Name: Christopher Carl
Email: ccarl2@fau.edu
Date: 16.07.14
Z Number: Z23146703

LAB TITLE: Function Calls

Question 1: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 1? Explain your answer.

No. The variables are all locally scoped to the function only. The swap int a and int b are compiled to an anonymous function with anonymous variable names as parameters, then are compiled to assembly as a custom tailored single execution statement that replaces all of those anonymous variables with the actual numerical values.

Question 2: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 1? Explain your observations.

No. They were not. Int a and int b in the main function persist with the same value as they were first initialized because the swap function was a pass by value function. Therefore, any modifications to the variables passed into the function stay within the function.

Question 3: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 2 (hint: are they pass by value or by address)? Please explain your answer.

Yes. The swap function accesses the variables directly by using their memory address and replacing the contents of each memory address of int a and int b with each other. The value is not the variable itself. The variable is the address.

Question 4: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 2? Explain your observations.

Yes. The values stored in the variables were correctly swapped because the actual variable was being accessed using its address rather than a making a “copy” or “ghost” of it to have its value.

Question 5: What is the purpose of the ‘&’ and ‘*’ symbols in the main and swap functions in the program in Step 3? Explain your answer.

The answers lies in the verbiage used to describe them. The third swap takes in two addresses as parameters and access the variable contents directly when performing the swap operation.

Question 6: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 3? Explain your answer.

Yes. The swap function accesses the variables directly by using their memory address and replacing the contents of each memory address of int a and int b with each other. The value is not the variable itself. The variable is the address.

Question 7: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 3? Explain your observations.

Yes. The values stored in the variables were correctly swapped because the actual variable was being accessed using its address rather than a making a “copy” or “ghost” of it to have its value.

Question 8: What types of function calls (reference, value, etc.) were used in the programs in Steps 1, 2 and 3?

Step 1: Pass by value

Step 2: Pass by reference

Step 3: Pass by pointer

LAB TITLE: Declaration of Functions

Ans:

Question 1: What are the differences between the two function declarations shown above?

The function declaration that was first shown is just more descriptive, while the second declaration is much more abstract and can only be useful for function prototyping rather than creating a full function.

Question 2: Based on our previous lab, we learned about data types and the size of space allocated in memory [for instance an **int** may take up 4 bytes of space]. Based on this knowledge and different ways to write a prototype, what observations can you make regarding how a compiler "knows" how much space to allocate?

The size of the function that the compiler allocates is determined by the type of the parameters, not by the name of the parameters used within the function.

Question 3: What specific information does the compiler need to perform a successful function call?

The information required include: formal parameters and local arguments, so it can calculate the memory allocation and create the custom assembly subroutine.

Question 4: Write the header and signature for the following function. Also write the two forms of its prototype.

Header: double Print_Message(double balance, double limit);
Signature: Print_Message(double balance, double limit);
Prototype 1: double Print_Message(double balance, double limit);
Prototype 2: double Print_Message(double, double);

LAB TITLE: Function Overloading

Ans:

Question 1: Please execute the program in Step 1 and explain how function overloading was used?

Function overloading was used in order to preserve programming readability. By making three functions with the name swap and have different sets of formal parameters, the program can be written to swap different data using the seemingly same function.

Question 2: Without changing the original program, would the following function cause an error if added in the source file that contains the program in Step 1? If so, list the error message(s), and explain why the error(s) occurred.

Yes, it would cause an error that would prevent program compilation. The replaced function by itself is sound, but it conflicts with another function with a different return type with the same formal parameters. Function overloading cannot be differentiated by return type alone. The compiler would return a message that reports a similar problem.