

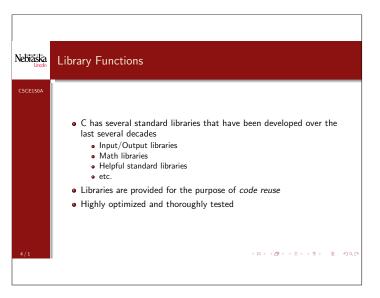
		_
		_
		_
		_
		-
		-
		-

Nebraska Lincoln	Chapter 3
CSCE150A	
	<ul> <li>3.1 Building Programs from Existing Information</li> <li>3.2 Library Functions</li> <li>3.4 Functions without Arguments</li> <li>3.5 Functions with Arguments</li> <li>3.6 Common Programming Errors</li> </ul>
2/1	(D) (B) (E) (E) E TQQ

Notes		
		•
		-
		-
		-
		-
		-

,	
Nebraska Lincoln	Existing Information
CSCE150A	<ul> <li>Programmers seldom start off writing completely original programs.</li> <li>Often the solution can be developed from information that already exists or from the solution to another problem.</li> <li>No point in "reinventing the wheel"</li> <li>Designing pseudocode generates important information before you even begin to code a program.         <ul> <li>A description of a problem's data requirements</li> <li>A description of a problem's solution algorithm</li> </ul> </li> <li>This provides a starting point in coding your program.         <ul> <li>What portions of this program can be taken care of by standard library functions?</li> <li>What portions of this code can be grouped into a stand-alone function?</li> </ul> </li> </ul>

Notes			



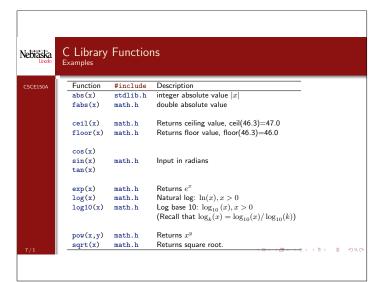
		_
		_
		_
		_
		_

Nebiaska Lincoln	Predefined Functions and Code Reuse
CSCE150A	<ul> <li>A primary goal of software engineering is to write error-free code.</li> </ul>
	<ul> <li>This is facilitated by code reuse, reusing program fragments that have already been written and tested</li> <li>C promotes reuse by providing many predefined functions that can be used to perform mathematical computations.</li> </ul>
5/1	+D+ (B+ (Z+ (Z+ (Z+ (Z+ (Z+ (Z+ (Z+ (Z+ (Z+ (Z

Not	-es			
1400	.03			

Nebraska Lincoln	Standard Math Library
CSCE150A	<ul> <li>Functions such as sqrt are found in the standard math library to perform the square root computation.</li> <li>The function call in the assignment statement y = sqrt(x); activates the code for function sqrt, passing x to the function.</li> <li>After execution, the result of the function is substituted for the function call.</li> <li>If x is 16.0, the assignment statement above is evaluated as follows: √16.0 is evaluated to 4.0, the call sqrt(x) is replaced with 4.0, and then y takes the value 4.0.</li> <li>To include, use: #include<math.h></math.h></li> <li>Note: Generally, when using the math library, you must use -lm: prompt:&gt;gcc -lm myMathProgram.c (though cse's gcc doesn't require it)</li> </ul>
6/1	(D) (B) (E) (E) (E) (D)

Notes			



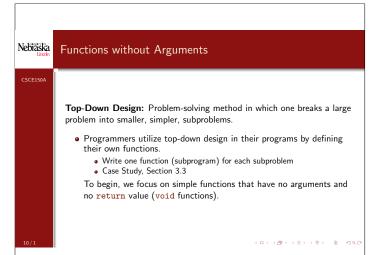


### Punction Specifics • abs(x) is the only function listed with an int value argument and result. • All others have both double as the argument and double as the result. • tan(x), cos(x) and sin(x) take as their input the radians • If a function is called with an argument that is not the argument's data type, the argument is converted to the required data type before it is used. • Conversion of type int to type double cause no problems, but a conversion of type double to type int leads to the loss of any fractional part. • The arguments for sqrt, log and log10 must be positive. • Invalid inputs may result in NaN, inf, -inf, etc.

No	tes		

Nebřäska Lincoln	Functions Without Arguments
CSCE150A	
	In C, functions have three important parts:
	<ul> <li>Function Prototypes - contains the name, return type and arguments of a function</li> </ul>
	• Function Definitions - the implementation of the function
	• Placement of Functions in a Program - how do we use functions?
9 / 1	(D) (B) (E) (E) E VQ(

	-
	-
	-
	-
	_



## Notes \_\_\_\_\_\_

## Function Prototypes As with other identifiers in C, a function must be declared before it can be referenced (used). One way to declare a function is to insert a function prototype before the main function. Standard library functions (printf, scanf, sqrt, etc.) are prototyped in header (.h) files A function prototype tells C compiler the data type of the function, the function name, and information (number, data type) about the arguments that the function expects. Data Type of the function is the type of value returned by the function. Functions that return no value are of type void

No	tes			

Nebřaska Lincoln	Function Definitions
CSCE150A	<ul> <li>The function prototype (declaration) does not specify the function operation (what it does).</li> <li>The variable declaration: int c; does not tell you how c will be used.</li> <li>To do this, you need to provide a definition for each function subprogram (similar to the definition of the main function).</li> <li>The function heading is similar to the function prototype, but not ended by the symbol ';'.</li> <li>The function body (enclosed in braces) contains the implementation of the function (specifies what it does)</li> <li>The return statement is optional for void functions</li> </ul>
12/1	(D) (B) (E) (E) E 900

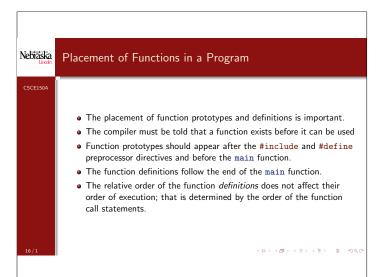
Notes			

		_
		-
		-
		-
		-
		-
		-

## Function Definition Scope • Each function body may contain declarations for its own variables. • These variables are considered local to the function • They can be referenced only within the function. • No other function has access to their values and they are destroyed after the return statement. • This is known as a variable's scope

Notes			

Nebraska		ction Definition  Example
CSCE150A	1	/* function prototype */
	2	<pre>void hereIsAFunction();</pre>
	3 4	int main(void)
	5	int main(void)
	6	int x;
	7	
	8	}
	9	
	10	/* function definition */
	11	void hereIsAFunction(void)
	12 13	{
	14	int y;
	15	 }
	** L	,
15/1	mai	n cannot see y and hereIsAFunction cannot see x

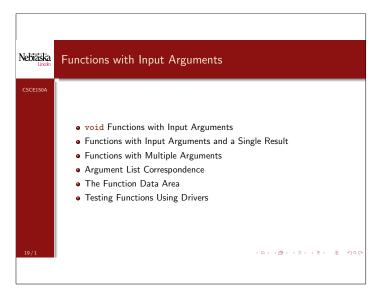


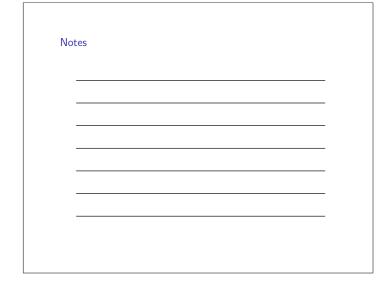
### Nebraska Full Example /\* Program Hello, World \*/ #include <stdio.h> /\*Function Prototypes \*/ void Hello\_World(void); 5 6 int main(void) { Hello\_World(); return 0; 8 10 11 12 /\* Function Definitions \*/ /\* Prints Hello, World \*/ void Hello\_World(void) { printf("Hello, World\n"); 13 14 15

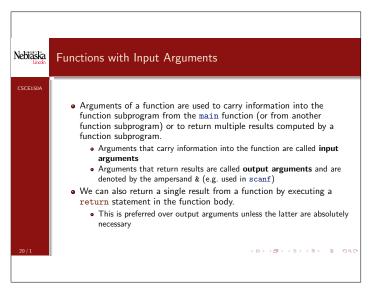
Not	es	

Nebiäska Lincoln	Displaying User Instructions
CSCE150A	<ul> <li>Simple functions have limited capability.</li> <li>Without the ability to pass information into or out of a function, we can use functions only to do local computation</li> <li>Example: display multiple lines of program output, instructions to a program user or a title page or a special message that precedes a program's result.</li> </ul>
18/1	(B) (B) (E) (E) E 0900

Not	es		



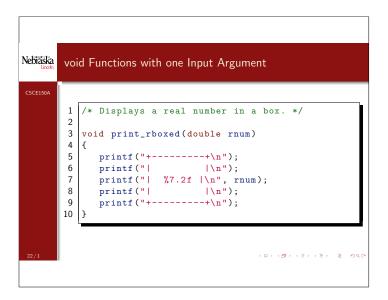


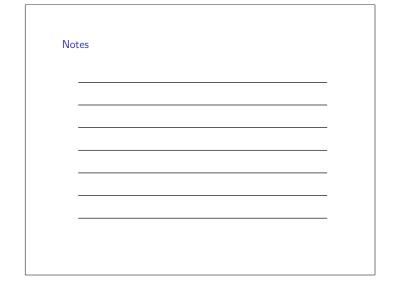


No	tes			

Nebraska Lincoln	void Functions with Input Arguments
CSCE150A	<ul> <li>Functions without arguments are too limited.</li> <li>We can use a void function with an argument to "dress up" our program output by having the function display its argument value in a more attractive way.</li> <li>(Recall that a void function does not return a result.)</li> </ul>
21/1	- 15 + 15 + 15 + 10 + 10 + 10 + 10 + 10 +

Notes		
		_
		_
		_
		_
		_
		_



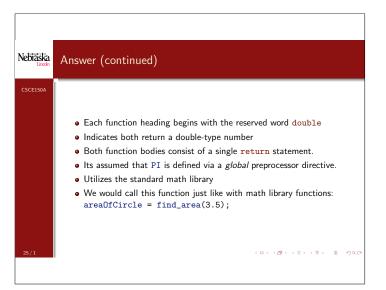


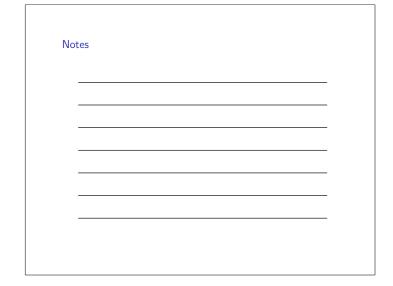
# Problem Design two functions to compute the area and circumference of a circle using one input argument to each (the radius). Functions with Input Argument and a Single Result o C functions can only ever return one value o sqrt(x), abs(x), pow(x,y), etc. return one value of type double o May return any built-in type or user-defined type Problem Design two functions to compute the area and circumference of a circle using one input argument to each (the radius).

		_

Nebraska	Answer
CSCE150A	<pre>double find_circum(double r) 2 { 3     return (2.0 * PI * r); 4 } 5  6 double find_area(double r) 7 { 8     return (PI * pow(r,2)); 9 }</pre>
24 / 1	(D) (B) (E) (E) E VQC

Notes		



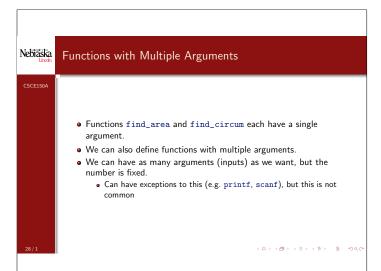


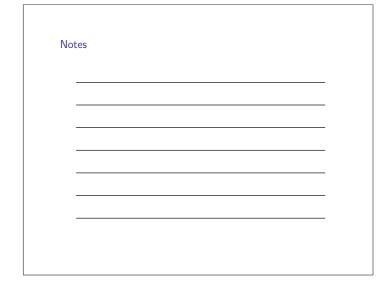
Nebraska Lincoln	Additional Considerations
CSCE150A	<ul> <li>What happens if we pass a negative value to find_area?</li> <li>Can we make it more efficient?</li> <li>Can we make it more readable?</li> </ul>
26/1	(□) (∰) (≷) (≷) (≷) € 9Q(P

Notes		

Nebřaška Lincoln	Be	tter Area Function
CSCE150A	1 2 3 4 5 6 7 8 9 10 11 12 13 14	<pre>* Return Value: area */ double find_area(double radius) {    double area;    if(radius &lt; 0)         area = 0.0;    else</pre>
27/1	14	( The state of the

_



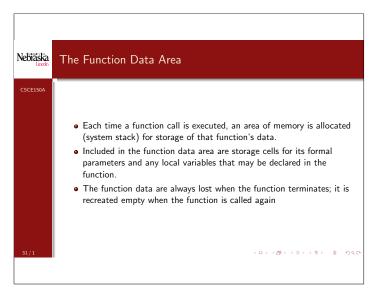


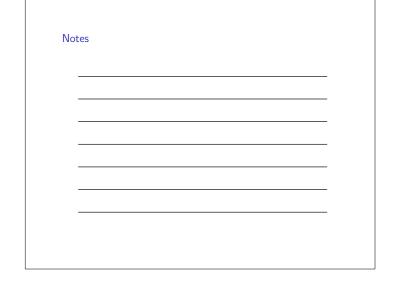
### Nebraska \* Multiplies its first argument by 10 raised $\ast$ its second power, i.e. 3 4 \* x \* 10^y, \* where x is the first argument and y 6 st is the second argument 7 8 double scale(double x, int y) 9 10 double scale\_factor; 11 scale\_factor = pow(10, y); 12 return (x \* scale\_factor); 13 4 m >

Notes			
Notes			

### Argument List Correspondence When using multiple-argument functions, be careful to include the correct number of arguments in the function call. The order or the actual arguments used in the function call must correspond to the order of the formal parameters listed in the function prototype. The type of each argument must match when calling the function: do not pass a double into a function where the formal parameter is data type int

Notes			

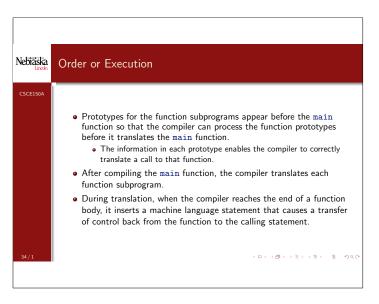


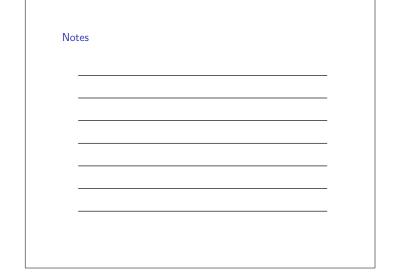


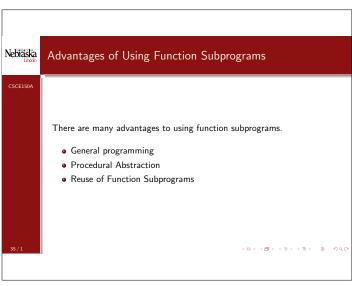
Nebřaska Lincoln	Testing Functions Using Drivers
CSCEISOA	<ul> <li>A function is an independent program module, meaning it can be tested separately from the program that uses it.</li> <li>To run such a test, you should write a short driver function.</li> <li>A driver function defines the function arguments, calls the functions, and displays the value returned.</li> </ul>
32/1	くロン(費)(き)(き) (き) (き)

No	tes			

Nebraska Lincoln	Wrap-Up
CSCE150A	<ul> <li>Program Style</li> <li>Order of Execution of Function Subprograms and Main Function</li> <li>Advantages of Using Function Subprograms</li> <li>Displaying User Instructions</li> </ul>
33/1	(B) (분) (분) (분) (원)







N.			
No	tes		
			_

Nebraska Lincoln	General Programming
CSCE150A	<ul> <li>Their availability changes the way in which an individual programmer organizes the solution to a programming problem</li> <li>For a team of programmers working together on a large problem, each member can focus on solving a set of subproblems.</li> <li>Simplify programming tasks by providing building blocks for new programs.</li> </ul>
36 / 1	(D) (B) (E) (E) E DQ(

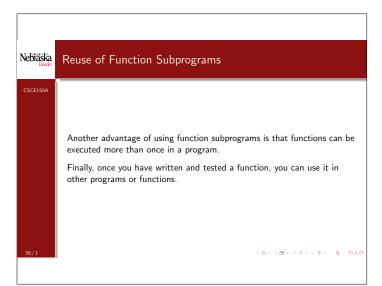
Notes			



### Procedural Abstraction

- Function subprograms allow us to remove from the main function the code that provides the detailed solution to a subproblem.
  - $\bullet$  Because these details are provided in the function subprograms and not in the main function, we can write the main function as a sequence of function call statements as soon as we have specified the  $% \left\{ 1,2,\ldots ,n\right\}$ initial algorithm and before we refine any of the steps.
  - $\bullet$  We should delay writing the function for an algorithm step until we have finished refining the previous step.
- With this approach to program design, called procedural abstraction, we defer implementation details until we are ready to write an individual function subprogram.
- Focusing on one function at a time is much easier than trying to write the complete program at once.
- E.g. program circle in textbook





Nebraska Lincoln
---------------------

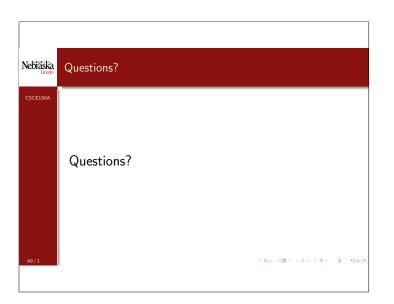
### Common Programming Errors

- Remember to use a **#include** preprocessor directive for every standard library from which you are using functions.
- Use appropriate options for the compiler to link in the libraries you need
- Place prototypes for your own function subprogram in the source file preceding the main function; place the actual function definitions after the main function.
- The acronym **not** summarizes the requirements for argument list correspondence:
  - Provide the required number of arguments,
  - Make sure the order of arguments is correct, and
  - $\bullet$  Each function argument is the correct  $t\mbox{\sc type}$  or that conversion to the correct type will lose no information.
- Also be careful in using functions that are undefined on some range of values. 4 m >

Not	tes	

Not	es			

Not	es	
-		_
-		_
-		-
-		-
-		-
-		-



		_
		-
		-
		-
		_
		_

Nebraska Lincoln	Exercise
CSCE150A	Problem  Design a program that takes prompts for inputs, a, b, c and uses two functions quadraticRootOne, quadraticRootTwo which return the real-valued roots of the quadratic equation,
	$ax^2+bx+c$ Hint: recall the quadratic equation:
	$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
	Be sure to format your output (you may assume that the input doesn't result in any complex roots, that is, $b^2 \geq 4ac$ ).
41/1	(B) (E) (E) E 99(

Not	res	