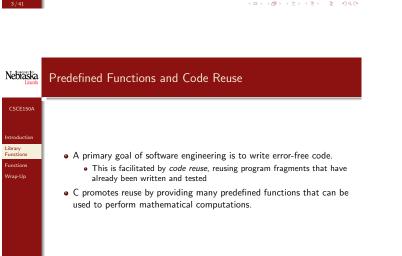
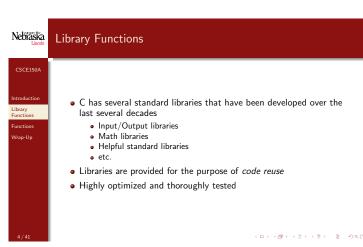
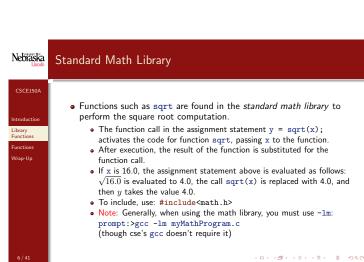


CSCE150A Introduction Library Functions Wrap-Up Programmers seldom start off writing completely original programs. Often the solution can be developed from information that already exists or from the solution to another problem. No point in "reinventing the wheel" Designing pseudocode generates important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

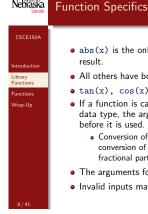


4 D > 4 D > 4 E > 4 E > E 9 Q C





C Library Functions Nebraska Description Function #include abs(x) stdlib.h integer absolute value |x|fabs(x) math.h double absolute value ceil(x) math.h Returns ceiling value, ceil(46.3)=47.0 math.h Returns floor value, floor (46.3)=46.0 floor(x) cos(x) sin(x) math.h Input in radians tan(x) exp(x) math.h Returns e^x log(x) math.h Natural log: $\ln(x), x > 0$ log10(x) math.h Log base 10: $\log_{10}(x), x > 0$ (Recall that $\log_k(x) = \log_{10}(x)/\log_{10}(k)$) pow(x,y) math.h Returns x^y sqrt(x) math.h Returns square root.



Nebraska

- 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.
- \bullet 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.

4 D > 4 B > 4 E > 4 E > E 9 Q C

Nebraska

Functions Without Arguments

CSCE150A

In C. functions have three important parts:

- Function Prototypes contains the name, return type and arguments of a function
- Function Definitions the implementation of the function
- Placement of Functions in a Program how do we use functions?



Functions without Arguments

CSCE150A

Top-Down Design: Problem-solving method in which one breaks a large problem into smaller, simpler, subproblems.

- Programmers utilize top-down design in their programs by defining their own functions.
 - Write one function (subprogram) for each subproblem
 - Case Study, Section 3.3

To begin, we focus on simple functions that have no arguments and no return value (void functions).



40 × 40 × 42 × 42 × 2 990





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



Function Definitions

- The function prototype (declaration) does not specify the function operation (what it does).
 - The variable declaration: int c; does not tell you how c will be used.
- To do this, you need to provide a definition for each function subprogram (similar to the definition of the main function).
- The function heading is similar to the function prototype, but not ended by the symbol ':'.
- \bullet The function ${\bf body}$ (enclosed in braces) contains the implementation of the function (specifies what it does)
- The return statement is optional for void functions

4 m > 4 m >

```
Function Prototypes & Definition
Nebraska
         1
               /* function prototype */
         3
               void printProgramInfo();
         5
               int main(void)
         6
                 return 0;
               3
         g
         10
         12
               /* function definition */
         13
               void printProgramInfo(void)
         15
                 printf("Program Example for CSCE 150A\n");
                  printf(" copyright(c) C. Bourke 2009\n");
         16
         17
                 return;
```

```
Function Definition
Scope

CSCE150A

Introduction
Library
Functions
Functions
No Argaments
With Argaments
Multiple Args
Wrap-Up

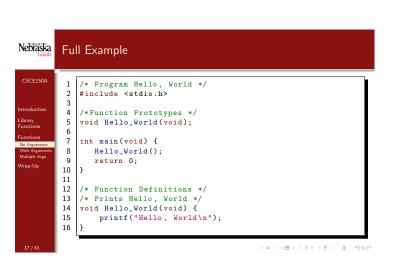
Wrap-Up

They can be referenced only within the function.

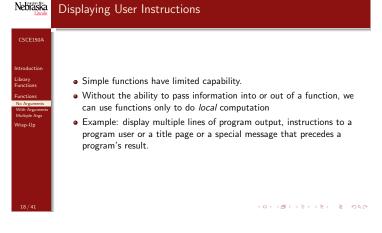
No other function has access to their values and they are destroyed after the return statement.

This is known as a variable's scope
```

Function Definition Nebraska /* function prototype */ void hereIsAFunction(): 3 int main(void) int x; } g /* function definition */ 10 void hereIsAFunction(void) 12 13 int y; 14 main cannot see y and hereIsAFunction cannot see x



Placement of Functions in a Program CSCE150A The placement of function prototypes and definitions is important. The compiler must be told that a function exists before it can be used Functions No Arguments With Arguments Which place Args Wyap-Up The function definitions follow the end of the main function. The relative order of the function definitions does not affect their order of execution; that is determined by the order of the function call statements.



Nebraska Functions with Input Arguments • void Functions with Input Arguments • Functions with Input Arguments and a Single Result • Functions with Multiple Arguments • Argument List Correspondence • The Function Data Area • Testing Functions Using Drivers 4 D > 4 D > 4 E > 4 E > E 9900

Functions with Input Arguments • Arguments of a function are used to carry information into the function subprogram from the main function (or from another function subprogram) or to return multiple results computed by a function subprogram. • Arguments that carry information into the function are called input arguments • Arguments that return results are called **output arguments** and are denoted by the ampersand & (e.g. used in scanf) • We can also return a single result from a function by executing a return statement in the function body. • This is preferred over output arguments unless the latter are absolutely

4 D > 4 B > 4 E > 4 E > 9 Q P

Nebraska

Nebraska

Answer

Nebraska void Functions with Input Arguments • Functions without arguments are too limited. • 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. • (Recall that a void function does not return a result.)

Nebraska void Functions with one Input Argument /* Displays a real number in a box. */2 3 void print_rboxed(double rnum) 4 printf("+----+\n"); 5 printf("| |\n"); 6 7 printf("| %7.2f |\n", rnum); printf("| |\n"); 8 printf("+----+\n"); 9 10 4 m > 4 d > 4 d > 4 d > 4 d > 9 q Q

```
Nebraska
          Functions with Input Argument and a Single Result
             • C functions can only ever return one value
             • sqrt(x), abs(x), pow(x,y), etc. return one value of type double
             • May return any built-in type or user-defined type
           Design two functions to compute the area and circumference of a circle
           using one input argument to each (the radius).
                                                        4D > 4B > 4B > 4B > B 990
```

```
double find_circum(double r)
2
      return (2.0 * PI * r);
5
   double find_area(double r)
      return (PI * pow(r,2));
                                          4 D > 4 B > 4 E > 4 E > E 9940
```

Answer (continued) CSCE150A Library • Each function heading begins with the reserved word double • Indicates both return a double-type number • Indicates both function bodies consist of a single return statement. • Its assumed that PI is defined via a global preprocessor directive. • Utilizes the standard math library • We would call this function just like with math library functions: areaOfCircle = find_area(3.5);

4 D > 4 B > 4 B > 4 B > 3 8 9 9 9

```
Additional Considerations

CSCE190A

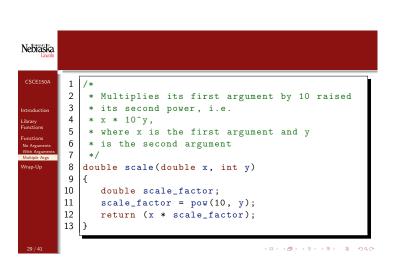
Introduction
Library
Functions
Punctions
Work Arguments
Work Arguments
Multiple Args
Wap-Up

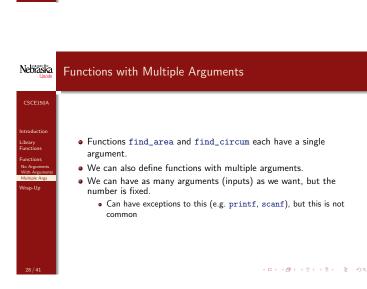
What happens if we pass a negative value to find_area?

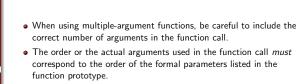
Can we make it more efficient?

Can we make it more readable?
```

Nebraska Better Area Function 2 Compute the area of a circle 3 * Input: double radius 4 * Return Value: area 5 */ 6 double find_area(double radius) 7 8 double area; 9 if(radius < 0) 10 area = 0.0; 11 area = 3.14159265 * radius * radius; 12 13 return area; 14







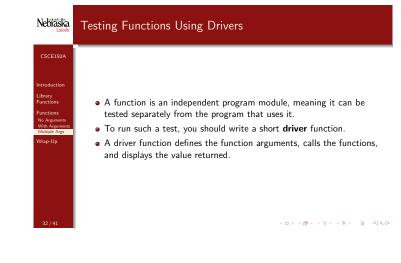
Argument List Correspondence

Nebraska

 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

4 B > 4 B > 4 B > 4 B > 9 Q C

The Function Data Area CCCCE150A Introduction Library Functions Functions New Argaments With Argaments With Argaments With Argaments With Paperunts With P



 Prototypes for the function subprograms appear before the main function so that the compiler can process the function prototypes

• After compiling the main function, the compiler translates each

 During translation, when the compiler reaches the end of a function body, it inserts a machine language statement that causes a transfer of control back from the function to the calling statement.

• The information in each prototype enables the compiler to correctly

before it translates the main function.

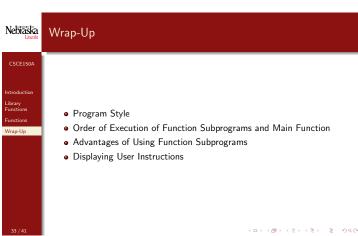
translate a call to that function.

function subprogram.

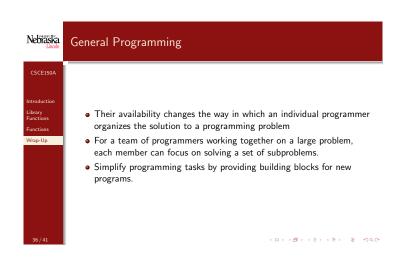
Nebraska

CSCE150A

Order or Execution







Nebraska Procedural Abstraction • Function subprograms allow us to remove from the main function the code that provides the detailed solution to a subproblem. • 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 initial algorithm and before we refine any of the steps. • 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

4 D > 4 B > 4 E > 4 E > E 994 P

