Chapter 4

Parameters

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-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

Display 4.1  Formal Parameter Used as a Local Variable

```cpp
//Law office billing program.
#include <iostream>
using namespace std;

const double RATE = 150.00; //Dollars per quarter hour.

double fee(int hoursWorked, int minutesWorked);
//Returns the charges for hoursWorked hours and
//minutesWorked minutes of legal services.

int main()
{
    int hours, minutes;
    double bill;
    cout << "Welcome to the law office of\n" << "Dewey, Cheatham, and Howe.\n" << "The law office with a heart.\n" << "Enter the hours and minutes" << " of your consultation:\n";
    cin >> hours >> minutes;
    bill = fee(hours, minutes);
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "For " << hours << " hours and " << minutes << " minutes, your bill is $" << bill << endl;
    return 0;
}
```

The value of minutes is not changed by the call to fee.
Call-by-Value Example

```
Display 4.1  Formal Parameter Used as a Local Variable
26    double fee(int hoursWorked, int minutesWorked) {
27        int quarterHours;
28    
29        minutesWorked = hoursWorked*60 + minutesWorked;
30        quarterHours = minutesWorked/15;
31        return (quarterHours*RATE);
32    }
```

**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;
Call-by-Reference Example

Display 4.2  Call-by-Referencce Parameters

```cpp
1 //Program to demonstrate call-by-reference parameters.
2 #include <iostream>
3 using namespace std;
4
4 void getNumbers(int& input1, int& input2);
5 //Reads two integers from the keyboard.
6 void swapValues(int& variable1, int& variable2);
7 //Interchanges the values of variable1 and variable2.
8 void showResults(int output1, int output2);
9 //Shows the values of variable1 and variable2, in that order.
10
10 int main()
11 {
12    int firstNum, secondNum;
13    getNumbers(firstNum, secondNum);
14    swapValues(firstNum, secondNum);
15    showResults(firstNum, secondNum);
16    return 0;
17 }
18
18 void getNumbers(int& input1, int& input2)
19 {
20    cout << "Enter two integers: ";
21    cin >> input1
22      >> input2;
23 }
24
24 void swapValues(int& variable1, int& variable2)
25 {
26    int temp;
27    temp = variable1;
28    variable1 = variable2;
29    variable2 = temp;
30 }
31
32 void showResults(int output1, int output2)
33 {
34    cout << "In reverse order the numbers are: 
35        " << output1 << " " << output2 << endl;
36 }
```

Display 4.2  Call-by-Reference Parameters

**SAMPLE DIALOGUE**

Enter two integers: 5 6
In reverse order the numbers are: 6 5
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

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