CSE 142
Computer Programming I

Input and Output (I/O)

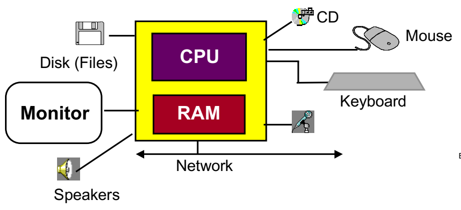
Overview

Topics

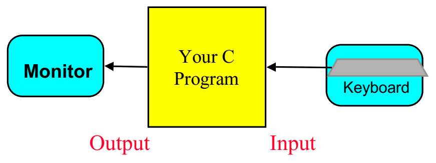
- What is "I/O"?
- What is "Stream I/O"?
- What Functions Do Programmers Require?
- Output in C: printf
- Input in C: scanf
- Basic format codes

What Is I/O? (Input/Output)

I/O refers to any movement of information between the CPU (central processing unit) and other devices



The I/O We're Considering



An Example C Program

Convert temperature in Fahrenheit to Celsius.

What Do We Notice?

- *printf* is used to produce output

```
printf ("Please input a Fahrenheit temperature: ");
```

- *scanf* is used to obtain input

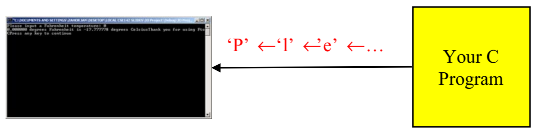
```
numItemsRead = scanf ("%lf", &degreesInFahrenheit);
```

(I'll explain both of these in detail in a moment.)

- The screen has lines and columns, but we don't seem to mention them in our *printf* statement
- Why? How does C know where on the screen to put what we want to print?

Stream Output

printf ("Please input a Fahrenheit temperature: ");



E-7

The Screen

printf ("Please input a Fahrenheit temperature: ");

P	l	e	a	s	e	●	i	n	p
u	t								

“Wraps”

The Output Window

E-8

What If You Want To Go To The Next Line?

‘\n’ (which is a single character called “newline”)

printf ("Please\ninput\na\nFahrenheit temperature: ");

P	l	e	a	s	e	\n			
i	n	p	u	t	\n				
a	\n								
F	a	h	r	e	n	h	e	i	t

E-9

Only the ‘\n’s Are Important, Not the Number of printf’s

```
printf (“this is a test\n”);
printf (“this is line two\n”);
```

This is the usual way to write this

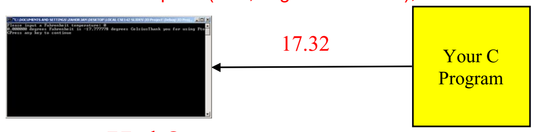
```
printf (“this is a test\nthis is line two\n”);
```

```
printf (“this is”);
printf (“ a test\nthis is”);
Printf (“ line two\n”)
```

E-10

Outputting Variable Values

```
degreesInCelsius = 17.32;
printf (“%f“, degreesInCelsius);
```

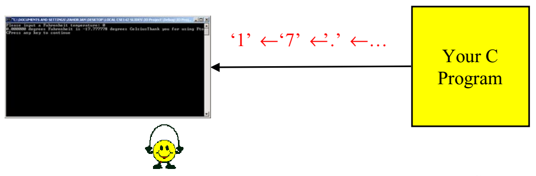


Huh?

E-11

Printf Converts Numeric Values to Characters for You

```
degreesInCelsius = 17.32;
printf (“%f“, degreesInCelsius);
```



E-12

Printf

printf(“%f degrees Celsius”, degreesInCelsius) ;

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

Control string says what to print.

“%f degrees Celsius” →
print a double followed by “ degrees Celsius”

%f is a placeholder (“conversion character”) for a *double* value.

Expressions are the values to output.

E-13

Conversion Characters

- %f means the expression is a double
  (Note: The expression better be a double!)
- %d means the expression is an integer
- %c means the expression is a character

There are others. Check the book if you’re interested.

E-14

Getting a Little Fancier

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

printf might have more than one expression in its list:

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

E-15

Multiple Output Expressions

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

```
double multiplier;
double pi;
pi = 3.14;
multiplier = 2.0;
printf(" %d times %f is %f. \n",
        multiplier , pi , (double) multiplier * pi );
```

Output: 2.00000 times 3.14000 is 6.28000.

E-16

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!

E-17

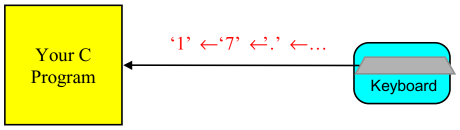
Output Format Examples

%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

E-18

Input

numItemsRead = scanf ("%lf", &degreesInFahrenheit);



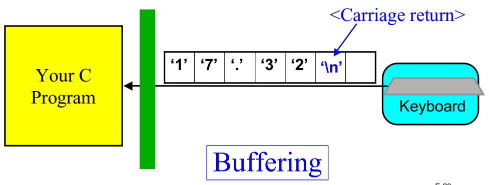
This scheme won't work. Why not?

- I want a double, not characters
- What if the user types a backspace?

E-19

What if the User Types a Backspace?

numItemsRead = scanf ("%lf", &degreesInFahrenheit);

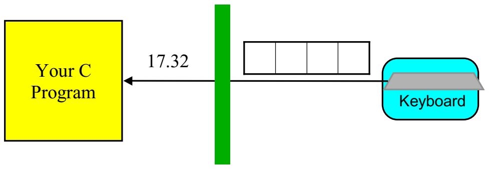


Buffering

E-20

Conversion

numItemsRead = scanf ("%lf", &degreesInFahrenheit);



E-21

scanf( )
"Just like printf" (sort of)

numItemsRead = scanf ("%lf", &degreesInFahrenheit);

numItemsRead = scanf ("control string", &variablename) ;

Control string:
- **"%lf" means "read a double"**
- **"%d" means "read an int"**
- **"%c" means "read a char"**

&variablename:
- **A variable name preceded by a '&'**
- **(It had better be a variable of the correct type!)**

E-22

If You Forget the '&'

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

E-23

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

E-24

Possible, But Bad Ideas

```
numItemsRead = scanf (" %d %lf", &studentID , &grade ) ;

numItemsRead = scanf("My grade is %lf", &grade);
```

E-25

Programs Would Work a Lot Better if Users Were Smarter

Please input a Fahrenheit temperature: just above freezing

E-26

Input Errors Happen

- Your prompts will never be clear to all 8 billion people on the planet
- A typing mistake on the 7th online order form web page shouldn't make you start over
- Data is never "clean"

E-27

What Should Happen

1. You notice the problem, print better information, ask for the input again, and give the user a "cancel" option
2. You notice the problem, print the same prompt, ask again (and maybe provide a cancel).
3. You quit.
4. You crash.
5. You pretend you didn't notice, and eventually print incorrect results as though they were correct.

E-28

How Can You Tell That Scanf Has Failed?

# items actually read

```
numItemsRead = scanf("%d %d", &x, &y);
if (numItemsRead != 2) {
     /* do something appropriate */
     …
}
```

input stored in x, y

E-29

What Should You Do?

For now:

```
numItemsRead = scanf("%lf", &degreesFahrenheit);
if (numItemsRead != 1) {
     printf ("My very informative error message\n");
     return –1;
}
```

E-30

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

E-31

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-32

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-33

Initializing when Declaring

Declaration without initializer	Declarations with initializers
int product;	int product = 40;
product = 40;	

For everything we’ll do for a while (and maybe ever), these are functionally identical.

The only difference is one of style.

E-34

Next Time

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

E-35