

Function parameters and return values

- *A function* is a set of instructions
 - When *executed*, it accomplishes a task
- Most functions require *input parameters*
 - Pieces of data that the function needs to do its job
 - This is **not the same** as stream input from the user
- Many functions *output a return value*
 - A piece of data that is the result of that job
 - This is **not the same** as stream input to the screen

Functions vs. operators

- Both instruct the computer to perform an operation
- Operator syntax

```
operand1 operator operand2
```

- Operator is a special symbol
- All operators are binary (two operands)

```
5 + 2
```

```
cout << "Hello"
```

- Evaluates to (returns) a single value

- Function syntax

```
function( parameter2, parameter2 ... )
```

- Function name is an identifier
- Any number of parameters allowed (including none)

```
operator+( 5, 2 )
```

```
operator<<( cout, "Hello" )
```

- Evaluates to (returns) either a single value *or no value*

Example functions

- `pow(x, y)` calculates x^y
 - `pow(2.0, 3.0)` is 8.0
 - Input: two *parameters* `x` and `y` of type `double`
 - Output: returns a value of type `double`
- `sqrt(x)` calculates the nonnegative square root of `x`, for `x >= 0.0`
 - `sqrt(2.25)` is 1.5
 - Input: one parameter `x` of type `double`
 - Output: returns a value of type `double`

Example functions

- `floor(x)` calculates the largest whole number less than `x`
 - `floor(48.79)` is `48.0`
 - Input: one parameter `x` of type `double`
 - Output: returns a value of type `double`

Predefined functions

- Functions someone else wrote that you can use
- Predefined functions are organized into separate libraries
 - Stream I/O functions are in `iostream` library
 - Math functions are in `cmath` library
- Each library has a *header* file
- To use a predefined function, you `#include` the appropriate header file

Calling Functions

- Every function has 3 parts you need to know in order to use it (besides what it does, of course):
 - A name
 - Follows the same rules as variables names
 - Can't be the same as the name of a variable or a reserved word
 - *A parameter list*
 - These are the input values that the function needs in order to do its job
 - Each parameter is a specific data type (int, double, char, string, etc)
 - *A return type*
 - This is the data type that the function returns when it is done

Calling Functions

- Functions are *called* by name:

```
y = sqrt( x );
```

- When you call a function, you have to provide it with appropriate parameter values
 - Same number it expects to get
 - Same order
 - Same types
- We say these values are *passed in* to the function

Using the Return Value

- The functions we've looked at all *return* a value
 - Sometimes we call this the *output* of the function
 - This is different than output to the screen!
 - Return values aren't printed to the screen, they are returned to the calling statement
 - You could also say the function evaluates to its return value
- Some examples:

```
x = 3 + 4;
```

evaluates to: `x = 7;`

```
y = sqrt( 16.0 );
```

evaluates to: `y = 4;`

Function Calls and Return Values

- When calling a function, you typically:
 - Save the return value for further calculation
 - Use the return value in some calculation
 - Print the return value
- In other words, functions are called:
 - In an assignment statement
 - In an expression
 - As an actual parameter to another function

Example Predefined Functions

TABLE 6-1 Predefined Functions

Function	Header File	Purpose	Parameter(s) Type	Result
<code>abs (x)</code>	<code><cstdlib></code>	Returns the absolute value of its argument: <code>abs (-7) = 7</code>	<code>int</code>	<code>int</code>
<code>ceil (x)</code>	<code><cmath></code>	Returns the smallest whole number that is not less than <code>x</code> : <code>ceil (56.34) = 57.0</code>	<code>double</code>	<code>double</code>
<code>cos (x)</code>	<code><cmath></code>	Returns the cosine of angle <code>x</code> : <code>cos (0.0) = 1.0</code>	<code>double</code> (radians)	<code>double</code>
<code>exp (x)</code>	<code><cmath></code>	Returns e^x , where $e = 2.718$: <code>exp (1.0) = 2.71828</code>	<code>double</code>	<code>double</code>
<code>fabs (x)</code>	<code><cmath></code>	Returns the absolute value of its argument: <code>fabs (-5.67) = 5.67</code>	<code>double</code>	<code>double</code>

Example Predefined Functions

TABLE 6-1 Predefined Functions (continued)

Function	Header File	Purpose	Parameter(s) Type	Result
<code>floor(x)</code>	<code><cmath></code>	Returns the largest whole number that is not greater than <code>x</code> : <code>floor(45.67) = 45.00</code>	<code>double</code>	<code>double</code>
<code>pow(x, y)</code>	<code><cmath></code>	Returns <code>x^y</code> ; If <code>x</code> is negative, <code>y</code> must be a whole number: <code>pow(0.16, 0.5) = 0.4</code>	<code>double</code>	<code>double</code>
<code>tolower(x)</code>	<code><cctype></code>	Returns the lowercase value of <code>x</code> if <code>x</code> is uppercase; otherwise, returns <code>x</code>	<code>int</code>	<code>int</code>
<code>toupper(x)</code>	<code><cctype></code>	Returns the uppercase value of <code>x</code> if <code>x</code> is lowercase; otherwise, returns <code>x</code>	<code>int</code>	<code>int</code>

Exercises

1. Given three integer variables, `a`, `b` and `c`, write a C++ statement to find the greatest common denominator of `a` and `b`, and store it in `c`
 - How? Using a function!
 - The name of the function to compute the greatest common denominator is `gcd`
 - This function takes two integers as parameters
 - It returns a single integer
2. Given two string variables, `s1` and `s2`, write a C++ statement to assign `s2` the characters in `s1` in reverse order
 - How? Using a function!
 - The name of the function to reverse a string is `sreverse`
 - This function takes a string as its parameter
 - It returns a string

Exercises

3. Given three integer variables, `num1`, `num2` and `num3`, write a C++ statement to print the largest one
 - How? Using a function!
 - The name of the function to find the largest of three integers is `largest`
 - This function takes three integers as parameters
 - It returns a single integer

4. Given three integer variables, `num1`, `num2` and `num3`, write a C++ statement to print the largest one
 - This time using a different function
 - The name of the function to find the *larger* of *two* integers is `larger`
 - This function takes *two* integers as parameters
 - It returns a single integer

Side-effects

- Besides returning a value, functions can also have *side-effects*
 - For now, we'll focus on I/O related side-effects
- Examples:
 - Reading user input from a stream
 - Printing output to a stream
 - Drawing on the screen