Call-by-Value Parameters

- Copy of actual argument passed
- Considered “local variable” inside function
  - Function has no access to “actual argument” from caller
- This is the default method
  - Used in all examples so far

---

**Parameters**

- Two methods of passing arguments to parameters
  - Call-by-value
    - Also called Pass-by-value
    - “copy” of value is passed
  - Call-by-reference
    - Also called Pass-by-reference
    - “address of” argument is passed
    - Cannot be used when passing literals
**Call-by-Reference Example**

```cpp
//Program to demonstrate call-by-reference parameters.
#include <iostream>

using namespace std;

void getNumbers(int& input1, int& input2);
    //Reads two integers from the keyboard.

void swapValues(int& variable1, int& variable2);
    //Interchanges the values of variable1 and variable2.

void showResults(int output1, int output2);
    //Shows the values of variable1 and variable2, in that order.

int main()
{
    int firstNum, secondNum;
    getNumbers(firstNum, secondNum);
    swapValues(firstNum, secondNum);
    showResults(firstNum, secondNum);
    return 0;
}
```

**Call-by-Value Example**

```cpp
double fee(int hoursWorked, int minutesWorked)
{
    int quarterHours;
    minutesWorked = hoursWorked*60 + minutesWorked;
    quarterHours = minutesWorked/15;
    return quarterHours*RATE;
}
```

**Sample Dialogue**

Welcome to the law office of Dewey, Cheatham, and Howe. 
The law office with a heart.
Enter the hours and minutes of your consultation:
5 46
For 5 hours and 46 minutes, your bill is $3450.00

**Call-by-Reference**

- Used to provide access to caller's actual argument
- Caller's data can be modified by called function
- Typically used for input function
  - To retrieve data for caller
  - Data is then “given” to caller
- Specified by ampersand, &, after type in parameter list e.g. int& x;
Mixed Parameter Lists

- Can combine passing mechanisms
- Parameter list can include
  - Pass-by-value
  - Pass-by-reference
  - Const pass-by reference
- Order of arguments in function call is critical
  - Pay attention to the order defined in the function header

Parameter Names

- Same rules as naming variables
  - Must be meaningful
  - Should reflect its use
- Functions are self-contained modules
  - Designed separately from rest of program
  - Parameter names can be same as argument names
- Function names should also be meaningful

Call-by-Reference Details

- What's really passed in?
- A “reference” back to caller's argument
  - Reference to memory location of argument
  - Called “memory address”
  - Unique number referring to a distinct place in memory

Constant Reference Parameters

- Reference arguments can be dangerous
  - Caller's data can be changed
  - Often this is desired, sometimes not
- To protect data & still pass-by-reference
  - Use const keyword
  - No copy is made, only reference is passed
  - Argument is now read-only

```c
void displayData(const int& value1, const int& value2);
```