Hello.  Do 5 pushups.
         Do them now.
```

E-10

Getting a Little Fancier

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

printf might have more than one expression in its list:

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

E-11

Multiple Output Expressions

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

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

```
Output:  2 times 3.14000 is 6.28000.
```

E-12

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

%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 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.
```

input list variables MUST be preceded by an &.

input list variables MUST be preceded by an &.

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 and newlines, are read separately

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 ) ;
```

Input Errors

What happens if the user doesn't type the right thing for scanf?

- Number with a decimal point when integer expected…
- Character when number expected…

Answer: scanf halts - doesn't change corresponding variables

Can we detect this when it happens?

E-19

scanf function result

Besides storing input values in variables, scanf also returns a result that is the number of input values successfully read

That result can be used to detect input errors and react (once we know a bit more about C)

```
                  # items actually read
nValuesRead = scanf("%d %d", &x, &y);
if (nValuesRead != 2) {
      /* do something appropriate */
      …
}                           input stored in x, y
```

E-20

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 & !

E-21

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!

E-22

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

E-23

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

E-24

Initializing when Declaring

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

product = 40;
i = 5;
	i = 6;

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

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