Arguments

Passing Variables by *VALUE* and *REFERENCE*

Jessica Sizemore, SIP 2015, 24 JUN 2015

Void Functions (Review)

- **Void functions** performs tasks but does not return a value
- A Void function may be used to do something like display information on the scree
- Header begins with keyword `void`, instead of a return data type
- Function body does not contain a `return` statement
- Call a `void` function by including its name and actual arguments (if any) in a statement
- Call to a void function appears as a self-contained statement, not part of another statement

Passing Variables to a Function

- You can pass a variable's value or its address
- Passing a variable's value is referred to as **passing *by value***,
- Passing a variable's address is referred to as **passing *by reference***
- It depends on whether the receiving function should have access to the variable in memory

- Passing *by* ***value*** WILL NOT permit the function to change the contents of the variable
- Passing *by* ***reference*** WILL permit the function to change the contents of the variable

Passing Variables by Value

- Passing a variable ***by value*** means that only a copy of the variable's contents is passed, not the address of the variable
- The receiving function cannot change the contents of the variable

- It is best to pass ***by value*** when the receiving function needs to know the value of the variable but does not need to change it

Passing Variables by Value

```
#include <iostream>
using namespace std;

float LBStoKG(float); // DECLARATION

int main () {

  float lbs;

  cout << "\nEnter your weight in pounds(lbs): ";
  cin << lbs;
  cout << "Your weight in kiligrams is " << LBStoKG(lbs) << endl;

}

float LBStoKG (float pounds)
{
  return (0.453592 * pounds);
}
```

- You are returning a value that is specific to this function.

Passing Variables by Reference

- Passing a variable's address to a function is referred to as passing ***by reference***
- When you want the receiving function to change the contents of the variable

- You include an **ampersand (&)** before the name of the formal parameter in the receiving function's header
- Ampersand (&) is the **address-of operator**
  - Tells the computer to pass the variable's address rather than a copy of its contents

Passing Variables by Reference

```
#include <iostream>
using namespace std;

int main () {

  void order (int&, int&);

  int n1 = 99, n2 = 11;
  int n3 = 22, n4 = 88;

  order (n1, n2);
  order (n3, n4);

  cout << "n1 = " << n1 << endl;
  cout << "n2 = " << n2 << endl;
  cout << "n3 = " << n3 << endl;
  cout << "n4 = " << n4 << endl;

}
```

- Void functions use variables passed *by reference* to send information back to the calling function, instead of a `return` value

```
void order (int& num1, int& num2)
{
  if (num1 > num2)
    {
      int temp = num1;
      num1 = num2;
      num2 = temp;
    }
}
```

Worked Cited

- Robert Lafore, "Object-Oriented Programming in C++, Third Edition",
  - Chapter 5 – Functions.
- Diane Zax, "An Introduction to Programming with C++, Sixth Edition",
  - Chapter 10 – Void Functions.
- Towson University, Professor Robert Eyer, COSC 175,
  - Chapter 10 Lecture Slides..