



Quick Review of Functions



Overview of Functions

- Requires:
 - Function prototype
 - Informs compiler of what kind of input the function expects, what kind of output it generates, and the function name
 - Function definition
 - The actual code that is executed
 - Function call
 - Invoking the function from another function



Getting Data INTO a Function

- Use parameters to get data into a function
- Parameters specified in the function heading are the formal parameters
- Parameters specified in the function call are the actual parameters



Parameter Lists

- The formal parameter list is a comma-separated list of data type/parameter name pairs
- The actual parameter list is a comma-separated list of variables and/or expressions
- The prototype's parameter list must contain a comma-separated list of data types.
- All parameter lists must correspond in number and type.



Three Methods of Parameter Passing

- Pass-by-value
 - The actual parameter is evaluated completely, and the value is copied into a memory location created for the formal parameter in the function. Changes to the formal parameter in the function have no effect on any variables in the calling function.
 - This is the default parameter passing method



Three Methods of Parameter Passing

- Pass-by-reference
 - The actual parameter is a variable of the same data type as the formal parameter. The memory location for the formal parameter refers to the memory location of the actual parameter. Changes to the formal parameter in the function change the contents of the corresponding variable in the calling function.
 - To pass by reference, the data type of the formal parameter must be followed by the & character



Three Methods of Parameter Passing

- Pass-by-constant-reference
 - The actual parameter is a variable of the same data type as the formal parameter. The memory location for the formal parameter refers to the memory location of the actual parameter. Changes to the formal parameter in the function are not allowed
 - To pass by constant reference, the data type of the formal parameter must be preceded by the keyword **const**.



Getting Data OUT OF a Function

- Give the function a data type other than void, and use a return statement in the function.
 - Generally used when returning/modifying the value of one variable, or when the function computes a value to be used in another expression.
- Give the function a void return value, and use reference parameters to change the value of one or more variables in the calling function.
 - Generally used when the function must “return” or modify more than one variable value in the calling function.



Scope of Identifiers

- The scope of an identifier is the location in the program where its use is valid.
- Identifiers can only be used in a block if they have been declared (variables and constants) or their prototype specified (functions)



Scope of Identifiers

- Global scope
 - Identifiers declared before all function definitions can be used in any function.
 - VERY BAD DESIGN CHOICE!!!
- Local scope
 - Identifiers declared within a block (usually a function block) are valid only within that block.
 - The block for a function includes the formal parameter list.
- Breaking a scope “tie”
 - If an identifier has both local and global scope, the local scope always takes precedence.